Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КАЛИНИНГРАДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ»

В. В. Подтопельный

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ ОТКРЫТЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

Учебно-методическое пособие по выполнению лабораторных работ по дисциплине для студентов специальности
10.05.03 «Информационная безопасность автоматизированных систем»
специализация «Безопасность открытых информационных систем»

Калининград
Издательство ФГБОУ ВО «КГТУ»
2026

УДК 004.056.5(075)

Рецензент
кандидат физико-математических наук, доцент, доцент кафедры информационной безопасности института информационных технологий ФГБОУ ВО «Калининградский государственный технический университет» Н. Я. Великите.

Подтопельный, В. В.
Информационная безопасность открытых информационных систем: учебно-методическое пособие по выполнению лабораторных работ по дисциплине для студентов специальности 10.05.03 «Информационная безопасность открытых информационных систем», специализация «Безопасность открытых информационных систем». – Калининград: Изд-во ФГБОУ ВО «КГТУ», 2026. – 120 с.

Учебно-методическое пособие включает в себя набор лабораторных работ по дисциплине «Информационная безопасность открытых информационных систем». В учебно-методическом пособии приведен тематический план изучения дисциплины. Представлены методические указания лабораторных работ по дисциплине. Пособие предназначено для студентов 5 курса специальности 10.05.03 «Информационная безопасность автоматизированных систем» и смежных специальностей
Табл. 4, список лит. – 15 наименований

Учебно-методическое пособие рекомендовано к использованию в учебном процессе в качестве локального электронного методического материала методической комиссией института цифровых технологий ФГБОУ ВО «Калининградский государственный технический университет» от 29 сентября 2026 г., протокол № 7

УДК УДК 004.056.5(075)

© Федеральное государственное бюджетное образовательное учреждение высшего образования «Калининградский государственный технический университет».
© Подтопельный В. В., 2026 г.

ОГЛАВЛЕНИЕ

Введение……..4
Лабораторная работа № 1. Изучение трафика…….8
Лабораторная работа № 2. Nmap-сканирование……15
Лабораторная работа № 3. «Изучение работы инструмента hping3, организация стресс-теста сети с его помощью. Настройка брандмауэра с помощью ufw и iptables»……24
Лабораторная работа № 4. Аудит сетевой инфраструктуры средствами PowerShell……..34
Лабораторная работа № 5. Исследование современных средств поиска уязвимости в ОС……43
Лабораторная работа № 6. Основы настройки и эксплуатации системы обнаружения вторжений Snort…….51
Лабораторное задание № 7. Работа с препроцессорами SNORT……..63
Лабораторное работа № 8. Сигнатурный анализ и обнаружение аномалий…...70
Лабораторная работа № 9. Аудит информационной безопасности веб-приложений……84
Лабораторная работа № 10. OSSEC HIDS…….95
Лабораторная работа № 11. Настройка работы IPsec…….104
Лабораторная работа № 12. Программное логирование сетевых пакетов в файл…….110
Заключение……...……...118
Литература…….....119

ВВЕДЕНИЕ

Данное учебно-методическое пособие предназначено для студентов специальности 10.05.03 «Информационная безопасность автоматизированных систем», специализация «Безопасность открытых информационных систем», изучающих дисциплину «Информационная безопасность открытых информационных систем».

Целью освоения дисциплины «Информационная безопасность открытых информационных систем» является: выявление уязвимостей и угроз безопасности, их нейтрализацию в открытых информационных системах (в их программном обеспечении, аппаратных средствах, средствах связи), в том числе при взаимодействии с удаленными системами, разработку и внедрение открытых систем в защищенном исполнении, а также средств защиты для них, обеспечение контроля и управления установленными средствами защиты информации.

Цель изучения лабораторного курса дисциплины: обучить студентов выявлять и анализировать угрозы и уязвимости инфраструктур цифрового типа на предприятиях, определять особенности поведения злоумышленников в распределенных системах обработки информации.

Лабораторный практикум содержит 12 лабораторных работ.

Компетенции:

– ОПК-5. Способен применять нормативные правовые акты, нормативныеи методические документы, регламентирующие деятельность по защите информации:

– ОПК-5.1. Способен разрабатывать и реализовывать политику информационной безопасности открытых информационных систем;

– ОПК-5.2. Способен разрабатывать и эксплуатировать системы защиты информации открытых информационных систем.

В результате выполнения лабораторных заданий обучающийся с учетом освоения материала всего курса дисциплины должен освоить и сформировать навыки, знания, умения, а именно:

знать:

– общие принципы построения открытых систем в защищенном исполнении (на лабораторных осваивается материал: концепция диспетчера доступа);

– особенности политики безопасности и способы ее внедрения на предприятии (на лабораторных осваивается материал: методы и средства ограничения доступа к ресурсам распределенной ВС);

– основные этапы процесса проектирования и методы, используемые припостроении проектируемой системы и способы нарушения

информационной безопасности при работе автоматизированных систем обработки информации (на лабораторных осваивается материал: методы и средства обнаружения уязвимостей распределенной ВС; методы и средства обнаружения атак на ресурсы распределенной ВС; методы и средства противодействия атакам на ресурсы распределенной ВС);

уметь:

– выявлять возможные способы нарушения информационной безопасности при работе автоматизированных систем (открытых ИС) обработки информации и описывать их с учетом методических рекомендаций регуляторов в областизащиты информации (на лабораторных учится: организации защиты распределенной ВС; организации защиты от атак на ресурсы распределенной ВС; защиты программ от изменений в распределенной ВС; осуществлять контроль трафика в рамках распределенной ВС);

– выявлять известные уязвимости открытых информационных систем (на лабораторных учится: осуществлять контроль трафика в рамках распределенной ВС);

владеть:

– навыком формирования перечня мероприятий по предотвращению угрозбезопасности информации автоматизированных систем (открытых ИС);

– навыком разработки модели угроз безопасности информации и нарушителей в автоматизированных системах (на лабораторных овладеваетследующиминавыками: правлениясредствами защиты в распределенной ВС от несанкционированного доступа и нарушения функциональности ее подсистем, средствами борьбы с атаками злоумышленников на ресурсы серверов баз данных; методикой контроля информационной целостности в распределенной ВС).

Программное обеспечение

1. MicrosoftDesktopEducation. Операционные системы: MicrosoftWindowsDesktopoperatingsystems, офисные приложения (MicrosoftOffice), по соглашению V9002148 OpenValueSubscription (срок действия: три года)

2. Программное обеспечение, распространяемое по лицензии GNU GeneralPublicLicense (лицензия на свободное программное обеспечение, созданная в рамках проекта GNU, по указанной автор передаёт программное обеспечение в общественную собственность):

– Snort (свободная сетевая система предотвращения вторжений (IPS) и обнаружения вторжений (IDS) с открытым исходным кодом);

– Ethereal (программы перехвата и анализа сетевых пакетов, доусти использование Wireshark), commviewdemo (или данного аналог);

– NMAP (программа сканирование сетевых ресурсов);

– MySQL (система управления базами данных).

Типовое ПО для всех ПК:

1. Microsoft DesktopEducation (операционные системы MicrosoftWindowsDesktopoperatingsystem, офисные приложения MicrosoftOffice, по соглашению V9002148 OpenValueSubscription). Дата заключения контракта 05.07.2018. Номер контракта 0335100016118000073-0484577-02.

2. Антивирусное программное обеспечение KasperskyTotalSpaceSecurityRussianEdition, лицензия 17EO-171225-104659-470-270, срок использования с 2017-12-26 до 2020-03-13.

Специализированное ПО:

1.VMWare Workstation, Страж-NT, Панцирь-К (по государственному контракту №10/13А от 19 апреля 2013 года), (на два компьютера – VmwareLicensePurchaseInformation № 22033811ОВ); OpenValueSubscription.

Таблица 1 – Шкала оценок уровня

Оценка			
Неудовлетво-рительный	**Пороговый**	**Углубленный**	**Продвинутый**
«2» (неудовлетво-рительно)	**«3» (удовлетво-рительно)**	**«4» (хорошо)**	**«5» (отлично)**
Работа выполнена в полном объеме. Отчет не оформлен и представлен. При защите отчетных материалов правильные ответы даны менее чем на 50 % включительно. Материал излагается непоследова-тельно, сбивчиво, не представляет	Работа выполнена в полном объеме. Отчет оформлен и представлен. При защите отчетных материалов правильные ответы на 51–64 % вопросов. Допускаются нарушения в последователь-ности изложения. Демонстрируются поверхностные знания вопроса.	Работа выполнена в полном объеме. Отчет оформлен и представлен. При защите отчетных материалов правильные ответы на 65–94 % вопросов. Ответы на поставленные вопросы излагаются систематизировано и последовательно. Материал излагается	Работа выполнена в полном объеме. Отчет оформлен и представлен. При защите отчетных материалов правильные ответы даны на 95–100 % вопросов. Ответы на поставленные в билете вопросы излагаются логично, последовательно

Оценка			
Неудовлетво-рительный	**Пороговый**	**Углубленный**	**Продвинутый**
«2» (неудовлетво-рительно)	**«3» (удовлетво-рительно)**	**«4» (хорошо)**	**«5» (отлично)**
определенной системы знаний по работе	Имеются затруднения с выводами. Допускаются нарушения норм литературной речи	уверенно. Демонстрируется умение анализировать материал, однако не все выводы носят аргументиро-ванный и доказательный характер	и не требуют дополнительных пояснений. Делаются обоснованные выводы. Демонстрируют-ся глубокие знания предмета

ЛАБОРАТОРНАЯ РАБОТА № 1. ИЗУЧЕНИЕ ТРАФИКА

Общие сведения

Цель: получение навыков обследования организации при проведении аудита с помощью снифера информационной безопасности

Программное обеспечение:среда виртуализации, дистрибутивы BacktrackLinux или KaliLinux, среда выявления и эксплуатации уязвимостей MetasploitFramework, среда тестирования защищенности веб-приложений BurpSuite, среда эксплуатации уязвимостей веб-приложений BeEF, небезопасные веб-приложения проекта WebGoat, образы небезопасных веб-приложений проекта PentesterLab, современные веб-браузеры (InternetExplorer, GoogleChrome, MozillaFirefox), ПО Wireshark, программные средства инструментального анализа защищенности (XSpider, MaxPatrol ЗАО Позитив Текнолоджис или аналогичных программ), а также специализированные сканеры уязвимостей веб-приложений (например, Acunetix, AppScan, BurpSuitePro, W3AF).

Методический материал по выполнению лабораторных заданий

Wireshark представляет собой современный анализатор сетевых протоколов с открытым исходным кодом, широко применяемый для диагностики сетей, разработки и тестирования программного обеспечения, а также в образовательных целях и при проведении расследований инцидентов информационной безопасности. Программа поддерживает захват пакетов в реальном времени с сетевых интерфейсов, чтение ранее сохранённых файлов захвата в форматах pcap, pcapng, а также декодирование более трёх тысяч протоколов всех уровней сетевой модели OSI.

Архитектура Wireshark построена на библиотеке libpcap (в Unix-подобных системах) или WinPcap/Npcap (в Windows) для непосредственного захвата пакетов, и на собственной системе диссекторов, обеспечивающей декодирование полей протоколов. Программа предоставляет три основных панели: список пакетов с краткой информацией, детальное дерево декодирования протоколов и шестнадцатеричный дамп пакета с подсветкой выбранных полей. Поддержка фильтров захвата (BPF-синтаксис) и фильтров отображения (собственный язык выражений) позволяет эффективно работать с большими объёмами трафика.

Современные версии Wireshark (4.x) добавили поддержку расшифровки TLS 1.3 при наличии ключей сеанса, декодирование протокола HTTP/2 и HTTP/3 (QUIC), анализ трафика современных облачных протоколов и протоколов контейнерных оркестраторов. Программа интегрируется с инструментами tshark и dumpcap для работы из командной строки, что

особенно полезно при автоматизации задач анализа трафика на серверах без графического интерфейса.

С позиции информационной безопасности Wireshark применяется при расследовании инцидентов, анализе вредоносного трафика, проверке корректности работы криптографических протоколов, выявлении утечек данных и аномалий сетевой активности. Однако следует учитывать, что в современных сетях значительная часть трафика зашифрована (TLS), и для полноценного анализа содержимого сессий требуется доступ к ключам шифрования либо применение методов анализа метаданных (размер пакетов, временные интервалы, направление передачи) без декодирования полезной нагрузки.

Для эффективной работы с Wireshark необходимо понимать принципы инкапсуляции протоколов, структуру заголовков Ethernet, IP, TCP, UDP, ICMP, а также особенности работы протоколов прикладного уровня. Особое внимание следует уделять корректному выбору сетевого интерфейса для захвата, настройке режима promiscuousmode, применению фильтров для ограничения объёма записываемых данных и использованию возможностей раскраски пакетов для визуального выделения аномалий.

Wireshark – программа для анализа протоколов (анализатор пакетов), используемая для поиска и устранения неполадок в сети, анализа, разработки программного обеспечения и протоколов, а также обучения. По мере движения потоков данных по сети анализатор «захватывает» каждую единицу данных протокола (PDU), после чего расшифровывает или анализирует содержимое согласно соответствующему документу RFC или другим спецификациям.

Wireshark – полезный инструмент для всех, кто работает с сетями. Указанную программу можно использовать для анализа данных, а также для поиска и устранения неполадок при выполнении большинства лабораторных работ в рамках курсов CCNA. В ходе лабораторной работы будет освоено использование программы Wireshark для захвата IP-адресов пакетов данных ICMP и MAC-адресов Ethernet-кадров.

Методические указания и порядок выполнения работы. Задание

Цель лабораторной работы связана с освоением методики захвата, фильтрации и анализа сетевого трафика с применением анализатора Wireshark, изучением структуры заголовков базовых протоколов стека TCP/IP и приобретение навыков выявления характерных признаков сетевых явлений по содержимому пакетов.

В рамках лабораторной работы выполняются следующие задачи:

1. Освоение интерфейса Wireshark и настройка параметров захвата трафика.

2. Захват и анализ ICMP-трафика, формируемого диагностическими утилитами ping и traceroute.

3. Изучение структуры заголовков протоколов Ethernet, IP, ICMP, ARP, TCP, UDP.

4. Применение фильтров захвата и фильтров отображения для выделения интересующих пакетов.

5. Анализ процесса установления TCP-соединения (трёхстороннее рукопожатие) и сеансов HTTP.

6. Выявление характерных признаков фрагментации IP-дейтаграмм и сборка фрагментов.

7. Анализ трафика современных протоколов (HTTP/2, TLS, DNS over HTTPS) при возможности.

8. Сохранение результатов захвата в файлы формата pcapng для последующего анализа.

9. Изучить принципы сниффинг с использованием фильтров Wireshark/Ethereal

10. Захват и анализ кадров Ethernet с помощью программы Wireshark

Задание 1. Установка и настройка Wireshark, освоение интерфейса

1. Скачайте актуальную версию Wireshark с официального сайта wireshark.org. При установке в Windows обязательно установите драйвер Npcap (современная замена WinPcap) с поддержкой rawsocketcapture. В Linux установите пакет: sudoaptinstallwireshark, после чего добавьте пользователя в группу wireshark для работы без прав root: sudousermod -aGwireshark $USER.

2. Запустите Wireshark. В главном окне изучите структуру интерфейса: панель списка интерфейсов захвата (начальная страница), панель фильтра отображения, три рабочие панели (список пакетов, дерево декодирования протоколов, шестнадцатеричный дамп), а также строку состояния с информацией о захваченных пакетах.

3. Выберите сетевой интерфейс для захвата, щёлкнув по его имени на начальной странице. Перед началом захвата нажмите на значок шестерёнки рядом с интерфейсом для настройки параметров. В открывшемся окне проверьте следующие опции: активирован режим promiscuous (захват всех пакетов в сегменте сети, а не только адресованных данному узлу); размер буфера захвата установлен не менее 1 Мб; снимите флажок с опции ограничения размера каждого пакета (Limiteachpacketto) для полного захвата; при необходимости задайте фильтр захвата в формате BPF (например, host 192.168.1.1 для захвата только трафика с указанным узлом).

4. Запустите захват, нажав кнопку Start. В строке состояния убедитесь, что счётчик захваченных пакетов увеличивается. Остановите захват через 30–60 секунд кнопкой Stop (красный квадрат на панели инструментов).

5. Изучите цветовую раскраску пакетов: Wireshark автоматически применяет цветовые правила для различных протоколов (TCP – светло-синий, UDP – светло-зелёный, HTTP – зелёный, DNS – тёмно-синий, TLS – светло-серый). Настройте собственные правила раскраски через меню View → ColoringRules для выделения пакетов, представляющих интерес для анализа.

Задание 2. Захват и анализ ICMP-трафика

1. Очистите ARP-кэш операционной системы для формирования полного цикла ARP-запрос/ответ перед ICMP-обменом. В Windows выполните: netshinterfaceipdeletearpcache. В Linux выполните: sudoipneighflushall.

2. Запустите захват трафика в Wireshark на выбранном интерфейсе. В строке фильтра отображения введите icmp и нажмите Enter– в списке будут отображаться только ICMP-пакеты.

3. В командной строке выполните ping с указанием IP-адреса соседнего хоста или шлюза по умолчанию: ping -n 4 192.168.1.1 (Windows) или ping -c 4 192.168.1.1 (Linux). Дождитесь завершения команды и остановите захват в Wireshark.

4. Проанализируйте захваченные ICMP-пакеты. Для каждого пакета определите: тип ICMP (Type 8 –EchoRequest, Type 0 –EchoReply); значения полей Identifier и SequenceNumber, позволяющие сопоставить запросы и ответы; значение поля TTL в заголовке IP (для определения количества пройденных маршрутизаторов); размер пакета и полезной нагрузки.

5. Выберите первый ICMP-запрос в списке пакетов. В панели дерева декодирования последовательно разверните заголовки: Ethernet II – определите MAC-адреса источника и назначения, а также поле EtherType (0x0800 для IPv4); InternetProtocolVersion 4 – определите IP-адреса, поле TTL, поле Protocol (1 для ICMP), общую длину дейтаграммы; InternetControlMessageProtocol– определите Type, Code, Checksum, Identifier (BE и LE), SequenceNumber (BE и LE) и данные.

6. Сопоставьте значения полей Identifier и SequenceNumber в запросах и ответах. Убедитесь, что каждому EchoRequest соответствует EchoReply с теми же значениями Identifier и SequenceNumber. Зафиксируйте в отчёте значения всех полей для одной пары запрос-ответ.

7. Сохраните захваченный трафик в файл формата pcapng: File → SaveAs, укажите имя файла lab1_icmp.pcapng. Формат pcapng поддерживает аннотации, несколько интерфейсов захвата и расширенные метаданные.

Задание 3. Анализ ARP-трафика

1. Очистите ARP-кэш: netsh interface ip delete arpcache (Windows) или sudoip neigh flush all (Linux).

2. Запустите захват трафика. В строке фильтра отображения введите arp и нажмите Enter.

3. Выполните ping IP-адреса соседнего хоста, с которым ранее не было обмена. Поскольку ARP-кэш очищен, перед ICMP-обменом будет выполнен ARP-запрос. Остановите захват после получения ответов на ping.

4. Найдите в списке ARP-запрос (Opcode: request, whohas<IP>? tell<IP>) и ARP-ответ (Opcode: reply, <IP>isat<MAC>). Развернитезаголовок Address Resolution Protocol ипроанализируйтеполя: Hardware type, Protocol type, Hardware size, Protocol size, Opcode, Sender MAC Address, Sender IP Address, Target MAC Address, Target IP Address.

5. Сопоставьте значения полей Sender MAC/IP в запросе с MAC/IP-адресами отправителя, а Target MAC/IP – с адресами получателя. В ARP-запросе поле Target MAC Address обычно заполнено нулями (00:00:00:00:00:00), так как адрес запрашивается. В ARP-ответе все поля заполнены фактическими адресами.

6. Сохраните трафик в файл lab1_arp.pcapng. Зафиксируйте в отчёте значения всех полей ARP-заголовка для запроса и ответа.

Задание 4. Анализ установления TCP-соединения и HTTP-сессии

1. Запустите захват трафика без фильтра. Откройте веб-браузер и перейдите по адресу http://<IP-адрес_тестового_сервера> (используйте HTTP, а не HTTPS, для возможности анализа содержимого). Остановите захват после загрузки страницы.

2. В строке фильтра отображения введите tcp.streameq 0 для отображения первого TCP-потока. Найдите последовательность из трёх пакетов, соответствующих трёхстороннему рукопожатию TCP: SYN (флаги: SYN, SequenceNumber=0), SYN-ACK (флаги: SYN, ACK, SequenceNumber=0, AcknowledgmentNumber=1), ACK (флаги: ACK, SequenceNumber=1, AcknowledgmentNumber=1).

3. Выберите пакет SYN и проанализируйте заголовок TCP: определите порты источника и назначения, начальный последовательный номер (InitialSequenceNumber), размер окна TCP, набор опций TCP (MSS, WindowScale, SACK Permitted, Timestamps). Зафиксируйте значения всех полей.

4. Найдите HTTP-запрос GET в составе TCP-потока. Разверните заголовок HypertextTransferProtocol и определите метод запроса, URI, версию HTTP, заголовки (Host, User-Agent, Accept и другие). Найдите HTTP-ответ и определите код состояния (200 OK), версию HTTP, заголовки ответа (Content-Type, Content-Length, Server).

5. Примените фильтр tcp.flags.syn == 1 andtcp.flags.ack == 0 для отображения всех SYN-пакетов. Подсчитайте количество установленных TCP-соединений. Сохраните трафик в файл lab1_tcp_http.pcapng.

Задание 5. Анализ фрагментации IP-дейтаграмм

1. Запустите захват трафика. Выполните ping с размером пакета, превышающим MTU (обычно 1500 байт для Ethernet): ping -l 3500 -n 1 192.168.1.1 (Windows) или ping -s 3500 -c 1 192.168.1.1 (Linux). Остановите захват.

2. Примените фильтр ip.src == <ваш_IP>orip.dst == <ваш_IP>. Найдите фрагментированные IP-пакеты (в столбце Info отображается фрагмент: offset и Morefragments).

3. Для каждого фрагмента разверните заголовок IP и определите: поле Identification (одинаковое для всех фрагментов одной дейтаграммы); поле Flags (бит MoreFragments установлен для всех фрагментов, кроме последнего); поле FragmentOffset (смещение данных фрагмента в исходной дейтаграмме); общую длину каждого фрагмента.

4. Сопоставьте количество фрагментов и их размеры с теоретическими положениями фрагментации IP. При размере полезной нагрузки 3500 байт и MTU 1500 байт дейтаграмма должна быть разбита на три фрагмента: первый размером 1480 байт данных (смещение 0), второй размером 1480 байт (смещение 1480), третий размером 540 байт (смещение 2960).

5. Сохраните трафик в файл lab1_fragmentation.pcapng. Зафиксируйте в отчёте значения полей для каждого фрагмента и схему сборки исходной дейтаграммы.

Задание 6. Анализ DNS-трафика

1. Запустите захват трафика. В строке фильтра введите dns. Очиститекэш DNS: ipconfig /flushdns (Windows) илиsudosystemd-resolve --flush-caches (Linux).

2. Откройте веб-браузер и перейдите по новому доменному имени (например, http://example.com). Остановите захват.

3. Найдите DNS-запрос типа A (Address) для запрошенного домена. Развернитезаголовок Domain Name System иопределите: Transaction ID, Flags (query/response, recursion desired), Questions, Answer RRs. В разделе Queries определите запрошенное имя и тип записи.

4. Найдите DNS-ответ и определите разрешённый IP-адрес в разделе Answers. Обратите внимание на значение TTL DNS-записи (не путать с TTL в IP-заголовке) – это время кэширования записи.

5. Проверьте наличие DNS over HTTPS (DoH) трафика, который передаётся по TCP-порт 443 и не виден в DNS-запросах. При использовании DoH DNS-запросы инкапсулируются в TLS-сессии. Сохраните трафик в файл lab1_dns.pcapng.

Задание 7. Анализ трафика traceroute

1. Запустите захват трафика. Выполните трассировку маршрута: tracert example.com (Windows) или traceroute example.com (Linux). Остановите захват после завершения трассировки.

2. Примените фильтр icmp. Проанализируйте ICMP-сообщения: EchoRequest (Type 8) с увеличивающимся значением TTL, ICMP TimeExceeded (Type 11) от промежуточных маршрутизаторов, EchoReply (Type 0) от целевого узла.

3. Определите количество промежуточных маршрутизаторов по количеству ICMP TimeExceeded сообщений. Для каждого маршрутизатора зафиксируйте IP-адрес и время отклика. Сопоставьте результаты с теоретической моделью работы traceroute.

4. Сформируйте итоговый отчёт по результатам выполнения всех заданий лабораторной работы. Отчёт должен содержать: описание выполненных действий, значения наблюдаемых полей протоколов, скриншоты ключевых пакетов из Wireshark, выводы по каждому заданию.

Индивидуальное задание

Студент самостоятельно выбирает инфраструктуру организации для рассмотрения при решении лабораторных задач.

Требования к отчету и защите

Защита предполагает опрос по материалу, изложенному в отчете.
Отчет должен быть оформлен: требуется наличие титульного листа, введение (содержит: указание цели лабораторной работы, задачи, краткие теоретические сведения), последовательное изложение разработанного материала, подписанные скриншоты хода работы, если использовалась компьютерная техника, заключение (выводы, анализ полученных результатов).

Проверяется знание теоретического материала с учетом знаний ответов на **Контрольные вопросы**, приведенные в пункте «**Контрольные вопросы**».

Задаются вопросы по пунктам, приведенным в пункте «Методические указания и порядок выполнения работы».

Критерии к оценке защиты отчета студентом приведены в таблице 1 «Шкала оценок уровня».

Контрольные вопросы

1. Дайте определения понятий «безопасность информации», «уязвимость информации», «защищённость информации», «угроза безопасности информации», «защита информации». Перечислите источники угроз безопасности информации и охарактеризуйте их классификацию.

2. Охарактеризуйте основные принципы сбора информации об инфраструктуре исследуемого объекта при проведении аудита безопасности. Укажите роль сетевого анализа в общем процессе аудита.

3. Поясните суть обследования инфраструктуры информационной системы некоторой организации. Какие методы получения сведений о сетевой топологии применяются на подготовительном этапе?

4. Опишите структуру кадра Ethernet II и поля его заголовка. Какое поле указывает на тип инкапсулированного протокола сетевого уровня?

5. Опишите структуру заголовка IP. Какие поля обеспечивают фрагментацию и повторную сборку дейтаграмм? Каково назначение поля TTL?

6. Опишите структуру заголовка ICMP. Чем отличаются сообщения типа 0, 8 и 11? В каких случаях формируется каждое из них?

7. Опишите структуру заголовка TCP. Какие флаги устанавливаются при трёхстороннем рукопожатии? Каково назначение поля SequenceNumber?

8. Поясните принцип работы протокола ARP. Чем отличаются запрос и ответ ARP по составу полей? Как формируется таблица ARP-кэша?

9. В чём различие между фильтром захвата (capturefilter) и фильтром отображения (displayfilter) в Wireshark? В каких случаях целесообразно применять каждый из них?

10. Каковы ограничения анализа зашифрованного трафика TLS в Wireshark? Какие методы применяются для расшифровки TLS-сессий при наличии ключей?

11. Что такое метаданные сетевого трафика? Какие сведения о сессии можно получить без расшифровки полезной нагрузки?

12. Поясните назначение и принцип работы утилит ping, traceroute, pathping. Какие сообщения ICMP используются каждой из них?

ЛАБОРАТОРНАЯ РАБОТА № 2. NMAP-СКАНИРОВАНИЕ

Общие сведения

Цель: получение навыков обследования организации Nmap-сканированием.

Программное обеспечение: среда виртуализации, дистрибутивы BacktrackLinux или KaliLinux, ПО Wireshark, nmap.

Методический материал по выполнению лабораторных заданий

Nmap (NetworkMapper) представляет собой современное средство разведки сетей и аудита безопасности с открытым исходным кодом. Программа применяется для обнаружения хостов в сети, определения состояния портов,

идентификации версий служб и операционных систем, а также для выполнения специализированных скриптов проверки уязвимостей через встроенную систему NmapScriptingEngine (NSE). Nmap доступен для всех основных операционных систем, включая Linux, Windows, macOS, и поддерживает работу как в интерактивном режиме, так и в составе автоматизированных конвейеров проверки защищённости.

Основные возможности Nmap включают: обнаружение активных хостов методами ARP, ICMP, TCP ACK/SYN и UDP; сканирование портов более чем десятью различными методами; определение версий служб путём анализа баннеров и сигнатур ответов; определение операционной системы методом TCP/IP fingerprinting; выполнение NSE-скриптов для автоматизированной проверки уязвимостей, извлечения сведений о сертификатах TLS, перечисления пользователей SMB, проверки настроек SSL/TLS и многих других задач. Современные версии Nmap (7.x) поддерживают сканирование IPv6-сетей и работу с большими диапазонами адресов в распределённом режиме через утилиту dnmap.

Состояние порта, определяемое Nmap, классифицируется следующим образом: «open» – порт открыт, служба принимает подключения; «closed» – порт доступен, но ни одна служба не прослушивает его; «filtered» – межсетевой экран или иное средство защиты блокирует определение состояния порта; «unfiltered» – порт доступен, но Nmap не может определить, открыт он или закрыт; «open|filtered» и «closed|filtered» – комбинированные состояния, когда невозможно однозначно классифицировать порт.

Основные методы сканирования Nmap включают: TCP SYN (-sS) – скрытное «полуоткрытое» сканирование, не устанавливающее полное TCP-соединение; TCP Connect (-sT) – полное TCP-соединение, надёжно регистрируемое целевой системой; UDP (-sU) – сканирование UDP-портов, требующее значительного времени; TCP FIN/Xmas/NULL (-sF/-sX/-sN) – скрытные методы, основанные на различиях реакции стеков TCP/IP; ACK (-sA) – определение наличия фильтрации; Window (-sW) – определение открытых портов по размеру окна TCP; RPC (-sR) – сканирование служб Sun RPC; Idle (-sI) – «слепое» сканирование через промежуточный «зомби»-хост.

Для определения версии службы применяется опция -sV, которая отправляет специфичные зонды и анализирует ответы, сравнивая их с базой сигнатур. Опция -O включает определение операционной системы путём анализа ответов TCP/IP стека. NmapScriptingEngine (NSE), активируемый опцией -sC (дефолтные скрипты) или --script, позволяет выполнять более 600 готовых скриптов категорий default, safe, vuln, brute, auth, discovery, intrusive и других, обеспечивая комплексную проверку целевых систем.

С позиции информационной безопасности Nmap является основным инструментом фазы разведки при проведении тестирования на проникновение, а также применяется при аудите сетевой инфраструктуры для выявления незакрытых портов, устаревших версий служб и некорректных настроек межсетевых экранов. Обнаружение факта сканирования Nmap осуществляется средствами IDS/IPS (Snort, Suricata) по характерным сигнатурам – нестандартным комбинациям TCP-флагов, последовательностям пакетов с малыми интервалами на различные порты.

Nmap предназначен для сканирования сетей с любым количеством объектов, определения состояния объектов сканируемой сети, а также портов и соответствующих данным служб. Для данного nmap использует много различных методов сканирования, таких, как:

– UDP connect();
– TCP connect();
– TCP SYN (полуоткрытое);
– FTP proxy (прорыв через ftp);
– Reverse-ident;
– ICMP (ping);
– FIN-сканирование;
– ACK-сканирование;
– Xmastree-сканирование;
– SYN-сканирование;
– NULL-сканирование.

Nmap также поддерживает большой набор дополнительных возможностей, а именно:

– определение операционной системы удаленного хоста с использованием TCP/IP fingerprint,
– «невидимое» сканирование,
– динамическое вычисление времени задержки и повтор передачи пакетов,
– параллельное сканирование,
– определение неактивных хостов методом параллельного ping-опроса,
– сканирование с использованием ложных хостов,
– определение наличия пакетных фильтров,
– прямое (без использования portmapper) RPC-сканирование,
– сканирование с использованием IP-фрагментации,
– произвольное указание IP-адресов и номеров портов сканируемых сетей.

Результатом работы Nmap является список отсканированных портов удаленной машины с указанием номера и состояния порта, типа используемого

протокола, а также названия службы, закрепленной за данным портом. Порт характеризуется тремя возможными состояниями «открыт», «фильтруемый» и «нефильтруемый»:

– открыт (open) – удаленная машина прослушивает данный порт;

– фильтруемый (filtered) – межсетевой экран, пакетный фильтр или другое устройство блокирует доступ к этому порту и nmap не смог определить данного состояние;

– «Нефильтруемый» (closed) – по результатам сканирования nmap воспринял данный порт как закрытый, при данном средстве защиты не помешали nmap определить данного состояние. Имеется состояние nmap определяет в любом случае (даже если большинство сканируемых портов хоста фильтруются).

В зависимости от указанных опций, nmap также может определить следующие характеристики сканируемого хоста:

- операционная система хоста,
- метод генерации TCP ISN,
- имя пользователя владельца процесса, зарезервировавшего сканируемый порт,
- символьные имена, соответствующие сканируемым IP-адресам и т. д.

Опции выбора метода сканирования:

-sT TCP Connect()

-sS TCP SYN

-sF FIN-сканирование

-sXXmasTree сканирование

-sN NULL-сканирование

-sPPing-сканирование

-sV Определение версий

-sU UDP-сканирование

-sO Сканирование протоколов IP

-sI<zombie_хост[:порт]> - Сканирование «вхолостую»

-sA ACK-сканирование

-sW TCP Window

-sR RPC-сканирование

-b<ftprelayhost> – «ПрорывчерезFTP». В качестве аргумента передаетсяURLftp-сервера, используемого в качестве «доверенного» (имя_пользователя: пароль@сервер:порт)

Некоторые опции настройки и выбора дополнительных возможностей:

-P0 – не производить ping-опрос хостов перед указанных непосредственным сканированием. Данная опция позволяет просканировать

сети, блокирующие обработку ICMP-эха с помощью межсетевых экранов. Примером подобный сети является microsoft.com.

-O – данная опция позволяет определить операционную систему сканируемого хоста с помощью метода TCP/IP fingerprint. Другими словами, nmap активизирует мощный алгоритм, функционирующий на основе анализа свойств сетевого программного обеспечения установленной на указанном хосте ОС. В результате сканирования получается формализованный «отпечаток», состоящий из стандартных тестовых запросов и «ответов» хоста на указанных хостах. Затем полученный отпечаток сравнивается с имеющейся базой стандартных ответов известных ОС, и на основании данного принимается решение о типе и версии ОС сканируемого хоста. Данный метод требует наличия хотя бы одного закрытого и одного открытого порта на целевом хосте.

-p <диапазон(ы)_портов>– данная опция указывает nmap, какие порты необходимо просканировать. Например, '-p 23' означает сканирование 23 порта на целевой машине. Если указано выражение типа '-p 20-30,139,60000-' Nmap будет сканировать порты с номерами с 20 по 30 включительно, 139 и от 60000 и выше (до 65535). По умолчанию nmap сканирует все порты в диапазоне 1-1024

Способы задания целевого хоста. Все, что не является опцией или данной аргументом, nmap воспринимает как адрес или имя целевого хоста (т.е. Хоста, подвергаемого сканированию). Простейший способ задать сканируемый хост - указать данного имя или адрес в командной строке после указания опций и аргументов. Если требуется просканировать подсеть IP-адресов, при выполнении необходимо указать параметр '/mask' («маска») после имени или IP-адреса сканируемого хоста.

Маска может принимать следующие значения:

'/0' – сканировать весь Интернет;

'/16' – сканировать адреса класса B;

'/24' – сканировать адреса класса C;

'/32' – сканировать только указанный хост.

Nmap позволяет также гибко указать целевые IP-адреса, используя списки и диапазоны для каждого указанных элемента. Например, необходимо просканировать подсеть класса B с адресом 128.210.*.*. Задать данную сеть можно любым из следующих способов: 128.210.*.*

128.210.0-255.0-255

128.210.1-50,51-255.1,2,3,4,5-255

128.210.0.0/16

Все данные строки эквивалентны. Заметим, что если используется «звездочку» ('*'), то в большинстве оболочек при выполнении понадобится отделить данной обратной "косой чертой" либо апострофами. Приведем еще

один пример. Если указано в качестве целевого IP-адреса строку '*.*.5.6-7', Nmap отсканирует все IP-адреса, оканчивающиеся на 5.6 либо

Методические указания и порядок выполнения работы. Задание

Цель лабораторной работы предполагает освоение методик сканирования сетевых ресурсов с применением Nmap, изучение особенностей различных методов сканирования, приобретение навыков определения состояния портов, версий служб и операционных систем, а также выявления признаков сканирования средствами сетевого анализа.

В рамках лабораторной работы выполняются следующие задачи:

1. Освоение основных опций и методов сканирования Nmap на локальном хосте.
2. Сравнительный анализ результатов сканирования TCP и UDP-портов различными методами.
3. Определение версий служб и операционных систем целевых хостов.
4. Применение NSE-скриптов для получения дополнительных сведений о целевых хостах.
5. Выявление признаков сканирования в сетевом трафике с применением анализатора Wireshark.
6. Анализ эффективности работы средств защиты (межсетевого экрана, IDS) по обнаружению различных методов сканирования.
7. Произвести сканирование с использованием Nmap
8. Определить специфику работы Nmap

Задание 1. Установка Nmap и сканирование локального хоста

1. Проверьте наличие установленного Nmap: nmap --version. В Linux установите пакет: sudoaptinstallnmap. В Windows скачайте установщик с официального сайта nmap.org и выполните установку. Для работы SYN-сканирования в Windows установите драйвер Npcap, входящий в состав установщика Nmap.

2. Изучите справочную информацию: nmap -h. Ознакомьтесь с основными опциями: -sS (SYN-сканирование), -sT (TCP Connect), -sU (UDP-сканирование), -sV (определение версий), -O (определение ОС), -p (указание портов), -T (тайминг-шаблон), -oN/-oX/-oA (сохранение результатов).

3. Выполните базовое сканирование TCP-портов локального хоста без прав root: nmap 127.0.0.1. По умолчанию Nmap сканирует 1000 наиболее распространённых TCP-портов методом TCP Connect. Зафиксируйте список открытых портов и определённых служб.

4. Выполните SYN-сканирование (требует прав root): sudonmap -sS 127.0.0.1. SYN-сканирование является более скрытным, так как не устанавливает полное TCP-соединение и не регистрируется в журналах

приложений. Сравните результаты с предыдущим сканированием – они должны быть идентичными, но SYN-сканирование выполняется быстрее.

5. Выполните сканирование с определением версий служб: nmap -sV 127.0.0.1. Nmap отправит специфичные зонды на каждый открытый порт и сопоставит ответы с базой сигнатур. Зафиксируйте выявленные названия и версии служб. Сопоставьте результаты с фактическими версиями, выполнив команды: ssh -V, apache2 -v (или аналогичные для выявленных служб).

6. Определите операционную систему локального хоста: sudonmap -O 127.0.0.1. Nmap анализирует ответы TCP/IP стека и сопоставляет их с базой отпечатков ОС. Сопоставьте результат с фактической операционной системой.

7. Выполните комплексное сканирование с применением NSE-скриптов: nmap -sC 127.0.0.1. Опция -sC запускает скрипты категории default, обеспечивающие получение дополнительных сведений о выявленных службах. Сохраните результаты в трёх форматах: nmap -sS -sV -O -sC -oAscan_localhost 127.0.0.1.

Задание 2. Сканирование сетевого сегмента

1. Определите IP-адрес и маску подсети лабораторного стенда: ipconfig (Windows) или ipaddr (Linux). Зафиксируйте адрес подсети в формате CIDR (например, 192.168.1.0/24).

2. Выполните обнаружение активных хостов в сети без сканирования портов: sudonmap -sn 192.168.1.0/24. Nmap использует комбинацию ARP-запросов (в локальной сети), ICMP EchoRequest и TCP ACK/SYN для определения активных хостов. Зафиксируйте список обнаруженных хостов с MAC-адресами и определёнными производителями сетевых карт.

3. Выполните быстрое сканирование 100 наиболее распространённых портов всех активных хостов: nmap -F 192.168.1.0/24. Зафиксируйте хосты с открытыми портами и выявленными службами.

4. Выберите один из обнаруженных хостов и выполните детальное сканирование всех 65535 TCP-портов с определением версий: sudonmap -sS -sV -p- <IP-адрес_хоста>. Зафиксируйте все открытые порты и версии служб.

5. Выполните сканирование UDP-портов (требует прав root и значительного времени): sudonmap -sU --top-ports 50 <IP-адрес_хоста>. Ограничение количества портов (--top-ports 50) ускоряет сканирование. Зафиксируйте открытые и фильтруемые UDP-порты.

6. Примените NSE-скрипты для получения дополнительных сведений. Выполните проверку уязвимостей: nmap --scriptvuln<IP-адрес_хоста>. Зафиксируйте выявленные уязвимости с указанием CVE-идентификаторов и CVSS-оценок.

7. Выполните проверку настроек SSL/TLS на выявленных HTTPS-службах: nmap --scriptssl-enum-ciphers,ssl-cert -p 443 <IP-адрес_хоста>.

Зафиксируйте поддерживаемые наборы шифров и выявленные слабые алгоритмы.

Задание 3. Сравнительный анализ методов сканирования

1. Запустите Wireshark на целевом хосте для захвата трафика. Установите фильтр: ip.src == <IP-адрес_сканирующего_хоста>.

2. С сканирующего хоста последовательно выполните различные методы сканирования одного и того же целевого хоста, сохраняя результаты каждого сканирования в отдельный файл: sudonmap -sS -oN scan_syn.txt <IP-адрес>; sudonmap -sT -oN scan_connect.txt <IP-адрес>; sudonmap -sF -oN scan_fin.txt <IP-адрес>; sudonmap -sX -oN scan_xmas.txt <IP-адрес>; sudonmap -sN -oN scan_null.txt <IP-адрес>; sudonmap -sA -oN scan_ack.txt <IP-адрес>.

3. После каждого сканирования сохраните захваченный в Wireshark трафик в отдельный файл pcap. Проанализируйте характерные признаки каждого метода: SYN-сканирование – пакеты с флагом SYN, при открытом порте ответ SYN-ACK, при закрытом – RST; TCP Connect– полное трёхстороннее рукопожатие с последующим RST; FIN-сканирование – пакеты с флагом FIN, при закрытом порте ответ RST, при открытом – отсутствие ответа; Xmas-сканирование – пакеты с флагами FIN+PSH+URG; Null-сканирование – пакеты без флагов; ACK-сканирование – пакеты с флагом ACK для определения наличия фильтрации.

4. Сопоставьте результаты каждого метода сканирования с захваченным трафиком. Сформируйте таблицу соответствия методов сканирования и наблюдаемых ответов целевого узла для открытых, закрытых и фильтруемых портов.

5. Сформулируйте сигнатуры каждого метода сканирования, которые могут быть использованы для настройки правил IDS/IPS. Сопоставьте полученные сигнатуры с правилами Snort для выявления сканирования (правила с sid из диапазона 1–100 в категории scan).

Задание 4. Обнаружение и предотвращение сканирования

1. На целевом хосте установите Snort или Suricata в режиме IDS. Настройтеправиладлявыявления SYN-сканирования: alert tcp any any -> $HOME_NET any (flags:S,12; msg:"SYN scan"; detection_filter:trackby_src, count 10, seconds 5; sid:1000001;).

2. Выполните SYN-сканирование с сканирующего хоста. Проверьте срабатывание правила в журнале Snort: sudotail -f /var/log/snort/alert.

3. Настройте блокировку сканирования средствами iptables на целевом хосте: sudoiptables -A INPUT -p tcp --tcp-flags ALL SYN -m recent --set --name SYN_SCAN; sudoiptables -A INPUT -p tcp --tcp-flags ALL SYN -m recent --update --seconds 5 --hitcount 10 --name SYN_SCAN -j DROP. Эти правила блокируют IP-адреса, отправившие более 10 SYN-пакетов за 5 секунд.

4. Повторите SYN-сканирование и убедитесь, что блокировка срабатывает. Зафиксируйте изменения в поведении Nmap (тайм-ауты, сообщения о фильтрации портов).

5. Сформируйте итоговый отчёт по результатам выполнения всех заданий. Отчёт должен содержать: результаты сканирования в различных форматах, анализ захваченного трафика, описание сигнатур методов сканирования, настройку правил IDS/IPS, выводы по эффективности обнаружения и блокирования сканирования.

Индивидуальное задание

Студент самостоятельно выбирает инфраструктуру организации для рассмотрения при решении лабораторных задач.

Требования к отчету и защите

Защита предполагает опрос по материалу, изложенному в отчете.

Отчет должен быть оформлен: требуется наличие титульного листа, введение (содержит: указание цели лабораторной работы, задачи, краткие теоретические сведения), последовательное изложение разработанного материала, подписанные скриншоты хода работы, если использовалась компьютерная техника, заключение (выводы, анализ полученных результатов).

Проверяется знание теоретического материала с учетом знаний ответов на **Контрольные вопросы**, приведенные в пункте «**Контрольные вопросы**».

Задаются вопросы по пунктам, приведенным в пункте «Методические указания и порядок выполнения работы».

Критерии к оценке защиты отчета студентом приведены в таблице 1 «Шкала оценок уровня».

Контрольные вопросы

1. Что такое статический и динамический IP-адрес? В чём принципиальная разница между ними? Какие методы выделения адресов применяются в современных сетях?

2. В чём заключается метод сканирования протоколов IP (опция -sO)? Какие сведения позволяют получить результаты такого сканирования?

3. На какие пакеты большинство операционных систем должны ответить флагом RST согласно RFC 793? Чем это объясняется?

4. Назначение, цели и описание технологии Honeypot. Какие типы Honeypot-систем применяются в современных инфраструктурах безопасности?

5. Какие цели может преследовать злоумышленник, проводящий разведку сетевой инфраструктуры организации? Чем эти цели обусловлены?

6. Какое наказание предусмотрено в Российской Федерации за несанкционированный доступ к компьютерной информации (статьи 272–274 УК РФ)?

7. Как выявлять Honeypot-системы при проведении разведки? Какие признаки могут выдать наличие Honeypot в сети?

8. Что такое DHCP? Какова роль DHCP в современной сетевой инфраструктуре? Какие угрозы безопасности связаны с протоколом DHCP?

9. Для чего используется RPC-сканирование? Какие службы обычно обнаруживаются этим методом?

10. Перечислите основные методы сканирования Nmap и охарактеризуйте каждый из них с точки зрения скрытности и надёжности.

11. Чтотакое NSE (Nmap Scripting Engine)? Какие категории скриптов существуют и для каких задач они применяются?

12. Каким образом средства IDS/IPS обнаруживают факты сканирования Nmap? Какие сигнатуры используются для выявления различных методов сканирования?

ЛАБОРАТОРНАЯ РАБОТА № 3. «ИЗУЧЕНИЕ РАБОТЫ ИНСТРУМЕНТА HPING3, ОРГАНИЗАЦИЯ СТРЕСС-ТЕСТА СЕТИ С ЕГО ПОМОЩЬЮ. НАСТРОЙКА БРАНДМАУЭРА С ПОМОЩЬЮ UFW И IPTABLES»

Общие сведения

Цель работы: освоение порядка работы инструмента hping3, организация стресс-теста сети, настройка брандмауэра с помощью ufw и iptables

Программное обеспечение: ОС MicrosoftWindows XP/2003/2008R2/Vista/7/8/8.1/2012/2012R2, ОС Linux, hping3, nmap, ufw и iptables.

Методический материал по выполнению лабораторных заданий

Hping3 представляет собой инструмент командной строки для формирования и анализа TCP/IP-пакетов с возможностью настройки всех полей заголовков. Программа применяется для тестирования межсетевых экранов, сканирования портов, эмуляции сетевых атак, проверки работы сетевых стеков и диагностики маршрутизации. В отличие от классического ping, работающего только с ICMP, hping3 поддерживает формирование пакетов TCP, UDP, ICMP и Raw IP с произвольной комбинацией флагов и произвольным содержимым полей.

Программа поддерживает режимы работы: -0 (Raw IP), -1 (ICMP), -2 (UDP), -8 (сканирование портов), -9 (прослушивание). Для TCP-пакетов доступны опции установки флагов -S (SYN), -A (ACK), -R (RST), -F (FIN), -P (PUSH), -U (URG), -X (XMAS), -Y (YMAS). Дополнительные опции позволяют задавать размер пакета (-d), количество пакетов (-c), порт источника (-s), порт назначения (-p), окно TCP (-w), время жизни TTL (--ttl), режим флуда (--flood) с максимальной скоростью отправки без ожидания ответов, и подмену адреса источника (--spoof или -a).

Брандмауэр (межсетевой экран) представляет собой сетевое устройство или программное средство, контролирующее входящий и исходящий трафик на основе набора правил безопасности. Современные брандмауэры реализуют фильтрацию пакетов, анализ состояния соединений (statefulinspection), а также возможности глубокого анализа пакетов (DPI) и выявления вторжений (IDS/IPS). В операционных системах Linux функции брандмауэра реализуются в ядре через подсистемы Netfilter (iptables, nftables) и обеспечивают фильтрацию на уровнях сетевого и транспортного интерфейсов.

UFW (UncomplicatedFirewall) является высокоуровневой надстройкой над iptables, предназначенной для упрощения настройки брандмауэра в дистрибутивах Ubuntu и других Debian-подобных системах. UFW предоставляет простой синтаксис для разрешения или запрета соединений по именам сервисов или номерам портов, поддерживает профили приложений, ведение журнала и настройку правил для IPv4 и IPv6. UFW рекомендуется для быстрой настройки базовой защиты серверов и рабочих станций без необходимости глубокого понимания синтаксиса iptables.

Iptables представляет собой низкоуровневую утилиту настройки таблиц правил Netfilter в ядре Linux. Iptables использует три основные таблицы: filter (фильтрация пакетов), nat (преобразование сетевых адресов), mangle (изменение полей заголовков). Каждая таблица содержит цепочки правил: INPUT (входящие пакеты), FORWARD (транзитные пакеты), OUTPUT (исходящие пакеты), PREROUTING (до маршрутизации), POSTROUTING (после маршрутизации). Каждое правило содержит критерии отбора (адреса, порты, протоколы, состояния) и целевое действие: ACCEPT, DROP, REJECT, RETURN, LOG, MASQUERADE, DNAT, SNAT. Современной альтернативой iptables является nftables, обеспечивающая более эффективную обработку правил и единый синтаксис для фильтрации IPv4, IPv6, ARP и мостов.

С позиции информационной безопасности hping3 применяется при тестировании на проникновение для проверки корректности настройки средств защиты, эмуляции сетевых атак (в частности, SYN-флуда, FIN-сканирования, подмены адреса источника) и стресс-тестирования сетевой инфраструктуры. UFW и iptables обеспечивают базовую защиту серверов и рабочих станций, а

также применяются для сегментации сетей и реализации политик доступа в соответствии с принципом минимальных привилегий.

1. Hping3 – инструмент для создания и анализа пакетов ICMP или TCP. В дополнение к обнаружению пробелов в безопасности также допускается использовать команду для проверки сетей и хостов. Анализатор пакетов TCP/IP адаптирован к командной строке на основе команды ping, и даже имеет режим traceroute. Указанную утилиту также можно использовать для отправки файлов между фактически скрытыми соединениями. Другие возможные применения включают в себя тесты межсетевого экрана, расширенное сканирование портов, сетевые тесты с различными протоколами, фрагментацию или проверку стеков TCP/IP. С помощью hping начинающие и профессионалы получают полное представление о специфике TCP/IP. Данная утилита функционирует во всех Unix-подобных системах, таких как Linux, FreeBSD, NetBSD, OpenBSD и Solaris [2].

Программа hping3 по умолчанию (без параметров) отправляет нулевой пакет с заголовком TCP на порт 0.

допускается выбрать использование другого протокола с помощью числовой опции, доступной для каждого из указанных режимов [2]:

a) -0 (режим Raw IP);
b) -1 (режим ICMP);
c) -2 (режим UDP);
d) -8 (Режим сканирования);
e) -9 (Режим прослушивания).

Поскольку hping3 использует TCP по умолчанию, отсутствие указанных ниже параметров отправит пакет с сегментом TCP. При использовании протокола TCP можно решить: либо опустить флаги (по умолчанию), либо установить флаг, используя один из следующих параметров [2]:

a) -S (SYN);
b) -A (ACK);
c) -R (RST);
d) -F (FIN);
e) -P (PUSH);
f) -U (URG);
g) -X (XMAS);
h) -Y (YMAS).

2. Брандмауэр. Брандмауэр – сетевое устройство, отслеживающее входящий и исходящий сетевой трафик и разрешает или блокирует пакеты данных на основе набора правил безопасности. Цель – установить барьер между используемой внутренней сетью и входящим трафиком из внешних источников (таких как Интернет), чтобы блокировать вредоносный трафик, такой как вирусы и хакеры [3].

Брандмауэры тщательно анализируют входящий трафик на основе заранее установленных правил и фильтруют трафик, поступающий из незащищенных или подозрительных источников, для предотвращения атак. Брандмауэры защищают трафик в точке входа компьютера, называемой портами, где происходит обмен информацией с внешними устройствами.

В Linux-системах для настройки брандмауэра чаще всего используются две утилиты: Ufw и Iptables.

3. Утилита Ufw. UFW (UncomplicatedFirewall) является самым простым и довольно популярным инструментарием командной строки для настройки и управления брандмауэром в дистрибутивах Ubuntu и Linux. Правильно функционирующий брандмауэр является наиболее важной частью полной безопасности системы Linux.

4. Утилита Iptables. Iptables – брандмауэр для Linux, используемый для мониторинга входящего и исходящего трафика и фильтрации трафика в соответствии с заданными пользователем правилами для исключения несанкционированного доступа к системе. При помощи Iptables можно разрешить на используемом сервере движение только определенного трафика. В данном руководстве рассматривается использование Iptables для обеспечения безопасности веб-приложения.

Все данные передаются по сети в виде пакетов. Ядро Linux предоставляет интерфейс для фильтрации пакетов входящего и исходящего трафика при помощи специальных таблиц. Iptables – приложение командной строки и межсетевой экран для Linux, указанным приложением можно пользоваться для создания, поддержания работоспособности и проверки данных таблиц.

Можно создать несколько таблиц. В каждой таблице может содержаться несколько цепочек. Цепочка – набор правил. Каждое правило определяет, что делать с пакетом, если пакет соответствует условиям. При соответствии пакета над указанным пакетом выполняется целевое действие (TARGET). Может быть проверка следующей цепочкой или один из следующих вариантов:

- ACCEPT: разрешить передачу пакета.
- DROP: запретить передачу пакета.
- RETURN: пропустить текущую цепочку и перейти к следующему правилу в цепочке, вызвавшей текущую. В таблице фильтра есть три цепочки (набора правил):

–INPUT – используется для контроля входящих пакетов. Можно разрешать или блокировать подключения по порту, протоколу или IP-адресу источника.

–FORWARD – используется для фильтрации пакетов, приходящих на сервер, но перенаправляемых куда-либо еще.

–OUTPUT – используется для фильтрации исходящих пакетов.

5. Сетевые инструменты.

Помимо hping существуют и другие инструменты, связанные с формированием пакетов, эхо-запросов и сканированием портов:

ping – проверяет end-to-end соединение (задержку времени приёма передачи (RTT), флуктуации, потерю пакетов) удалённого хоста запросами ICMP echo/reply. Полезна для проверки статуса и доступности.

traceroute - определяет путь третьего уровня переадресации из локального хоста к удалённому конечному хосту посредством пакетов зондирования ICMP/UDP/TCP с ограниченным временем жизни (TTL). Полезна для решения таких сетевых проблем как доступность и роутинг.

mtr – вариант traceroute указанный комбинирует функциональность traceroute и ping и отображает статистику. Полезен для характеристики задержек роутинга по всему пути.

netcat/socat позволяет читать/записывать поток байтов на TCP/UDP. Полезен для решения проблем с политиками файервола и доступностью служб.

dig – инструмент по решению DNS проблем, указанный может генерировать прямые запросы, обратные запросы, находить авторитетные сервера имён, проверять CNAME, MX и другие DNS записи. Можно указать для запросов конкретный DNS сервер по используемому выбору.

nslookup – инструмент проверки/решения проблем с DNS. Работает со всеми DNS запросами и записями. Может делать запросы к конкретному DNS серверу.

dnsyo – инструмент тестирования DNS, указанный проверяет распространение DNS выполняя поиск DNS из ряда открытых резолверов, расположенных по в 1,500 различных сетях по всему миру.

lsof – оказывает информацию о файлах (например, обычные файлы, сокеты), указанные открыты процессами. Полезно для монитора процессов или пользователей с точки зрения указанных открытых сетевых соединений или открытых файлов.

iftop – основанная на ncurses утилита, указанная может применяться для монитора в реальном времени использования полосы пропускания и сетевых соединений для отдельных сетевых интерфейсов. Полезна для отслеживания пропускной способности приложений, потребляющих много системных ресурсов, пользователей, мест назначений и портов системных служб.

netstat – утилита сбора статистики сети, указанная может показать информацию о статусе, о статистике открытых сетевых соединений (TCP/UDP портах, IP адресах), таблицах роутинга, переданного/полученного трафика и протоколах. Полезна для связанных с сетью диагностик и тонкой настройки производительности.

tcpdump – сниффер пакетов, основанный на библиотеке по захвату пакетов libpcap. Может определять фильтры по захвату пакетов в формате BerkeleyPacketFilters.

tshark – ещё одна программа-сниффер пакетов командной строки с полной совместимостью с снифферомWireshark. Поддерживает 1000 протоколов. Полезна для решения проблем, анализа и сохранения информации по живым пакетам.

ip – разносторонняя утилита командной строки, указанная является частью пакета iproute2. Полезна для проверки и модификации таблиц маршрутизации, состояний сетевых устройств и настроек IP-туннелирования. Полезна для просмотра таблиц маршрутизации, добавления/удаления статичных маршрутов, при настройке сетевых интерфейсов и других функций по решению проблем с маршрутизацией.

ifup/ifdown – используется для влючения или отключения конкретного сетевого интерфейса.

Autossh – программа, указанная создаёт SSH-сессии и автоматически перезапускает сессии, указанные должны быть разъединены. Полезна для создания постоянного реверсного SSH-туннеля между ограниченными корпоративными сетями.

Методические указания и порядок выполнения работы. Задание

Цель лабораторной работы предполагает освоение порядка работы с инструментом hping3 для формирования и анализа TCP/IP-пакетов, организация стресс-теста сетевого узла, а также приобретение навыков настройки брандмауэра средствами ufw и iptables.

В рамках лабораторной работы выполняются следующие задачи:

Освоение основных команд и опций hping3 для формирования ICMP, TCP и UDP-пакетов.

Выполнение различных типов запросов (SYN, ACK, FIN, UDP) с применением hping3 и анализ ответов целевого узла.

Сканирование портов и трассировка маршрутов с применением hping3.

Организация стресс-теста сетевого узла с использованием режима флуда и подмены адреса источника.

Установка и настройка UFW для базовой защиты операционной системы Linux.

Настройка правил iptables для фильтрации трафика по портам, протоколам и IP-адресам.

Проверка корректности работы правил брандмауэра путём формирования тестового трафика с применением hping3 и nmap.

1.Изучить принципы сниффинг с использованием фильтров hping

2. Произветси тестирование Ethernet-сети с помощью программы hping

Задание 1. Освоение hping3 и формирование различных типов пакетов

1. Убедитесь в наличии установленного hping3: hping3 --version. При отсутствии установите: sudoaptinstall hping3. Ознакомьтесь со справкой: hping3 --help. Изучите основные опции: -0 (Raw IP), -1 (ICMP), -2 (UDP), -8 (сканирование), -9 (прослушивание), -S/-A/-R/-F/-P/-U/-X/-Y (флаги TCP), -c (количество пакетов), -s (порт источника), -p (порт назначения), -d (размер данных), --flood (режим флуда), --rand-source (случайный адрес источника).

2. Выполните ICMP-пинг целевого хоста с отправкой 10 пакетов: sudo hping3 -1 -c 10 <IP-адрес_цели>. Проанализируйте вывод: для каждого пакета отображается время прохождения (RTT), размер и последовательный номер. Опция -1 переводит hping3 в режим ICMP, -c 10 ограничивает количество пакетов.

3. Сравните результат с обычным ping: ping -c 10 <IP-адрес_цели>. Зафиксируйте различия в формате вывода и возможностях настройки полей пакетов.

4. Выполните SYN-запрос на порт 0 с указанием порта источника 5151: sudo hping3 -S -c 3 -s 5151 -p 0 <IP-адрес_цели>. Опция -S устанавливает флаг SYN в заголовке TCP, -s 5151 задаёт порт источника, -p 0 указывает порт назначения. Поскольку порт 0 не открыт, целевой узел ответит пакетом RST-ACK. Зафиксируйте наблюдаемые флаги ответа.

5. Выполните FIN-запрос на порт 0: sudo hping3 -F -c 1 -s 5151 -p 0 <IP-адрес_цели>. Опция -F устанавливает флаг FIN. Сравните ответ с SYN-запросом – при FIN-сканировании закрытые порты отвечают RST, а открытые игнорируют пакет (согласно RFC 793).

6. Выполните ACK-запрос на порт 80: sudo hping3 -A -c 3 -s 5151 -p 80 <IP-адрес_цели>. Опция -A устанавливает флаг ACK. Этот метод применяется для определения наличия фильтрации: если целевой узел отвечает RST, порт нефильтруемый; при отсутствии ответа или получении ICMP-сообщения порт фильтруется.

7. Выполните UDP-запрос на порт 80: sudo hping3 -2 -c 1 -s 5151 -p 80 <IP-адрес_цели>. Опция -2 переводит hping3 в режим UDP. При закрытом UDP-порте целевой узел ответит ICMP-сообщением PortUnreachable (Type 3, Code 3). Зафиксируйте полученный ответ.

Задание 2. Сканирование портов и трассировка с hping3

1. Выполните сканирование диапазона портов 50–81 в режиме сканирования: sudo hping3 -8 50-81 -S <IP-адрес_цели> -V. Опция -8 активирует режим сканирования, -S устанавливает флаг SYN, -V включает подробный вывод. Для каждого порта hping3 отобразит состояние: открытые порты (flags=SA – SYN-ACK), закрытые порты (flags=RA – RST-ACK).

2. Выполните трассировку UDP-маршрута: sudo hping3 -2 <IP-адрес_цели> -p 4444 -T -n. Опция -T включает режим traceroute с увеличением TTL, -n отключает разрешение DNS-имён. Зафиксируйте промежуточные маршрутизаторы и время прохождения до каждого из них.

3. Выполните трассировку с фиксированным TTL для анализа маршрута: sudo hping3 -S <IP-адрес_цели> -p 80 -T --ttl 15 --tr-keep-ttl -n. Опция --ttl 15 устанавливает начальное значение TTL, --tr-keep-ttl фиксирует TTL для всех пакетов, что позволяет наблюдать за изменениями маршрута при неизменном TTL.

Задание 3. Стресс-тест сети

ВНИМАНИЕ: Stress-тест выполнять только в изолированной лабораторной сети. Запрещено применять против производственных систем.

1. На целевом хосте запустите утилиту top для мониторинга загрузки процессора, памяти и сетевого интерфейса: top. Также откройте второй терминал с утилитой iftop для мониторинга сетевого трафика: sudoiftop -i <интерфейс>.

2. С атакующего хоста выполните SYN-флуд на FTP-порт (21) целевого хоста: sudo hping3 -S -c 1001 -d 120 -w 64 -p 21 --flood --rand-source<IP-адрес_цели>. Опция -c 1001 задаёт количество пакетов, -d 120 – размер данных каждого пакета, -w 64 – размер окна TCP, --flood– режим максимальной скорости отправки без ожидания ответов, --rand-source– случайная подмена адреса источника для обхода фильтрации по IP.

3. Наблюдайте за изменением загрузки процессора на целевом хосте. Зафиксируйте значения us (время работы пользовательских процессов), sy (время работы процессов ядра), si (время обработки программных прерываний) в выводе top. Также зафиксируйте сетевой трафик в iftop.

4. Остановите атаку нажатием Ctrl+C на атакующем хосте. Зафиксируйте время восстановления нормального состояния целевого хоста.

5. Повторите стресс-тест с применением UDP-флуда: sudo hping3 -2 --flood -p 53 --rand-source<IP-адрес_цели>. Сравните воздействие SYN-флуда и UDP-флуда на целевой хост. Зафиксируйте различия в загрузке процессора, памяти и сети.

Задание 4. Настройка брандмауэра с помощью UFW

1. Убедитесь в наличии установленного UFW: sudoufwstatusverbose. При отсутствии установите: sudoaptinstallufw. Проверьте, что IPv6 включён: откройте файл /etc/default/ufw и убедитесь, что IPV6=yes. При необходимости измените и сохраните файл.

2. Задайте политики по умолчанию: sudoufwdefaultdenyincoming– запретить все входящие соединения; sudoufwdefaultallowoutgoing– разрешить все исходящие соединения. Эти политики обеспечивают базовую защиту по принципу «запрещено всё, что не разрешено явно».

3. Разрешите подключения по SSH (порт 22): sudoufwallowssh. UFW автоматически определяет номер порта по имени сервиса. При необходимости укажите порт явно: sudoufwallow 22/tcp. Разрешите подключения по HTTP и HTTPS: sudoufwallowhttp и sudoufwallowhttps.

4. Разрешите подключения с определённых IP-адресов: sudoufwallowfrom 192.168.1.0/24 toanyport 22. Эта команда разрешает SSH-подключения только из локальной подсети.

5. Включите брандмауэр: sudoufwenable. Подтвердите включение, нажав y. Проверьте статус: sudoufwstatusnumbered. Убедитесь, что все правила применены корректно.

6. Включите журналирование: sudoufwloggingon. Проверьте журнал: sudotail -f /var/log/ufw.log. Журнал фиксирует заблокированные и разрешённые соединения.

7. Проверьте корректность работы правил, выполнив тестовые подключения с атакующего хоста. SSH-подключение должно устанавливаться, а подключения к другим портам – блокироваться. Зафиксируйте результаты проверки.

Задание 5. Настройка брандмауэра с помощью iptables

1. Просмотритетекущиеправилаiptables: sudoiptables -L -n -v --line-numbers. Изучите цепочки INPUT, FORWARD, OUTPUT и применённые правила.

2. Очистите существующие правила: sudoiptables -F (очистка всех цепочек) и sudoiptables -X (удаление пользовательских цепочек). Установите политики по умолчанию: sudoiptables -P INPUT DROP, sudoiptables -P FORWARD DROP, sudoiptables -P OUTPUT ACCEPT.

3. Разрешитевходящиесоединениянапорт 22 (SSH): sudo iptables -A INPUT -p tcp --dport 22 -m conntrack --ctstateNEW,ESTABLISHED -j ACCEPT. Использование модуля conntrack обеспечивает отслеживание состояния соединения и разрешение пакетов для установленных сессий.

4. Разрешитевходящиесоединениянапорты 80 и 443 (HTTP/HTTPS): sudo iptables -A INPUT -p tcp --dport 80 -m conntrack --ctstateNEW,ESTABLISHED -j

ACCEPT; sudo iptables -A INPUT -p tcp --dport 443 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT.

5. Разрешитевходящиепакетыдляужеустановленныхсоединений: sudo iptables -A INPUT -m conntrack --ctstateESTABLISHED,RELATED -j ACCEPT. Это правило должно находиться в начале цепочки INPUT для эффективности.

6. Разрешите входящие ICMP-пакеты (для диагностики): sudoiptables -A INPUT -p icmp -j ACCEPT. Разрешите трафик на loopback-интерфейсе: sudoiptables -A INPUT -i lo -j ACCEPT.

7. Добавьте правило для защиты от SYN-флуда: sudoiptables -A INPUT -p tcp --syn -m limit --limit 10/s --limit-burst 20 -j ACCEPT; sudoiptables -A INPUT -p tcp --syn -j DROP. Эти правила ограничивают количество SYN-пакетов до 10 в секунду с burst-значением 20.

8. Добавьте правило для журналирования заблокированных пакетов: sudoiptables -A INPUT -m limit --limit 5/min -j LOG --log-prefix "iptables-drop: " --log-level 4. Журнал будет записываться в /var/log/kern.log или /var/log/syslog.

9. Сохраните правила для применения после перезагрузки: sudoaptinstalliptables-persistent; sudonetfilter-persistentsave. Проверьте корректность работы правил, выполнив тестовые подключения.

10. Сформируйте итоговый отчёт по результатам выполнения всех заданий. Отчёт должен содержать: описание выполненных команд hping3 и наблюдаемых ответов, результаты стресс-теста, конфигурацию правил UFW и iptables, результаты проверки корректности работы брандмауэра.

Индивидуальное задание

Студент самостоятельно выбирает инфраструктуру организации для рассмотрения при решении лабораторных задач.

Требования к отчету и защите

Защита предполагает опрос по материалу, изложенному в отчете.

Отчет должен быть оформлен: требуется наличие титульного листа, введение (содержит: указание цели лабораторной работы, задачи, краткие теоретические сведения), последовательное изложение разработанного материала, подписанные скриншоты хода работы, если использовалась компьютерная техника, заключение (выводы, анализ полученных результатов).

Проверяется знание теоретического материала с учетом знаний ответов на **Контрольные вопросы**, приведенные в пункте «**Контрольные вопросы**».

Задаются вопросы по пунктам, приведенным в пункте «Методические указания и порядок выполнения работы».

Критерии к оценке защиты отчета студентом приведены в таблице 1 «Шкала оценок уровня».

Контрольные вопросы

1. Назначение и основные возможности hping3. Чем hping3 отличается от классической утилиты ping?
2. Какие режимы работы поддерживает hping3 и для каких задач применяется каждый из них?
3. Опишите назначение флагов TCP и приведите примеры сценариев их использования при тестировании сетевой инфраструктуры.
4. Что такое стресс-тест сети? Какие виды стресс-тестов применяются при аудите защищённости и какие ограничения существуют для их проведения?
5. Чем отличается режим --flood от обычной отправки пакетов? Какие риски связаны с применением данного режима?
6. Что такое брандмауэр? Какие типы брандмауэров существуют и как они классифицируются по принципу работы?
7. Опишите архитектуру Netfilter в Linux. Какова роль таблиц filter, nat и mangle?
8. Чем отличается UFW от iptables? В каких случаях целесообразно применять каждый из инструментов?
9. Опишите цепочки INPUT, FORWARD и OUTPUT в таблице filter. Для каких пакетов применяется каждая из них?
10. Какие целевые действия (TARGET) поддерживаются в iptables? Чем отличаются ACCEPT, DROP и REJECT?
11. Что такое statefulinspection? Как iptables отслеживает состояние TCP-соединений?
12. Что такое nftables и каковы преимущества перехода с iptables на nftables в современных дистрибутивах Linux?

ЛАБОРАТОРНАЯ РАБОТА № 4. АУДИТ СЕТЕВОЙ ИНФРАСТРУКТУРЫ СРЕДСТВАМИ POWERSHELL

Общие сведения

Цель работы: изучение проблем безопасности удаленного доступа к ОС MicrosoftWindows XP/2003/2008R2/Vista/7/8/8.1/10

Программное обеспечение: ОС MicrosoftWindows XP/2003/2008R2/Vista/7/8/8.1/10, ОС Linux, hping3, nmap, ufw и iptables

Методический материал по выполнению лабораторных заданий

PowerShell представляет собой современную объектно-ориентированную командную оболочку и язык сценариев корпорации Microsoft, построенную на платформе .NET. PowerShell применяется для автоматизации

администрирования операционных систем Windows, управления службами каталогов ActiveDirectory, облачными сервисами Microsoft 365 и Azure, а также для аудита информационной безопасности и расследования инцидентов. Современные версии PowerShell (7.x) являются кроссплатформенными и доступны для Windows, Linux и macOS, обеспечивая единый подход к управлению разнородной инфраструктурой.

Архитектура PowerShell основана на концепции командлетов (cmdlets) – специализированных команд, реализующих отдельные операции и возвращающих объекты .NET, а не текстовые строки. Командлеты следуют единому соглашению по именованию «Глагол-Существительное» (например, Get-Process, Set-Service, Stop-Service), что обеспечивает предсказуемость и удобство использования. Передача объектов между командлетами через конвейер позволяет строить сложные сценарии обработки данных без необходимости парсинга текстового вывода.

Основные командлеты, применяемые при аудите сетевой инфраструктуры, включают: Get-NetIPConfiguration и Get-NetIPAddress – получение сведений о сетевых настройках; Get-NetTCPConnection и Get-NetUDPEndpoint – анализ активных сетевых соединений; Test-NetConnection – проверка доступности узлов и портов; Resolve-DnsName – разрешение DNS-имён; Get-EventLog и Get-WinEvent – получение записей журналов событий Windows; Get-Process и Get-Service – анализ запущенных процессов и служб; Get-LocalUser и Get-LocalGroup – управление локальными пользователями и группами; Get-Acl – анализ прав доступа к объектам файловой системы.

Для аудита сетевой инфраструктуры PowerShell позволяет решать следующие задачи: перечисление активных сетевых соединений и выявление несанкционированных подключений; анализ запущенных процессов и служб с выявлением подозрительных; сбор и фильтрация записей журналов безопасности Windows; проверка настроек брандмауэра WindowsDefender; анализ локальных политик безопасности и прав пользователей; получение сведений об установленном программном обеспечении и обновлениях безопасности; проверка конфигурации служб удалённого доступа (RDP, WinRM).

PowerShell обеспечивает возможность удалённого выполнения команд через протокол WinRM (WS-Management), что особенно важно при аудите распределённых инфраструктур. Для безопасного применения PowerShell при аудите следует учитывать возможность ведения журналирования сценариев (ScriptBlockLogging), транскриптов (Transcription) и анализа событий в рамках расширенной защиты WindowsDefender ATP. Злоумышленники также активно применяют PowerShell для проведения атак, что делает мониторинг PowerShell-активности одним из ключевых элементов защиты.

С позиции информационной безопасности PowerShell является мощным инструментом как для администраторов и аудиторов, так и для злоумышленников. Корпоративные политики безопасности должны предусматривать ограничение выполнения сценариев через ExecutionPolicy, применение ConstrainedLanguageMode для ограничения возможностей, журналирование всех выполняемых сценариев и интеграцию событий PowerShell с SIEM-системами для выявления аномалий.

1. Аудит информационной безопасности компьютерных систем

1.1 Понятие аудита информационной безопасности

Под аудитом ИБ понимается системный процесс получения объективных качественных и количественных оценок текущего состояния ИБ организации в соответствии с определенными критериями и показателями на всех основных уровнях обеспечения безопасности: нормативно-методологическом, организационно-управленческом, процедурном и программно-техническом [11].

В результате проведения аудита могут быть получены как качественные, так и количественные оценки. При качественном оценивании, например, может быть приведен перечень уязвимостей в программно-аппаратном обеспечении с указанных классификацией по трехуровневой шкале опасности: высокая, средняя и низкая. Количественные оценки чаще всего применяются при оценке риска для активов организации, создаваемого угрозами безопасности. В качестве количественных оценок могут выступать, например, цена риска, вероятность риска, размер риска и т. п.

Объективность аудита обеспечивается, в частности, тем, что оценка состояния ИБ производится специалистами на основе определенной методики, позволяющей объективно проанализировать все составляющие СОИБ[11]. Выделяют три типа аудита ИБ:

− активный аудит;

− экспертный аудит;

− аудит на соответствие стандартам ИБ.

Активный аудит представляет собой обследование состояния защищенности компонентов информационных систем, относящихся к программно-техническому уровню. Например, вариант активного аудита, называемый тестом на проникновение (Penetrationtest), предполагает обследование подсистемы защиты сетевых взаимодействий.

Активный аудит включает:

− анализ текущей архитектуры и настроек элементов ПИБ;

− интервьюирование ответственных и заинтересованных лиц;

− проведение инструментальных проверок, охватывающих определенные ПИБ [11].

Опросные листы используются в процессе интервьюирования лиц, отвечающих за администрирование АИС, для получения субъективных характеристик АИС [11].

Параллельно с интервьюированием проводятся инструментальные проверки (тесты), могущие включать следующие мероприятия:

− визуальный осмотр помещений, обследование системы контроля доступа в помещения;

− получение конфигураций и версий устройств и ПО;

− проверка соответствия реальных конфигураций предоставленным исходным данным;

− получение карты сети специализированным ПО;

− использование сканеров защищенности (как универсальных, так и специализированных);

− моделирование атак, использующих уязвимости системы;

− проверка наличия реакции на действия, выявляемые механизмами обнаружения и реагирования на атаки. Аудитор может исходить из следующих моделей, описывающих степень данного знания исследуемой АИС (модель знания):

− модель «черного ящика»

– аудитор не обладает никакими априорными знаниями об исследуемой АИС [11].

Экспертный аудит предназначен для оценивания текущего состояния ИБ на нормативно-методологическом, организационно-управленческом и процедурном уровнях. Экспертный аудит проводится преимущественно внешними аудиторами, данного выполняют силами специалистов по системному управлению. Сотрудники организации-аудитора совместно с представителями заказчика проводят следующие виды работ:

− сбор исходных данных об АИС, данной функциях и особенностях, используемых технологиях автоматизированной обработки и передачи информации (с учетом ближайших перспектив развития);

− сбор информации об имеющихся организационно-распорядительных документах по обеспечению ИБ и указанных анализ;

− определение защищаемых активов, ролей и процессов СОИБ. Важнейшим инструментом экспертной оценки является сбор данных об АИС путем интервьюирования технических специалистов и руководства заказчика.

Основные цели интервьюирования руководящего состава организации:

− определение политики и стратегии руководства в вопросах обеспечения ИБ;

− выявление целей, указанные ставятся перед СОИБ;

− выяснение требований, указанные предъявляются к СОИБ;

− получение оценок критичности тех или иных подсистем обработки информации, оценок финансовых потерь при возникновении тех или иных инцидентов.

Основные цели интервьюирования технических специалистов:

− сбор информации о функционировании АИС;

− получение схемы информационных потоков в АИС;

− получение информации о технической части СОИБ;

Аудит на соответствие стандартам ИБ

В ряде случаев проводится аудит на соответствие стандартам ИБ. Специально уполномоченные организации-аудиторы по результатам аудита принимают решение и выдают документальное подтверждение о соответствии СОИБ тому или иному эталонному стандарту (проводят сертификацию). Сертификация является показателем качества СОИБ и поднимает престиж и уровень доверия к организации.

Аудит на соответствие стандартам чаще всего подразумевает проведение активного и экспертного аудита. По результатам могут быть подготовлены отчеты, содержащие следующую информацию:

− степень соответствия проверяемой ИС выбранным стандартам;

− количество и категории полученных несоответствий и замечаний;

− рекомендации по построению или модификации СОИБ, позволяющие привести данной в соответствие с требованиями рассматриваемого стандарта

Методические указания и порядок выполнения работы. Задание

Цель лабораторной работы предполагает освоение методик аудита сетевой инфраструктуры средствами PowerShell, приобретение навыков применения командлетов для получения сведений о сетевых настройках, активных соединениях, запущенных процессах и записях журналов событий Windows.

В рамках лабораторной работы выполняются следующие задачи:

1. Освоение базовых командлетовPowerShell для получения сведений о сетевой конфигурации операционной системы.
2. Анализ активных TCP и UDP-соединений, выявление несанкционированных подключений.
3. Проверка доступности сетевых узлов и портов с применением Test-NetConnection.
4. Сбор и фильтрация записей журналов событий Windows для выявления подозрительной активности.
5. Анализ запущенных процессов и служб с выявлением аномалий.
6. Проверка настроек локальной политики безопасности и прав пользователей.

7. Применение конвейерной обработки и форматирования вывода для формирования структурированных отчётов.

Задание 1. Получение сведений о сетевой конфигурации

1. Запустите PowerShell от имени администратора. В Windows используйте меню Пуск → WindowsPowerShell → Запуск от имени администратора. В Linux запустите pwsh. Проверьте версию: $PSVersionTable.PSVersion. Для работы с сетевыми командлетами требуются права администратора.

2. Получите полную сетевую конфигурацию: Get-NetIPConfiguration -Detailed. Командлет отображает IP-адреса, шлюзы, DNS-серверы и сетевые интерфейсы. Зафиксируйте IP-адрес, маску подсети, шлюз по умолчанию и DNS-серверы.

3. Получите список всех IP-адресов с фильтрацией по семейству адресов: Get-NetIPAddress | Where-Object {$_.AddressFamily -eq 'IPv4'} | Format-TableIPAddress, InterfaceAlias, PrefixLength, PrefixOrigin. Зафиксируйте все IPv4-адреса и соответствующие интерфейсы.

4. Получите сведения о DNS-клиенте и кэше: Get-DnsClient | Format-TableInterfaceAlias, ConnectionSpecificSuffix; Get-DnsClientCache | Sort-ObjectTimeToLive | Format-TableEntry, RecordName, RecordType, Data, TimeToLive -AutoSize. Зафиксируйте записи DNS-кэша и их TTL.

5. Выполните разрешение DNS-имени: Resolve-DnsName example.com -Type A. Определите IP-адреса, соответствующие доменному имени. Выполните обратное разрешение: Resolve-DnsName 93.184.216.34 -Type PTR. Сопоставьте результаты прямого и обратного разрешения.

6. Проверьте сетевое подключение: Test-NetConnection -ComputerName 8.8.8.8 -Port 53 -InformationLevel Detailed. Командлет выполняет проверку доступности узла и порта, выполняет traceroute и определяет MTU. Зафиксируйте результаты.

Задание 2. Анализ активных сетевых соединений

1. Получите список всех активных TCP-соединений: Get-NetTCPConnection | Sort-ObjectState | Format-TableLocalAddress, LocalPort, RemoteAddress, RemotePort, State, OwningProcess -AutoSize. Сгруппируйте соединения по состоянию: Listen, Established, TimeWait, CloseWait.

2. Получите список прослушиваемых TCP-портов: Get-NetTCPConnection -StateListen | Format-TableLocalAddress, LocalPort, OwningProcess -AutoSize. Для каждого порта определите процесс, прослушивающий его: Get-NetTCPConnection -State Listen | ForEach-Object { [PSCustomObject]@{ Port=$_.LocalPort; Process=(Get-Process -Id $_.OwningProcess -ErrorActionSilentlyContinue).ProcessName; PID=$_.OwningProcess } } | Format-Table -AutoSize.

3. Получите список прослушиваемых UDP-портов: Get-NetUDPEndpoint | Format-Table LocalAddress, LocalPort, OwningProcess -AutoSize. Определите процессы, использующие UDP-порты, аналогично TCP.

4. Выявите соединения с внешними адресами и определите процессы, которым они принадлежат: Get-NetTCPConnection -State Established | Where-Object {$_.RemoteAddress -notmatch '^(127\.|10\.|172\.(1[6-9]|2[0-9]|3[01])\.|192\.168\.|::1)'} | ForEach-Object { $p = Get-Process -Id $_.OwningProcess -ErrorActionSilentlyContinue; [PSCustomObject]@{ LocalPort=$_.LocalPort; RemoteAddress=$_.RemoteAddress; RemotePort=$_.RemotePort; Process=$p.ProcessName; Path=$p.Path } } | Format-Table -AutoSize. Зафиксируйте подозрительные соединения с процессами, не относящимися к известным приложениям.

5. Экспортируйте результаты в CSV для последующего анализа: Get-NetTCPConnection | Export-Csv -Path connections.csv -NoTypeInformation -Encoding UTF8. Откройте файл в табличном редакторе для удобного анализа.

Задание 3. Анализ журналов событий Windows

1. Получите список доступных журналов событий: Get-WinEvent -ListLog * | Where-Object {$_.RecordCount -gt 0} | Sort-Object RecordCount -Descending | Format-Table LogName, RecordCount, IsEnabled -AutoSize. Зафиксируйтежурналыснаибольшимколичествомзаписей.

2. Получите последние 50 записей из журнала Security: Get-WinEvent -LogName Security -MaxEvents 50 | Format-List TimeCreated, Id, LevelDisplayName, Message. Определите типы событий (Event ID): 4624 – успешный вход, 4625 – неудачный вход, 4634 – выход, 4688 – создание процесса.

3. Получите все записи о неудачных попытках входа за последние 24 часа: $startTime = (Get-Date).AddDays(-1); Get-WinEvent -FilterHashtable @{LogName='Security'; Id=4625; StartTime=$startTime} | Format-ListTimeCreated, Message. Извлеките IP-адреса источников атак из сообщений.

4. Подсчитайте количество неудачных попыток входа по IP-адресам источников: Get-WinEvent -FilterHashtable @{LogName='Security'; Id=4625} | ForEach-Object { $msg = $_.Message; $ip = [regex]::Match($msg, 'SourceNetworkAddress:\s+([0-9\.]+)').Groups[1].Value; if ($ip) { $ip } } | Group-Object | Sort-ObjectCount -Descending | Format-TableCount, Name -AutoSize. Зафиксируйте IP-адреса с наибольшим количеством неудачных попыток.

5. Получите записи журнала System с фильтрацией по уровню Error и Critical: Get-WinEvent -FilterHashtable@{LogName='System'; Level=1,2} -MaxEvents 30 | Format-List TimeCreated, Id, ProviderName, LevelDisplayName, Message. Зафиксируйте критические события и ошибки системы.

6. Получите записи о запуске служб (Event ID 7036) за последний час: $startTime = (Get-Date).AddHours(-1); Get-WinEvent -FilterHashtable@{LogName='System'; Id=7036; StartTime=$startTime} | Format-List TimeCreated, Message. Зафиксируйте службы, которые были запущены или остановлены.

Задание 4. Анализ процессов и служб

1. Получите список всех запущенных процессов с сортировкой по загрузке процессора: Get-Process | Sort-Object CPU -Descending | Select-Object -First 20 | Format-TableName, Id, CPU, WorkingSet, Path -AutoSize. Зафиксируйте процессы с наибольшей загрузкой ресурсов.

2. Получите список запущенных служб и их состояния: Get-Service | Where-Object {$_.Status -eq 'Running'} | Format-TableName, DisplayName, Status -AutoSize. Определите службы, запускаемые автоматически: Get-Service | Where-Object {$_.StartType -eq 'Automatic'} | Format-Table Name, DisplayName, StartType, Status.

3. Получите сведения о процессах, имеющих сетевые соединения: Get-NetTCPConnection -State Established | ForEach-Object { $proc = Get-Process -Id $_.OwningProcess -ErrorActionSilentlyContinue; [PSCustomObject]@{ LocalPort=$_.LocalPort; RemoteAddress=$_.RemoteAddress; RemotePort=$_.RemotePort; Process=$proc.ProcessName; Path=$proc.Path; Company=$proc.Company } } | Format-Table -AutoSize. Зафиксируйте процессы с сетевыми соединениями и проверьте их легитимность.

4. Получите сведения об установленном программном обеспечении: Get-ItemProperty HKLM:\Software\Microsoft\Windows\CurrentVersion\Uninstall* | Where-Object {$_.DisplayName} | Select-Object DisplayName, DisplayVersion, Publisher, InstallDate | Sort-Object DisplayName | Format-Table -AutoSize. Сопоставьте установленные версии с актуальными и выявите устаревшее ПО.

5. Получите сведения об установленных обновлениях Windows: Get-HotFix | Sort-Object InstalledOn -Descending | Format-Table HotFixID, Description, InstalledOn -AutoSize. Проверьте наличие последних обновлений безопасности.

Задание 5. Проверка настроек безопасности

1. Получите сведения о локальных пользователях: Get-LocalUser | Format-Table Name, Enabled, LastLogon, PasswordLastSet, PasswordRequired -AutoSize. Выявите отключённые учётные записи и записи без пароля.

2. Получите сведения о членах группы Administrators: Get-LocalGroupMember -Group Administrators | Format-Table Name, ObjectClass, PrincipalSource -AutoSize. Проверьте, что в группе нет неожиданных учётных записей.

3. Проверьте настройки брандмауэра Windows: Get-NetFirewallProfile | Format-Table Name, Enabled, DefaultInboundAction, DefaultOutboundAction, LogAllowed, LogBlocked -AutoSize. Убедитесь, что все профили включены.

4. Получите сведения о настройках ExecutionPolicyPowerShell: Get-ExecutionPolicy -List. Проверьте, что политика не установлена в значение Unrestricted, которое представляет риск безопасности.

5. Проверьте настройки парольной политики: netaccounts. Зафиксируйте минимальную длину пароля, максимальный срок действия, требования к сложности. Сопоставьте с рекомендациями по обеспечению безопасности.

6. Сформируйте итоговый отчёт по результатам выполнения всех заданий. Отчёт должен содержать: сведения о сетевой конфигурации, список активных соединений с процессами, анализ журналов событий, список процессов и служб, проверку настроек безопасности, выводы о состоянии защищённости системы.

Индивидуальное задание

Студент самостоятельно выбирает инфраструктуру организации для рассмотрения при решении лабораторных задач.

Требования к отчету и защите

Защита предполагает опрос по материалу, изложенному в отчете.

Отчет должен быть оформлен: требуется наличие титульного листа, введение (содержит: указание цели лабораторной работы, задачи, краткие теоретические сведения), последовательное изложение разработанного материала, подписанные скриншоты хода работы, если использовалась компьютерная техника, заключение (выводы, анализ полученных результатов).

Проверяется знание теоретического материала с учетом знаний ответов на **Контрольные вопросы**, приведенные в пункте «**Контрольные вопросы**».

Задаются вопросы по пунктам, приведенным в пункте «Методические указания и порядок выполнения работы».

Критерии к оценке защиты отчета студентом приведены в таблице 1 «Шкала оценок уровня».

Контрольные вопросы

1. Что такое PowerShell и каковы основные отличия PowerShell от классической командной строки cmd?
2. Что такое командлет (cmdlet)? Опишите соглашение по именованию командлетов и приведите примеры.
3. Как PowerShell обрабатывает данные в конвейере? В чём преимущество объектной модели по сравнению с текстовой?

4. Какие командлеты применяются для получения сведений о сетевой конфигурации Windows? Приведите примеры их использования.

5. Как с помощью PowerShell получить список активных TCP-соединений? Какие сведения доступны для каждого соединения?

6. Для чего применяется командлетGet-WinEvent? Чем он отличается от Get-EventLog?

7. Какие журналы событий Windows наиболее важны при аудите безопасности? Какие сведения можно получить из журналов Security, System и Application?

8. Как с помощью PowerShell получить список запущенных процессов и проанализировать их сетевую активность?

9. Чтотакое Execution Policy в PowerShell? Какие режимы существуют и какие риски связаны с некорректной настройкой?

10. Что такое ScriptBlockLogging и какое значение данная функция имеет для расследования инцидентов безопасности?

11. Какие методы удалённого выполнения команд поддерживает PowerShell? Как обеспечить безопасность удалённого управления?

12. Какие признаки подозрительной активности PowerShell следует отслеживать средствами SIEM-систем?

ЛАБОРАТОРНАЯ РАБОТА № 5. ИССЛЕДОВАНИЕ СОВРЕМЕННЫХ СРЕДСТВ ПОИСКА УЯЗВИМОСТИ В ОС

Общие сведения

Цель: исследование современных средств поиска уязвимости в ОС.

Программное обеспечение: ОС MicrosoftWindows XP/2003/2008R2/Vista/7/8/8.1/2012/2012R2, ОС KaliLinux, hping3, nmap, WindowsVulnerabilityScanner.

Методический материал по выполнению лабораторных заданий

Сканеры уязвимостей представляют собой программные или аппаратно-программные средства, предназначенные для автоматизированного выявления уязвимостей в информационных системах. Современные сканеры уязвимостей реализуют два основных подхода: сетевое сканирование (network-based) – удалённая проверка хостов через сеть без установки агентов; хостовое сканирование (host-based) – проверка с установленным на целевой системе агентом, обеспечивающая более глубокий анализ настроек и установленного программного обеспечения.

Основными характеристиками сканеров уязвимостей являются: покрытие базы уязвимостей (CVE, BID, OST); точность выявления (минимизация ложных

срабатываний); производительность сканирования; поддерживаемые типы проверок (сетевые службы, операционные системы, веб-приложения, базы данных, контейнеры); возможность интеграции с SIEM-системами и системами управления уязвимостями (VM). Современные сканеры обеспечивают проверку соответствия настроек требованиям стандартов (PCI DSS, HIPAA, ISO 27001, ФСТЭК) и предоставляют детальные отчёты с рекомендациями по устранению выявленных проблем.

Nessus, разрабатываемый компанией Tenable, является одним из наиболее распространённых коммерческих сканеров уязвимостей. Программа построена по клиент-серверной архитектуре и поддерживает более 100 тысяч проверок, включая сетевые службы, операционные системы, базы данных, веб-приложения, виртуальные инфраструктуры и облачные среды. Nessus обеспечивает гибкую политику сканирования, планирование проверок, интеграцию с системами управления уязвимостями и формирование детальных отчётов для различных аудиторий.

GreenboneVulnerabilityManagement (ранее OpenVAS) представляет собой open-source систему управления уязвимостями, обеспечивающую комплексное сканирование сетевой инфраструктуры. Greenbone включает сервер сканирования, базу данных сетевых тестов (NVT, более 80 тысяч проверок), веб-интерфейс управления и формирования отчётов. Система поддерживает сканирование IPv4 и IPv6-сетей, проверку веб-приложений, контейнеров Docker и сертифицированных средств виртуализации. Greenbone является предпочтительным выбором для организаций, нуждающихся в бесплатном решении корпоративного уровня.

Современные сканеры уязвимостей обеспечивают проверку облачных инфраструктур (AWS, Azure, GCP), контейнерных платформ (Kubernetes, Docker), а также CI/CD-конвейеров. Программные продукты, такие как Tenable.io, QualysCloudPlatform, Rapid7 InsightVM, интегрируются с системами управления конфигурациями (Ansible, Chef, Puppet) и обеспечивают автоматизированное устранение выявленных уязвимостей. Для сканирования веб-приложений применяются специализированные инструменты – BurpSuiteProfessional, Acunetix, OWASP ZAP, обеспечивающие выявление уязвимостей согласно OWASP Top 10.

С позиции информационной безопасности регулярное сканирование уязвимостей является обязательным элементом программой защиты информации, требуемым стандартами PCI DSS (ежеквартальное сканирование), ФСТЭК (при оценке защищённости значимых объектов КИИ) и ISO 27001. Результаты сканирования используются для формирования плана корректирующих мероприятий, оценки эффективности процессов управления уязвимостями и контроля соблюдения политик безопасности. Особое внимание

следует уделять корректной настройке сканеров для минимизации влияния на производительность проверяемых систем и исключения ложных срабатываний.

Сканеры уязвимостей – имеется программные или аппаратные средства, используемые для диагностики и мониторинга сетевых узлов на предмет обнаружения возможных проблем в системе безопасности.

Работу сканера уязвимостей можно разбить на 4 этапа [6]:

1. Обычно, сканер сначала обнаруживает активные IP-адреса, открытые порты, запущенную операционную систему и приложения.
2. Составляется отчёт о безопасности (необязательный шаг).
3. Попытка определить уровень возможного вмешательства в операционную систему или приложения (может повлечь сбой).
4. На заключительном этапе сканер может воспользоваться уязвимостью, вызвав сбой операционной системы или приложения.

Среди сканеров уязвимостей можно выделить:

- сканер портов;
- сканеры, исследующие топологию компьютерной сети;
- сканеры, исследующие уязвимости сетевых сервисов;
- сетевые черви;
- CGI-сканеры (помогают найти уязвимые скрипты) [13];

Рассмотрим некоторые виды сканеров уязвимостей. Сканер Nessus является мощным и надежным средством, указанное относится к семейству сетевых сканеров, позволяющих осуществлять поиск уязвимостей в сетевых сервисах, в межсетевых экранах, фильтрующих маршрутизаторах. Для поиска уязвимостей используются как стандартные средства тестирования и сбора информации о конфигурации и функционировании сети, так и специальные средства, эмулирующие действия злоумышленника по проникновению в системы, подключённые к сети [6].

В программе существует возможность включения собственных проверочных процедур. Для данного в сканере предусмотрен специальный язык сценариев, названный NASL (NessusAttackScriptingLanguage).

WindowsVulnerabilityScanner (WVS) – небольшая бесплатная утилита, указанная сканирует операционные системы Windows на предмет обнаружения возможных уязвимостей в системе безопасности ОС и позволяет осуществить подбор соответствующих патчей для указанных устранений. Программасовместимассистемами Windows XP (Home & Professional), Windows 2000 Server & Professional и Windows 2003. По окончании работы WindowsVulnerabilityScanner выводит список обнаруженных брешей, соответствующий уровень опасности и ссылку на патч, указанных устраняющий. Ведется постоянное логирование всех действий и результатов работы WVS [6].

MBSA (MicrosoftBaselineSecurityAnalyzer) предназначен для определения уязвимых мест в системе безопасности компьютеров под управлением Windows NT 4.0, Windows 2000, Windows XP и WindowsServer2003. Сканер находит типичные ошибки в конфигурации операционной системы Windows, служб InternetInformationServices (IIS), сервера SQL Server, обозревателя InternetExplorer и приложений Office. Кроме того, средство MBSA выявляет отсутствующие обновления безопасности для Windows, служб IIS, сервера SQL Server, обозревателя InternetExplorer, проигрывателя WindowsMedia, сервера ExchangeServer, компонентов доступа к данным Microsoft (MDAC), модуля Microsoft XML (MSXML), виртуальной машины Microsoft (VM), приложений ContentManagementServer, CommerceServer, BizTalkServer, HostIntegrationServer и пакета Office (допускается проверка только локальной системы). Версия 1.2.1 имеет графический пользовательский интерфейс, но может быть запущена и в режиме командной строки [6].

Nmap позволяет просканировать сеть и узнать, какие узлы к указанному узлу подключены, а также определить, какие сервисы на указанных хостах запущенны. В nmap есть возможность выполнять специальные скрипты, позволяющие определить некоторые уязвимости в определенном программном обеспечении.

Программа ScanOVAL предназначена для оперативного автоматизированного обнаружения уязвимостей программного обеспечения на рабочих станциях и серверах, функционирующих под управлением операционных систем семейства MicrosoftWindows.

Программа ScanOVAL обеспечивает выполнение следующих функций:

– загрузка XML-файлов с OVAL-описаниями уязвимостей, выполненными в соответствии со стандартом «The OVAL LanguageSpecification» версии не ниже 5.10.1;

– обнаружение (на основании результатов обработки данных, представленных в XML-файлах) уязвимостей программного обеспечения, установленного на локальной ПЭВМ, работающей под управлением операционной системы семейства MicrosoftWindows.

OpenVAS – сканер уязвимостей с открытым исходным кодом. OpenVAS предназначен для активного мониторинга узлов ЛВС на предмет наличия проблем, связанных с безопасностью, а также для оценки серьезности проблем безопасности информации.

Активный мониторинг означает, что OpenVAS выполняет следующее: сканирует открытые порты, посылает специальным образом сформированные пакеты для имитации атаки, получает доступ к консоли управления, и выполняет на консоли команды.

OpenVAS анализирует собранные данные и делает выводы о наличии каких-либо проблем с безопасностью. Данные проблемы, в большинстве случаев касаются установленного на узле необновленного ПО, где в указанном ПО имеются известные и описанные уязвимости, или же небезопасно настроенного ПО [7].

В основе работы OpenVAS лежит постоянно пополняемая коллекция NVT-тестов безопасности (тестов уже больше 58000), а также подключение к базе CVE, описывающей известные уязвимости. Исполнение NVT тестов позволяет выявить уязвимость, а CVE обеспечивает описание проблемы и способы указанного решения.

В ходе указанной работы OpenVAS может сканировать открытые порты, посылать сформированные специальным образом пакеты для авторизации на узле (либо имитации атаки), получать доступ к консоли и пробовать выполнять на консоли команды. На основе таких манипуляций собираются данные, за счет анализа данных можно сделать вывод об отсутствии, либо наличии проблем с безопасностью.

OpenVAS построен на клиент-серверной схеме и состоит из нескольких сервисов и инструментов. Ядром является сервис OpenVASScanner, указанный и выполняет сканирование (работает на 9391 порту). Причем в процессе проверок систем используются известные продукты: nmap и pnscan, ike-scan, pnscan, strobe и другие. Вся интеллектуальная часть содержится в OpenVASManager (порт 9390), указанная может собирать и анализировать результаты, полученные с нескольких установок OpenVAS. Имеется основной сервис, обеспечивающий полной контроль над уязвимостями, реализованы управление политиками, сканирование по расписанию, обнаружение ложных срабатываний, представление отчетов в самых разных форматах (XML, HTML, LateX и др.). Для управления сканерами используется протокол OTP (OpenVASTransferProtocol), указанный может использовать команды в XML виде пр использовании OpenVASManagementProtocol (OMP). Все настройки и собранная информация хранятся централизованно в SQL базе данных (SQLite)[7]. Одной из менее известных функций OpenVAS является интерфейс командной строки, с указанным можно взаимодействовать с помощью команды «omp». Можно использовать различные скрипты для удобства. Например, скрипт установки OpenVAS «openvas-setup» или bash-скрипт для автоматизации сканирования.

Методические указания и порядок выполнения работы. Задание

В рамках лабораторной работы выполняются следующие задачи:

1. Установка и настройка сканера уязвимостей (Nessus, Greenbone/OpenVAS или аналогичного).

2. Настройка политики сканирования и выбор целевых хостов для проверки.

3. Запуск сканирования и анализ хода выполнения проверки.

4. Анализ результатов сканирования: выявленные уязвимости, их критичность, рекомендации по устранению.

5. Сравнительный анализ результатов сканирования различных типов хостов (Windows, Linux).

6. Применение специализированных сканеров для проверки веб-приложений (BurpSuite, OWASP ZAP).

7. Формирование итогового отчёта по результатам сканирования с рекомендациями по устранению выявленных уязвимостей.

Задание 1. Установка и настройка GreenboneVulnerabilityManagement

1. Установите Greenbone (OpenVAS) на Linux: sudoaptupdate&&sudoaptinstallgvm. После установки выполните первоначальную настройку: sudogvm-setup. Процесс настройки включает загрузку баз уязвимостей (NVT, SCAP, CERT) и может занять значительное время (30–60 минут).

2. Проверьте корректность установки: sudogvm-check-setup. Команда проверяет наличие всех компонентов, актуальность баз и доступность сервисов. При выявлении ошибок следуйте рекомендациям команды для их устранения.

3. Запустите сервисы Greenbone: sudogvm-start. Проверьте статус: sudosystemctlstatusgvmdgsadospd-openvas. Убедитесь, что все сервисы активны.

4. Откройте веб-интерфейс в браузере: https://127.0.0.1:9392. При первом подключении примите самоподписанный сертификат TLS. Авторизуйтесь с учётными данными, заданными при установке (по умолчанию admin / admin, при первом входе потребуется сменить пароль).

5. В веб-интерфейсе откройте раздел Administration → FeedStatus и убедитесь, что базы NVT (NetworkVulnerabilityTests), SCAP (SecurityContentAutomationProtocol) и CERT актуальны. При необходимости выполните обновление: sudogvm-feed-update.

Задание 2. Настройка цели сканирования и запуск проверки

1. В веб-интерфейсе откройте раздел Configuration → Targets и создайте новую цель. Укажите: имя цели (например, Lab-Target); IP-адрес целевого хоста; порт для доступа (по умолчанию 22 для SSH или 135 для WMI); учётные данные для аутентифицированного сканирования (при наличии).

2. Настройте параметры обнаружения активных хостов: в поле AliveTest выберите ScanConfigurationDefault (использовать настройки из конфигурации сканирования). При сканировании в локальной сети рекомендуется использовать ARP-пинг для надёжного обнаружения.

3. Откройте раздел Scans → Tasks и создайте новую задачу сканирования. Укажите: имя задачи; цель сканирования; конфигурацию сканирования (Fullandfast– полный и быстрый, рекомендуемая для большинства случаев); расписание (при необходимости периодического сканирования).

4. Запустите задачу, нажав кнопку Start (зелёная стрелка). Наблюдайте за ходом выполнения через интерфейс – отображается прогресс сканирования, количество проверенных хостов, выявленных уязвимостей и текущее состояние.

5. После завершения сканирования откройте результаты. Проанализируйте сводную информацию: количество уязвимостей по категориям критичности (Critical, High, Medium, Low, Log); список выявленных служб и операционных систем; общее время сканирования.

Задание 3. Анализ результатов сканирования

1. Для каждой выявленной уязвимости высокой критности (Critical и High) изучите детальную информацию: CVE-идентификатор; CVSS-оценка (вектор атаки, сложность, требуемые привилегии, влияние на конфиденциальность, целостность, доступность); описание уязвимости; затронутые компоненты и версии; рекомендации по устранению.

2. Сформируйте сводный отчёт в формате PDF: в веб-интерфейсе откройте результаты сканирования → Report → Downloads → PDF. Отчёт содержит структурированное описание всех выявленных уязвимостей с рекомендациями.

3. Сопоставьте выявленные уязвимости с доступными обновлениями. Для каждой уязвимости определите: наличие обновления от производителя; возможность применения временных мер защиты (изменение конфигурации, блокировка порта); критичность устранения в зависимости от контекста эксплуатации.

4. Выполните повторное сканирование после применения обновлений и сравните результаты с предыдущим сканированием. Зафиксируйте устранённые и оставшиеся уязвимости.

Задание 4. Сканирование веб-приложения с применением OWASP ZAP

1. Установите OWASP ZAP: sudoaptinstallzaproxy (Linux) или скачайте с owasp.org/zap (Windows). Запустите ZAP и создайте новую сессию: File → NewSession.

2. Настройте браузер для проксирования через ZAP. В настройках прокси-сервера браузера укажите: адрес 127.0.0.1, порт 8080. Импортируйте сертификат CA ZAP: в ZAP откройте Tools → Options → Network → ServerCertificates → Generate, затем импортируйте сертификат в браузер.

3. В поле QuickStart введите URL тестового веб-приложения (например, http://testphp.vulnweb.com или локального DVWA). Нажмите Attack– ZAP

выполнит сканированиеSpider (обход структуры приложения) и ActiveScan (активную проверку уязвимостей).

4. После завершения сканирования откройте вкладку Alerts. Проанализируйтевыявленныеуязвимости, сгруппированныепотипам: Injection (SQL, Command, LDAP); XSS (Reflected, Stored, DOM); Broken Authentication; Broken Access Control; Security Misconfiguration; Cross-Site Request Forgery.

5. Для каждой выявленной уязвимости изучите описание: тип уязвимости, критичность (Risk), уверенность (Confidence), описание, URI, параметр, доказательство эксплуатации (Evidence), рекомендации по устранению (Solution).

6. Сформируйтеотчёт: Report → GenerateReport → HTML. Отчёт содержит структурированное описание всех выявленных уязвимостей с рекомендациями.

7. Сформируйте итоговый отчёт по результатам выполнения всех заданий. Отчёт должен содержать: описание настройки сканеров, результаты сканирования сетевой инфраструктуры и веб-приложения, анализ выявленных уязвимостей с рекомендациями по устранению, сравнительный анализ различных типов сканирования.

Индивидуальное задание

Студент самостоятельно выбирает инфраструктуру организации для рассмотрения при решении лабораторных задач.

Требования к отчету и защите

Защита предполагает опрос по материалу, изложенному в отчете.

Отчет должен быть оформлен: требуется наличие титульного листа, введение (содержит: указание цели лабораторной работы, задачи, краткие теоретические сведения), последовательное изложение разработанного материала, подписанные скриншоты хода работы, если использовалась компьютерная техника, заключение (выводы, анализ полученных результатов).

Проверяется знание теоретического материала с учетом знаний ответов на **Контрольные вопросы**, приведенные в пункте «**Контрольные вопросы**».

Задаются вопросы по пунктам, приведенным в пункте «Методические указания и порядок выполнения работы».

Критерии к оценке защиты отчета студентом приведены в таблице 1 «Шкала оценок уровня».

Контрольные вопросы

1. Что такое уязвимость информационной системы? Как классифицируются уязвимости по причине возникновения и критичности?

2. Что такое CVE, CVSS и NVT? Какова роль этих стандартов и баз в процессе управления уязвимостями?

3. Опишите принцип работы сетевого сканера уязвимостей. Чем он отличается от хостового сканера?

4. Что такое ложные срабатывания (falsepositives) и ложные пропуски (falsenegatives) при сканировании уязвимостей? Как минимизировать их количество?

5. Опишите архитектуру и основные возможности Nessus. В каких редакциях выпускается продукт?

6. Опишите архитектуру и основные возможности GreenboneVulnerabilityManagement (OpenVAS). Какие преимущества обеспечивает open-source подход?

7. Какие типы проверок выполняют сканеры уязвимостей? Чем сетевое сканирование отличается от сканирования веб-приложений?

8. Что такое OWASP Top 10? Какие категории уязвимостей входят в актуальный перечень OWASP Top 10 2021?

9. Какие специализированные сканеры применяются для проверки веб-приложений? Опишите их возможности.

10. Какие требования к регулярности сканирования уязвимостей предъявляют стандарты PCI DSS и ФСТЭК?

11. Как интегрируются сканеры уязвимостей с SIEM-системами и системами управления уязвимостями (VM)?

12. Какие меры следует применять для минимизации влияния сканирования на производительность проверяемых систем?

ЛАБОРАТОРНАЯ РАБОТА № 6. ОСНОВЫ НАСТРОЙКИ И ЭКСПЛУАТАЦИИ СИСТЕМЫ ОБНАРУЖЕНИЯ ВТОРЖЕНИЙ SNORT

Общие сведения

Цель: ознакомиться на практике с методами и средствами обнаружения вторжений.

Программное обеспечение: Компьютерная лаборатория с ЛВС, IDS Snort 2.6.1.4, снифферWinDump 3.8.3, сканер NMap 4.20, библиотека WinPcap 4.0, Unicorn, XSpider, hping3.

Методический материал по выполнению лабораторных заданий

Snort – бесплатно распространяемая система обнаружения вторжений, созданная Марти Решем. СОВ Snort современный инструмент обнаружения вредоносного трафика, является прототипом для многопоточных СОВ Suricata.

Snort функционирует в среде в UNIX любого вида и Windows. На WEB-сайте www.snort.org находится набор доступных правил, указанный обновляется.

Исполняемый сервис Snort в линуксоидных операционных системах ОС или в Windows, разворачивается в памяти и осуществляет фильтрацию сетевого трафика, реализуя таким образом сниффинг потоков данных, выступая как сниффер. Перехваченные пакеты поступают в анализатор, указанный, на основе параметров подключенных в конфигурации СОВ правил, определяемых администратором, дифференцирует по результатам анализа пакеты данных на вредоносные или легальные, сигнализируя об данном в логах или консоли администратора (информация поступает в систему предупреждения и регистрации). К анализатору могут быть подключены сторонние модули, для увеличения данного функциональности. Таким образом, распознаются разнообразные атаки типа переполнения буфера, скрытых просмотров порта, CGI-атак, попыток определения ОС и т. д. [14].

Snort может работать в трех различных режимах:

1. Режим пакетного сниффера. Когда Snort работает в данном режиме, данный объект читает и декодирует все сетевые пакеты и формирует дамп.

2. Режим регистрации пакетов. Данный режим записывает пакеты на диск и декодирует указанных в ASCII формат.

3. Режим обнаружения вторжений. Имеется основной режим и предназначен непосредственно для обнаружения атак и ведения log-файла.

Модуль распознавания СОВ использует сопоставительный анализ. Для выполнения операции сопоставления используются заранее заданные сигнатуры атак, включающие маркеры вредоносного трафика, и полученный с помощью снифинга трафик. Таким образом, сигнатуры включают в соответствующ состав ключевые параметры, слова или выражения, указанные являются маркерами вредоносной деятельности.

Методы обнаружения атак:

1. Сигнатуры, основанные на контроле частоты событий или превышении пороговой величины. В данных сигнатурах указаны периоды, в границах указанных фиксируются события, а также число фиксируемых событий. Если число фиксаций превышает заданные заранее показатели производится запись предупреждения в лог-файл, либо вывод предупреждения на экран (примером подобный сигнатуры является обнаружение сканирования портов или обнаружение атаки SYN Flood, в сигнатуре указывается пороговое просканированных число портов в заданный период времени).

2. Сигнатуры, основанные на шаблоне (pattern-basedsignatures), выражении или строке, характеризующих атаку или иную подозрительную деятельность.

Например, фрагмент «cwd ~root» в FTP-сеансе однозначно определяет факт обхода механизма аутентификации на FTP-сервере и попытке перейти в корневой каталог FTP-сервера. Маркер «CA FE BA BE указывает на присутствие апплетов Java в трафике на основе фрагмента, что позволяют обнаруживать многих троянских коней, при эксплуатации принятых «по умолчанию» портов [15].

3. Обнаружение аномалий. Сначала определяется шаблон при сопоставлении, с указанным производится обнаружение отклонения, указанное считается признаком нелегитимного трафика (например, фиксирует процедуры успешного подключения к сервису в заданный как подозрительный период времени)

Опции командной строки Snort:

1. Режимы обнаружения. Синтаксис команды: snort [-Режимы обнаружения] <Опции >

Режимы обнаружения:

-v или -vde– режим пакетного сниффера (режим анализа). Например: snort -v.

-l <directorytologpacketsto> – режим регистрации пакетов (режим протоколирования). Например: snort -vde -l c:/var/log/snort.

-dev -l <logdirectory> -h <host|net> -c <config> – режим обнаружения вторжений. Например: snort -de -l /var/log/snort -h 192.168.1.0/24 -c /etc/snort/snort.conf.

Опции Snort можно получить, запустив Snort без параметров. Опции команды:

-f full – полная информация о сигнале, включая прикладные данные. Имеется подразумеваемый режим сигнализации. Данный объект будет использоваться при отсутствии спецификаций.

-A fast – быстрый режим. Протоколируются только заголовки пакетов и тип сигналов. Имеется полезно в очень быстрых сетях, но если требуется дополнительная судебная информация, необходимо использовать опцию full.

-A unsock – посылает сигнал в UNIX-сокет с указанным номером, для прослушивания другая программа.

-A none – отключает сигналы тревоги.

Конфигурационным файлом служит snort.conf, указанный по-умолчанию помещается в /etc/snort.conf. В данном файле задаются все настройки Snort. Имя данного файла можно изменить, если при запуске Snort после ключа -c указать новое маршрутное имя.

Для редактирования конфигурационного файла необходимо использовать любой текстовой редактор. Многие строки в данном файле начинаются со знака #, за указанным следуют различные комментарии. Знак # служит стандартным началом строк комментариев, указанные многие интерпретаторы, командные или такие как Perl, игнорируют. Строки комментариев применяются для документирования программ или для отключения старого кода. Строками, реально воздействующими на конфигурацию, являются строки без знака # в начале. Остальные присутствуют только для информационных целей.

Конфигурационный файл snort.conf следует изменять следующим образом (настраивая конфигурацию СОВ):

При установке параметров домашней сети необходимо сообщить Snort адреса, указанные представляют домашнюю сеть, чтобы данный объект мог правильно интерпретировать внешние атаки. Имеется делается с помощью инструкции:

var HOME_NET адреса,

где адреса нужно заменить на реальное адресное пространство ЛВС. Если имеется несколько сетей, то можно ввести все сети, разделяя запятыми.

Можно также ввести имя интерфейса, чтобы IP-адрес и маска сети, присвоенные этому интерфейсу, использовались как HOME_NET. Для данного служит следующая инструкция:

var HOME_NET $ имя_интерфейса,

где имя_интерфейса заменяется интерфейсом, указанный должен слушать Snort (например, eth0 или eth1).

С помощью аналогичной инструкции можно определить внешние сети, заменяя HOME_NET на EXTERNAL_NET. Подразумеваемым значением обеих переменных служит any. Можно оставить указанных в таком виде или определить одну или обе. Рекомендуется определить внутреннюю сеть, но оставить внешние сети заданными как any.

В конфигурационном файле можно определить ряд серверов, включая WEB, MAIL, DNS, SQL и TELNET. Имеется уменьшит число ложных срабатываний для данных сервисов на данных машинах.

Можно также задать номера портов для данных сервисов, чтобы регистрировались только атаки на указанные порты. Все данные опции конфигурации позволяют сократить число ложных срабатываний, чтобы до при выполнении доводилась только информация, обладающая реальной ценностью. Имеется также раздел для добавления серверов AIM, чтобы отслеживать применение AOL InstantMessenger. Имеется имеет смысл только в том случае, если включен класс правил Chat.

При конфигурировании декодировщиков и препроцессоров Snort нужно использовать ряд ключей и настроек в конфигурационном файле, указанные управляют декодировщиками и препроцессорами Snort. Данные процедуры применяются к трафику, прежде чем данный объект пройдет через какой-либо набор правил, обычно с целью правильного форматирования или для обработки определенного вида трафика, указанный проще препроцессировать, чем применять наборы правил. Примером подобного типа трафика служат фрагментированные пакеты. В Snort присутствует декодировщик, собирающий фрагментированные пакеты. Многие атаки пытаются скрыть истинную природу, фрагментируя пакеты, так что описываемые возможности Snort являются весьма ценными.

Другой декодировщик предназначен для пакетов сканирования портов, так как данные имеют склонность приходить группами и в большом количестве, лучше обрабатывать указанных заранее общей массой, чем пытаться сравнивать каждый пакет с сигнатурой. Имеется также делает систему обнаружения вторжений более защищенной от атак на доступность. Подразумеваемые настройки для данных подсистем должны работать хорошо, однако, получив некоторый опыт работы со Snort, можно попытаться изменить указанных, чтобы повысить производительность и достичь лучших результатов.

Конфигурирование модулей вывода применяется, если нужно использовать базу данных при обработке вывода Snort. Здесь требуется указать программе, как обрабатывать данные сигналов тревоги. Имеется несколько модулей вывода, указанные можно применять в зависимости от требуемого формата данных: Syslog, Database и новый модуль Unified, поддерживающий универсальный бинарный формат, полезный для импорта данных другими программами.

Общий формат для конфигурирования модулей вывода:

outputимя_модуля: конфигурация опции, где имя_модуля следует заменить на alert_syslog, database или alert_unified в зависимости от используемого модуля.

Опции конфигурации для различных модулей вывода Syslog. Для систем UNIX/Linux нужно использовать следующую директиву:

output alert_syslog: LOG_AUTH LOG_ALERT

Для Windows-систем можно использовать любой из следующих форматов:

output alert_syslog: LOG_AUTH LOG_ALERT

output alert_syslog: host=имя_хоста, LOG_AUTH LOG_ALERT

output alert_syslog: host=имя_хоста:порт, LOG_AUTH LOG_ALERT,

где имя_хоста и порт нужно заменить, соответственно, на IP-адрес и порт сервера Syslog.

Общий формат для настройки вывода в базу данных:

outputdatabase: log, тип_базы_данных, user=имя_пользователя

password=пароль dbname=имя_БДhost=адрес_БД,

где тип_базы_данных заменяется одной из допустимых для Snort разновидностей баз данных (MySQL, postgresql, unixodbc или mssql), имя_пользователя – допустимым именем пользователя машины базы данных, а пароль – данного паролем. Переменная dbname задает имя базы данных для протоколирования. Наконец, адрес_базы_данных является IP-адресом сервера базы данных. Не рекомендуется устанавливать Snort и базу данных на один сервер. Помимо того, что безопаснее держать данные сигналов тревоги на другом компьютере, работа Snort и базы данных на одной машине будет существенно снижать производительность.

Unified - основной бинарный формат быстрого протоколирования и сохранения для будущего использования. Поддерживаются два аргумента – filename и limit, а вся директива может выглядеть примерно так:

output alert_unified: filename snort.alert, limit 128

Можно выполнить тонкую настройку Snort, добавляя или удаляя наборы правил. Файл snort.conf позволяет добавлять или удалять целые классы правил. В конце файла перечислены все наборы правил генерации сигналов тревоги.

Можно отключить целые категории правил, закомментировав строки с помощью знака # в начале. Например, можно отключить все правила icmp-info для снижения числа ложных срабатываний на трафике ping или все правила NetBIOS, если в сети нет машин Windows. Имеются также общедоступные наборы правил, уже настроенные для определенных сред.

Например, для тестирования работы Snort можно внести в конец файла snort.conf следующее правило:

alert ip any any -> any any (msg:"IP Packet detected";).

Первое слово «alert» говорит о том, как snort будет реагировать на пакеты, указанные попадут под критерии указанные далее.

Второе слово «ip» говорит о том, что под данное правило попадают все IP пакеты.

Первое слово «any» определяет IP адрес источника пакета, в данном случае – с любого источника.

Второе слово «any» определяет порт источника пакета, в данном случае, пакет может быть отправлен с любого порта.

Далее символы «->» определяют направление пакета.

Третье и четвертое слова «any» определяют соответственно IP адрес и порт назначения пакета. В данном случае под указанное правило попадает пакет, направленный на любой порт любого IP адреса.

И последняя группа – параметры шаблона пакета. Пример содержит только параматр «msg» – данное сообщение, указанное будет записано в лог-файл.

Таким образом, указанное правило является «плохим», так как будет реагировать на все IP-пакеты. Но для проверки конфигурации Snort вполне годится.

Следующее правило реагирует на все ICMP (InternetControlMessageProtocol, описан в RFC 792) пакеты:

alert icmp any any -> any any (msg:"ICMP Packet detected";).

чтобы протестировать данный правило, можно запустить программу ping и проверить, что snort запишет в соответствующие log-файлы.

Как говорилось выше, в скобках перечислены параметры шаблона пакета. Например, можно было бы использовать параметр ttl для выбора ICMP пакетов с полем ttl установленным в определенное значение:

alert icmp any any -> 10.116.1.112/32 any (msg:"ICMP Packet with TTL=100 detected"; ttl:100;).

данный правило реагирует на ICMP пакеты с временем жизни равным 100 и направленные на единственный адрес 10.116.1.112.

Итак, все правила Snort делятся на две части: заголовок и параметры. Заголовок имеет следующие поля:

1. Действие.
2. Протокол.
3. Адрес.
4. Порт.
5. Направление.
6. Адрес.
7. Порт.

Поле «Действие» определяет то, как snort будет поступать при нахождении пакета удовлетворяющего данному правилу. Поле «Протокол» определяет, какие пакеты будут проверяться. Поля «Адрес» и «Порт» определяют соответственно адрес и порт некоторого узла. Какая пара данных полей определяет источник, а какая приемник зависит от поля «Направление». Так, если поле «Направление» равно «->», то источник будет определяться парой «Адрес», «Порт» стоящими слева, а приемник парой, стоящей справа.

Возможные значения поля «Направление»: «->», «<-», «<>». Последнее значение говорит о том, что источником может быть как левая пара, так и правая, а приемником, соответственно, правая и левая.

Для протоколов транспортного уровня TCP и UDP поля «Порт» определяют порт источника и приемника, а для протоколов сетевого уровня, таких как IP и ICMP, не являются значащими, т. е. игнорируются.

Возможными значениями для поля «Действие» могут быть:

−pass говорит snort’у игнорировать пакеты;

−log говорит о том, чтобы snort записал информацию об данном пакете либо в log-файл, либо в базу данных;

−alert посылает информацию о пакете либо в файл, либо на консоль; отличие от действия log в том, что alert сначала посылает сигнал, а потом пишет информацию о пакете в log-файл;

−activate используется для активации других шаблонов с действием dynamic, описанным далее, для проверки других параметров;

−dynamic действие, указанное активируется действием с полем «Действие» равным activate;

−определенное пользователем действие.

Возможные значения для поля «Протокол»: ip, icmp, tcp, udp.

Адресом может быть IP-адрес, с указанием маски после символа «/» – число, указывающее количество установленных бит в маске. Так, адрес 192.168.0.0/24 определяет диапазон адресов 192.168.0.0-192.168.0.255.

Для перечисления адресов необходимо использовать разделитель «,». Для исключения адресов необходимо поместить указанных в квадратные скобки, перед указанными поставить символ «!».

Например, следующее правило фиксирует все ICMP пакеты с временем жизни равным 100, адресом источника указанных не являются сети 192.168.2.0/24 и 192.168.8.0/24:

alert![192.168.2.0/24, 192.168.8.0/24] any -> any any (msg:"Ping with TTL=100"; ttl:100;).

Номер порта – имеется число, либо диапазон, определяется минимальным номером порта и максимальным, разделенными символом «:». Либо, если отсутствует номер минимального порта, в диапазон входят все номера портов меньшие максимального номера. Если отсутствует максимальный номер порта, то в диапазон входят все номера портов большие или равные минимальному номеру порта. Для определения всех портов кроме заданных используется символ «!» стоящий перед номером либо диапазоном.

Пример определения диапазона портов:

alertudpany 1024:2048 ->anyany (msg:"UDP ports";)

Параметры правила задаются в круглых скобках в соответствии с форматом записи параметров:

(Имя параметра, символ «:», значение параметра, символ «;»)

Методические указания и порядок выполнения работы. Задание

Цель лабораторной работы предполагает освоение основ настройки и эксплуатации системы обнаружения вторжений Snort, изучение структуры правил сигнатур, приобретение навыков настройки конфигурации и анализа зарегистрированных событий.

В рамках лабораторной работы выполняются следующие задачи:

1. Установка и настройка Snort на операционной системе Linux.
2. Изучение структуры файла конфигурации snort.conf и основных переменных.
3. Освоение синтаксиса правил Snort и создание собственных сигнатур.
4. Запуск Snort врежимах sniffer, packet logger и IDS.
5. Генерация тестового трафика и анализ зарегистрированных событий.
6. Настройка препроцессоров для нормализации и анализа трафика.
7. Интеграция Snort с системой журналирования и формирование отчётов по результатам работы.
8. Настроить СОВ.
9. Применить СОВ для обнаружения вторжений.

Задание 1. Установка и настройка Snort 3

1. Установите зависимости Snort: sudoaptinstallbuild-essentialcmakelibpcap-dev libpcre2-dev libdnet-devlibdaq-dev libnghttp2-dev zlib1g-dev pkg-config libluajit-5.1-dev libssl-devlibhyperscan-dev.

2. Установите Snort через пакетный менеджер или из исходников. Для установки из репозитория: sudoaptinstallsnort. При установке укажите диапазон адресов локальной сети (HOME_NET), например 192.168.1.0/24.

3. Проверьте установленную версию: snort --version. Определите поддерживаемые препроцессоры и DAQ (DataAcquisition) модули: snort --daq-list.

4. Откройте файл конфигурации в текстовом редакторе: sudonano /etc/snort/snort.conf. Изучите структуру файла: раздел переменных сетевых сегментов (HOME_NET, EXTERNAL_NET, DNS_SERVERS, SMTP_SERVERS); раздел настройки препроцессоров; раздел подключения баз правил; раздел настройки вывода.

5. Настройте переменные сетевых сегментов в соответствии с топологией лабораторной сети: var HOME_NET 192.168.1.0/24; var EXTERNAL_NET !$HOME_NET; var DNS_SERVERS $HOME_NET. Проверьте пути к базам правил: var RULE_PATH /etc/snort/rules.

6. Проверьте настройки препроцессоров. Убедитесь, что включены: preprocessor frag3_global и frag3_engine (для сборки фрагментов); preprocessor stream5_global и stream5_tcp (для отслеживания TCP-сессий); preprocessorhttp_inspect (для нормализации HTTP); preprocessordns (для анализа DNS).

7. Протестируйте корректность конфигурации: sudosnort -T -c /etc/snort/snort.conf -i <интерфейс>. Команда -T запускает тестовый режим, при котором Snort проверяет конфигурацию без запуска захвата. Убедитесь в отсутствии ошибок и предупреждений.

Задание 2. Запуск Snort в режимах сниффера и записи пакетов

1. Запустите Snort в режиме сниффера с выводом заголовков пакетов: sudosnort -v -i <интерфейс>. Наблюдайте за выводом на экран. Сгенерируйте тестовый трафик (ping, веб-запросы) и зафиксируйте наблюдаемые пакеты.

2. Запустите Snort с декодированием заголовков: sudosnort -vd -i <интерфейс>. Опция -d добавляет вывод данных пакета. Сравните объём и детализацию выводимой информации с предыдущим режимом.

3. Запустите Snort с полным декодированием: sudosnort -vde -i <интерфейс>. Опция -e добавляет вывод заголовков канального уровня (Ethernet). Зафиксируйте структуру декодированных пакетов.

4. Запустите Snort в режиме записи пакетов в файл: sudosnort -l /var/log/snort -i <интерфейс>. Сгенерируйте тестовый трафик (ping, веб-запросы, DNS-запросы) и остановите Snort через 60 секунд. Проанализируйте записанные файлы в каталоге /var/log/snort/.

Задание 3. Создание пользовательских правил и запуск в режиме IDS

1. Откройте файл локальных правил: sudonano /etc/snort/rules/local.rules. Этот файл предназначен для пользовательских правил и не перезаписывается при обновлении баз.

2. Создайте правило для выявления ICMP-пинга: alert icmp any any -> $HOME_NET any (msg:"ICMP Ping detected"; sid:1000001; rev:1;). Правило генерирует предупреждение при получении ICMP-пакета извне. Поле sid (Snort ID) должно быть уникальным и находиться в диапазоне 1000000–1999999 для пользовательских правил.

3. Создайте правило для выявления SYN-сканирования: alert tcp $EXTERNAL_NET any -> $HOME_NET any (flags:S,12; msg:"SYN scan attempt"; detection_filter:trackby_src, count 10, seconds 5; sid:1000002; rev:1;). Опция detection_filter генерирует предупреждение при получении 10 SYN-пакетов от одного источника за 5 секунд.

4. Создайте правило для выявления SSH-брутфорс: alert tcp $EXTERNAL_NET any -> $HOME_NET 22 (msg:"SSH brute force attempt"; flow:to_server,established; content:"SSH-"; nocase; detection_filter:trackby_src, count 5, seconds 60; sid:1000003; rev:1;). Опция flow:to_server,established ограничивает правило пакетами в направлении сервера в установленных соединениях. Опция content:"SSH-" проверяет наличие строки в полезной нагрузке.

5. Создайте правило для выявления SQL-инъекцийвHTTP-запросах: alerttcpanyany -> $HOME_NET 80 (msg:"SQLInjectionattempt"; flow:to_server,established; content:"SELECT"; nocase; content:"FROM"; nocase; distance:0; within:50; sid:1000004; rev:1;). Правство проверяет наличие последовательности SELECT...FROM в HTTP-запросе.

6. Протестируйте корректность конфигурации: sudosnort -T -c /etc/snort/snort.conf. Убедитесь в отсутствии ошибок синтаксиса.

7. Запустите Snort в режиме IDS с выводом предупреждений в консоль: sudosnort -A console -q -c /etc/snort/snort.conf -i <интерфейс>. Опция -A console выводит предупреждения в консоль, -q подавляет вывод пакетов.

8. Сгенерируйте тестовый трафик, соответствующий созданным правилам: выполните ping с внешнего хоста (правило 1); выполните SYN-сканирование с применением nmap (правило 2); выполните множественные SSH-подключения с неверным паролем (правило 3); отправьте HTTP-запрос с параметром, содержащим 'SELECT...FROM' (правило 4).

9. Зафиксируйте сгенерированные Snort предупреждения в консоли. Для каждого срабатывания определите: время, сигнатуру (msg), sid, адреса источника и назначения, протокол и порт. Сопоставьте срабатывания с правилами, вызвавшими их.

Задание 4. Настройка журналирования и анализа событий

1. Настройте вывод событий в файл syslog. Bsnort.confдобавьте: output alert_syslog: LOG_AUTH LOG_ALERT. Перезапустите Snort. Проверьтежурнал: sudo tail -f /var/log/syslog | grep snort.

2. Настройтевыводсобытийвформате JSON дляинтеграциис SIEM: output alert_json: /var/log/snort/alert.json. Формат JSON обеспечиваетудобныйпарсингвнешнимисистемами.

3. Установите и настройте Barnyard2 для эффективной обработки событий: sudoaptinstall barnyard2. Настройте вывод событий в базу данных MySQL или внешнюю SIEM-систему.

4. Проанализируйте зарегистрированные события. Для каждого типа атаки (ICMP-пинг, SYN-сканирование, SSH-брутфорс, SQL-инъекция) сформируйте описание: сигнатура, характерные признаки, рекомендованные меры защиты.

5. Сформируйте итоговый отчёт по результатам выполнения всех заданий. Отчёт должен содержать: описание установки и настройки Snort, созданные правила с пояснениями, результаты тестирования правил, настройку журналирования, выводы по эффективности обнаружения атак.

Индивидуальное задание

Студент самостоятельно выбирает инфраструктуру организации для рассмотрения при решении лабораторных задач.

Требования к отчету и защите

Защита предполагает опрос по материалу, изложенному в отчете.

Отчет должен быть оформлен: требуется наличие титульного листа, введение (содержит: указание цели лабораторной работы, задачи, краткие теоретические сведения), последовательное изложение разработанного материала, подписанные скриншоты хода работы, если использовалась компьютерная техника, заключение (выводы, анализ полученных результатов).

Проверяется знание теоретического материала с учетом знаний ответов на **контрольные вопросы**, приведенные в пункте «**Контрольные вопросы**».

Задаются вопросы по пунктам, приведенным в пункте «Методические указания и порядок выполнения работы».

Критерии к оценке защиты отчета студентом приведены в таблице 1 «Шкала оценок уровня».

Контрольные вопросы

1. Что такое IDS и IPS? В чём принципиальная разница между системами обнаружения и предотвращения вторжений?

2. Опишите архитектуру Snort. Каковы функции основных компонентов системы?

3. Какие режимы работы поддерживает Snort? В каких случаях применяется каждый из них?

4. Опишите структуру правила Snort. Какие действия (actions) поддерживаются и в чём их различия?

5. Какие опции правил Snort применяются для проверки содержимого пакетов? Опишите назначение опций content, pcre, byte_test.

6. Что такое препроцессоры в Snort? Какие задачи они выполняют? Приведите примеры препроцессоров.

7. Какие переменные задаются в snort.conf? Каково назначение HOME_NET и EXTERNAL_NET?

8. Чем отличается Snort 3 от предыдущих версий? Какие новые возможности появились?

9. Как интегрируется Snort с SIEM-системами? Какие форматы передачи событий поддерживаются?

10. Какие ограничения имеет сигнатурный анализ? В каких случаях сигнатурный подход неэффективен?

11. Каковы признаки SYN-сканирования в сетевом трафике? Какое правило Snort выявит такую активность?

12. Какие меры следует применять для поддержания актуальности базы правил Snort?

ЛАБОРАТОРНОЕ ЗАДАНИЕ № 7. РАБОТА С ПРЕПРОЦЕССОРАМИ SNORT

Общие сведения

Цель: освоение настройки препроцессоров (preprocessor) Snort

Программное обеспечение: IDS Snort 2.6.1.4, снифферWinDump 3.8.3, сканер NMap 4.20, библиотека WinPcap 4.0, Unicorn, XSpider, hping3.

Методический материал по выполнению лабораторных заданий

Препроцессоры Snort представляют собой модули, выполняющие нормализацию и предварительную обработку сетевого трафика перед передачей в механизм обнаружения. Препроцессоры обеспечивают сборку фрагментированных пакетов, анализ потоков TCP, декодирование протоколов прикладного уровня и выявление аномалий, которые не могут быть обнаружены стандартными сигнатурами. Эффективная настройка препроцессоров является ключевой задачей при построении системы обнаружения вторжений на базе Snort.

Основные препроцессоры Snort включают: frag3 – сборка и анализ фрагментированных IP-пакетов, обеспечивающая защиту от атак, использующих фрагментацию для обхода сигнатур; stream5 – анализ потоков TCP и UDP, обеспечивающий отслеживание состояния соединений и выявление аномалий установления соединений; http_inspect– нормализация и анализ HTTP-трафика, выявление аномалий кодирования URL, методов и заголовков; dns– декодирование DNS-запросов и ответов; dce_rpc– анализ трафика DCE/RPC в средах Microsoft; ssl– анализ установления TLS-сессий и выявление аномалий сертификатов.

Препроцессор frag3 выполняет сборку фрагментированных IP-дейтаграмм и обеспечивает защиту от атак, основанных на особенностях обработки фрагментов различными операционными системами. Препроцессор выявляет наложения фрагментов, аномальные смещения и попытки обхода сигнатур через фрагментацию. Настройка frag3 включает задание размеров буферов сборки, тайм-аутов и политик обработки для различных операционных систем.

Препроцессор stream5 обеспечивает отслеживание состояния TCP-соединений, что позволяет выявлять аномалии установления соединений, попытки обхода межсетевых экранов и аномальные последовательности пакетов. Препроцессор поддерживает оба направления TCP-соединения, выявляет несанкционированные сбросы (RST), повторные передачи и аномалии изменения размера окна. Для UDP stream5 обеспечивает отслеживание псевдо-соединений на основе пар адресов и портов.

Препроцессор http_inspect обеспечивает нормализацию HTTP-трафика и выявление аномалий кодирования, обеспечивая корректную работу сигнатур для веб-атак. Препроцессор выявляет двойное кодирование URL, аномальные методы HTTP, попытки обхода фильтров через нестандартные последовательности символов и другие признаки атак на веб-приложения. Современные версии http_inspect поддерживают анализ HTTP/2 и выявление аномалий конкретных заголовков.

Препроцессоры ssl и dce_rpc обеспечивают анализ протоколов прикладного уровня и выявление аномалий, характерных для атак на соответствующие службы. Препроцессор ssl выявляет аномалии установления TLS-сессий, устаревшие версии протоколов, слабые наборы шифров и аномальные сертификаты. Препроцессор dce_rpc обеспечивает анализ вызовов DCE/RPC в средах Microsoft и выявляет атаки на службы Windows.

Настройка препроцессоров выполняется в файле snort.conf через директивы preprocessor. Каждый препроцессор имеет собственный набор параметров, задающих размеры буферов, тайм-ауты, политики обработки и другие настройки. Корректная настройка препроцессоров существенным

образом влияет на эффективность обнаружения атак и производительность системы. Рекомендуется регулярная проверка и актуализация настроек препроцессоров при изменении сетевой топологии или появлении новых типов атак.

PreprocessorSnort

В дополнение к основным настройкам Snort’а имеются так же плагины, указанные тоже настраиваются в файле snort.conf:

1. Preprocessor: используются для обработки пакетов до срабатывания правил, так как некоторые пакеты необходимо перевести в другой формат для срабатывания правил.

В СОВ Snort присутствует встроенный компонент. Компонент Snort (модуль), называемый «препроцессор» в терминологии разработчиков, описывается в файле конфигурации. С помощью данного компонента можно выявить сканирование портов подконтрольной системы [1].

Алгоритм обнаружения сканирования предполагает следующие:

– требуется отслеживать увеличение числа исходящих TCP-пакетов с установленным флагом RST;

– требуется отследить наличие флага RST на транслируемом ответном пакете. Наличие флага указывает на то, что порт сетевого сервиса, с указанным производилось соединение или попытка соединения, закрыт.

Таким образом, анализируя количество пакетов с установленным флагом RST, можно обнаружить факт сканирования портов системы.

Для подключения и описания настроек управления модуля требуется в конфигурационном файле snort.conf следует внести выделенное ниже:

preprocessorflow: stats_interval 0 hash 2 (в первой строке производятся инициализация препроцессора Flow. Данный объект требуется для функционирования модуля обнаружения);

preprocessorsfportscan: proto{ <протокол> } scan_type (вторая строка инициализирует препроцессор sfPortscan, при данном задаются следующие параметры.

<протокол>– анализируемый протокол (tcp, udp, icmp, ip_proto, all)

<тип_сканирования>– выявляемые типы сканирования (portscan, portsweep, decoy_portscan, distributed_portscan, all) <чувствительность>– чувствительность (low, medium, high) <файл_с_отчетом>– имя файла, в указанный будет помещен отчет об обнаруженных попытках сканирования портов;

{ <тип_сканирования> } sense_level

{ <чувствительность> } logfile { <файл_с_отчетом> }

Файл с отчетом находится в директории после параметра -c при запуске Snort. Чувствительность определяет перечень анализируемой информации и в

итоге сказывается на вероятности «ложной тревоги» (для high данная наибольшая). За более подробной информацией о параметрах модуля sfPortscan следует обращаться к документации на Snort. Приведем пример настройки модуля sfPortscan [1]:

preprocessorflow: stats_interval 0 hash 2 preprocessorsfportscan: proto{ all } scan_type { all } sense_level { medium } logfile { portscan.log }

При указанных настройках можно выявить активную реализацию методов сканирования портов. Необходимо отметить, что две и больше процедуры сканирования портов, выполненные во время одного сеанса работы Snort, будут отражены в файле регистрации событий только один раз. Имеется остается справедливым даже в том случае, когда используются разные методы сканирования. Таким образом, для поверки возможности обнаружения сканирования, выполняемого разными методами, Snort необходимо закрывать и запускать снова.

В Snort имеются следующие preprocessor'ы:

1.Flag2: используется для выявления фрагментированных пакетов на основе протокола IP. Обычно используется для выявления DDOS-атаки (атаки фрагментированными пакетами).

Настройки рассматриваемого препроцессора:

– timeout: время хранения в буфере;

– ttl_limit: ограничение памяти препроцессору до определенного количества байт;

– min_ttl: установка минимального времени жизни пакета для данного принятия;

2. Flag3: настройка нужна для работы с дефрагментацией пакетов IP-протокола, с целенаправленной обработкой фрагментов IP:

– max_frags: максимальное количество активных путей фрагментации (по умолчанию 8192);

– memcap: максимальный объем памяти для данного препроцессора (по умолчанию 4 Мб);

– prealloc_frags: максимальное количество фрагментов, могущие быть обработаны одновременно;

– detect_anomalies: активирует механизм обнаружения аномалий;

– bind_to: IP-адрес привязки «движок».

Методические указания и порядок выполнения работы. Задание

Цель лабораторной работы предполагает освоение принципов работы и настройки препроцессоров Snort, изучение их роли в обеспечении корректного обнаружения атак и приобретение навыков настройки параметров препроцессоров для различных типов трафика.

В рамках лабораторной работы выполняются следующие задачи:

1. Изучение назначения и принципов работы основных препроцессоров Snort (frag3, stream5, http_inspect, dns, ssl).
2. Настройка препроцессора frag3 для сборки фрагментированных пакетов и проверка его работы.
3. Настройка препроцессора stream5 для отслеживания TCP-соединений и анализ его работы при различных типах трафика.
4. Настройка препроцессора http_inspect для нормализации HTTP-трафика и выявление аномалий кодирования.
5. Настройка препроцессора dns для анализа DNS-запросов и выявление аномалий.
6. Настройка препроцессора ssl для анализа установления TLS-сессий и выявление аномалий сертификатов.
7. Анализ влияния настроек препроцессоров на эффективность обнаружения атак и производительность системы.
8. Изучить принципы работы препроцессоров
9. Произветси тестирование работы препроцессоров

Задание 1. Настройка препроцессора frag3

1. Откройте файл конфигурации Snort: sudonano /etc/snort/snort.conf. Найдите раздел настройки препроцессора frag3 (ищите по ключевому слову frag3).

2. Изучите параметры frag3_global: preprocessor frag3_global: max_frags 65536, mem_cap 4194304. Параметр max_frags задаёт максимальное количество одновременно отслеживаемых фрагментированных дейтаграмм, mem_cap– ограничение памяти для хранения фрагментов. Для лабораторного стенда достаточно значений по умолчанию.

3. Изучите параметры frag3_engine: preprocessor frag3_engine: policylinux, detect_anomalies, overlap_limit 10, min_fragment_length 0. Параметр policy задаёт целевую операционную систему для корректной сборки фрагментов, detect_anomalies включает выявление аномалий фрагментации, overlap_limit– максимальное количество наложений фрагментов, min_fragment_length– минимальный размер фрагмента.

4. Включите detect_anomalies и установите policy в соответствии с целевой операционной системой: preprocessor frag3_engine: policylinux, detect_anomalies, overlap_limit 10. Сохраните файл конфигурации.

5. Протестируйте корректность: sudosnort -T -c /etc/snort/snort.conf. Запустите Snort в режиме IDS: sudosnort -A console -q -c /etc/snort/snort.conf -i <интерфейс>.

6. Сгенерируйте фрагментированный трафик с применением hping3: sudo hping3 -1 -f <IP-адрес_цели> -c 10. Опция -f включает фрагментацию ICMP-пакетов. Также выполните ping с большим размером пакета: ping -s 3000 -c 1 <IP-адрес_цели>. Наблюдайте за срабатываниями frag3 в консоли Snort.

Задание 2. Настройка препроцессора stream5

1. В файле конфигурации найдите раздел настройки stream5 (ищите по ключевому слову stream5).

2. Изучите параметры stream5_global: preprocessor stream5_global: track_tcp yes, track_udp yes, track_icmp no, max_tcp 262144, max_udp 131072. Параметры track_tcp/track_udp/track_icmp включают отслеживание соответствующих протоколов, max_tcp/max_udp– максимальное количество отслеживаемых сессий.

3. Изучите параметры stream5_tcp: preprocessor stream5_tcp: policy linux, timeout 180, overlap_limit 10, small_segments 3 bytes 150, detect_anomalies, require_3whs 180. Параметр timeout задаёт тайм-аут неактивности TCP-сессии, small_segments включает выявление аномалий малых сегментов, detect_anomalies включает общее выявление аномалий TCP, require_3whs требует трёх стороннего рукопожатия для отслеживания сессии.

4. Включите detect_anomalies и установите policy в соответствии с целевой операционной системой: preprocessor stream5_tcp: policy linux, detect_anomalies, require_3whs 180. Сохраните файл конфигурации и протестируйте корректность.

5. Сгенерируйте TCP-трафик с аномалиями: выполните SYN-сканирование nmap -sS<IP-адрес_цели>; выполните FIN-сканирование nmap -sF<IP-адрес_цели>; отправьте TCP-пакеты без предшествующего рукопожатия sudo hping3 -A -p 80 <IP-адрес_цели>. Наблюдайте за срабатываниями stream5.

Задание 3. Настройка препроцессора http_inspect

1. В файле конфигурации найдите раздел настройки http_inspect.

2. Изучите параметры global и server: preprocessor http_inspect: global iis_unicode_mapunicode.map 1252; preprocessor http_inspect_server: server default profile all ports {80 8080 8443} no_alerts. Параметр profile задаёт профиль нормализации (all– все доступные проверки), ports– список портов для анализа.

3. Включите detection_anomalies и настройте выявление аномалий кодирования: preprocessor http_inspect_server: server default profile all ports {80 8080 8443} detect_anomalies. Сохраните файл и протестируйте корректность.

4. Сгенерируйте HTTP-запросы с аномалиями кодирования. Создайте тестовый скрипт curl с двойным кодированием URL: curl "http://<IP-адрес>/test%2520page". Отправьте запрос с нестандартными методами: curl -X TRACE http://<IP-адрес>/. Наблюдайте за срабатываниями http_inspect.

Задание 4. Настройка препроцессоров dns и ssl

1. В файле конфигурации найдите разделы настройки dns и ssl.

2. Изучитепараметрыdns: preprocessor dns: ports {53} enable_rdata_overflowenable_obsolete_rdata_typesenable_multiline_responses. Параметр enable_rdata_overflow включает выявление переполнений в DNS-ответах.

3. Изучите параметры ssl: preprocessor ssl: ports {443} trustserversnovalidate_encrypted. Параметр trustservers включает доверие сертификатам серверов, novalidate_encrypted отключает проверку зашифрованных записей.

4. Внесите изменения в настройки, включив выявление аномалий DNS и SSL. Сохраните файл и протестируйте корректность.

5. Сгенерируйте DNS-запросы и TLS-сессии: выполните nslookup с длинным доменным именем; выполните подключение к серверу с устаревшим TLS (openssls_client -connect<IP-адрес>:443 -tls1). Наблюдайте за срабатываниями препроцессоров.

6. Сформируйте итоговый отчёт по результатам выполнения всех заданий. Отчёт должен содержать: описание настройки каждого препроцессора, созданные правила, результаты тестирования, анализ влияния настроек на эффективность обнаружения атак и производительность системы.

Индивидуальное задание

Студент самостоятельно выбирает инфраструктуру организации для рассмотрения при решении лабораторных задач.

Требования к отчету и защите

Защита предполагает опрос по материалу, изложенному в отчете.

Отчет должен быть оформлен: требуется наличие титульного листа, введение (содержит: указание цели лабораторной работы, задачи, краткие теоретические сведения), последовательное изложение разработанного материала, подписанные скриншоты хода работы, если использовалась компьютерная техника, заключение (выводы, анализ полученных результатов).

Проверяется знание теоретического материала с учетом знаний ответов на **Контрольные вопросы**, приведенные в пункте «**Контрольные вопросы**».

Задаются вопросы по пунктам, приведенным в пункте «Методические указания и порядок выполнения работы».

Критерии к оценке защиты отчета студентом приведены в таблице 1 «Шкала оценок уровня».

Контрольные вопросы

1. Что такое препроцессоры в Snort? Какова их роль в общей архитектуре системы обнаружения вторжений?
2. Опишите назначение и принципы работы препроцессора frag3. Какие атаки позволяет выявить данный препроцессор?
3. Опишите назначение и принципы работы препроцессора stream5. Как осуществляется отслеживание состояния TCP-соединений?
4. Опишите назначение и принципы работы препроцессора http_inspect. Какие аномалии HTTP-трафика выявляет данный препроцессор?
5. Опишите назначение и принципы работы препроцессора dns. Какие аномалии DNS-трафика могут быть выявлены?
6. Опишите назначение и принципы работы препроцессора ssl. Какие атаки на TLS-сессии выявляет данный препроцессор?
7. Каковы основные параметры настройки препроцессора frag3? Как выбор параметров влияет на производительность и эффективность обнаружения?
8. Что такое statefulinspection в контексте работы stream5? Чем это отличается от анализа отдельных пакетов?
9. Какие атаки могут быть выявлены только с применением препроцессоров и не выявляются стандартными сигнатурами?
10. Каковы признаки атак, использующих фрагментацию пакетов для обхода сигнатур? Как frag3 обеспечивает защиту от таких атак?
11. Какие параметры настройки stream5 влияют на отслеживание соединений? Каковы риски некорректной настройки?
12. Как интегрируются препроцессоры с механизмом обнаружения Snort? Какие данные передаются от препроцессоров сигнатурам?

ЛАБОРАТОРНАЯ РАБОТА № 8. СИГНАТУРНЫЙ АНАЛИЗ И ОБНАРУЖЕНИЕ АНОМАЛИЙ

Общие сведения

Цель: получитьнавык использования сигнатурного анализа и обнаружения сетевых угроз средствами обнаружения вторжений.

Программное обеспечение: IDS Snort 2.6.1.4, снифферWinDump 3.8.3, сканер NMap 4.20, библиотека WinPcap 4.0, Unicorn, XSpider, hping3.

Методический материал по выполнению лабораторных заданий

Сигнатурный анализ представляет собой метод выявления атак и подозрительной активности, основанный на сопоставлении сетевого трафика или системных событий с заранее известными сигнатурами атак. Сигнатура – это формализованное описание признаков конкретной атаки, позволяющее однозначно идентифицировать её в потоке данных. Системы, реализующие сигнатурный анализ (Snort, Suricata, SymantecEndpointProtection, KasperskyEndpointSecurity), обеспечивают высокую точность выявления известных атак при минимальном уровне ложных срабатываний.

Преимущества сигнатурного анализа включают: высокую точность выявления известных атак; низкий уровень ложных срабатываний; предсказуемость работы и возможность настройки под конкретные требования; относительно невысокие требования к вычислительным ресурсам. Недостатки сигнатурного анализа: невозможность выявления неизвестных атак (zero-day); необходимость постоянного обновления баз сигнатур; ограниченная эффективность против атак, использующих полиморфизм и метаморфизм кода.

Обнаружение аномалий представляет собой метод выявления атак, основанный на отклонении наблюдаемого поведения системы от предварительно построенной модели нормального поведения. Модель нормального поведения формируется в процессе обучения на основе анализа трафика или системных событий в течение определённого периода. Системы обнаружения аномалий выявляют отклонения от модели, которые могут свидетельствовать о возникновении атаки, не предусмотренной сигнатурной базой.

Методы обнаружения аномалий классифицируются по подходу к построению модели нормального поведения: статистические методы – основаны на анализе статистических характеристик трафика (среднее значение, дисперсия, корреляция); методы машинного обучения – применяют алгоритмы кластеризации, классификации и регрессии для выявления аномалий; методы на основе искусственных нейронных сетей – обеспечивают выявление сложных нелинейных зависимостей в поведении системы; методы на основе анализа последовательностей – выявляют аномалии в порядке следования событий.

Современные системы обнаружения вторжений, такие как Suricata, Zeek (ранее Bro), ExtraHop, Darktrace, реализуют гибридный подход, сочетающий сигнатурный анализ с методами обнаружения аномалий. Гибридный подход обеспечивает выявление как известных атак с применением сигнатур, так и подозрительной активности, не предусмотренной сигнатурной базой. Методы машинного обучения применяются для анализа больших объёмов данных (bigdatasecurityanalytics) и выявления сложных целенаправленных атак (APT).

С позиции информационной безопасности сигнатурный анализ и обнаружение аномалий применяются в составе комплексных систем защиты: сетевых IDS/IPS (Snort, Suricata, CiscoFirepower); хостовых IDS (OSSEC, Wazuh); систем анализа логов и SIEM (Splunk, ElasticStack, QRadar); систем анализа сетевого трафика (NTA/NDR). Корреляция событий из различных источников обеспечивает повышение точности выявления атак и снижение уровня ложных срабатываний.

Правила Snort для выявления сетевых атак формируются на основе известных сигнатур. Например, для выявления SYN-сканирования применяется правило, проверяющее установку флага SYN при отсутствии предшествующего пакета установления соединения. Для выявления попыток подбора паролей по SSH применяется правило, подсчитывающее количество неудачных попыток аутентификации за определённый интервал времени. Для выявления атак на веб-приложения применяются правила, проверяющие наличие характерных последовательностей символов в HTTP-запросах (например, SQL-инъекций или XSS).

Сигнатурный анализ и обнаружение аномалий

Методы сигнатурного анализа трафика являются основными в СОВ. Так же используются методы обнаружения аномалий.

Выявление сетевой атаки предполагает перехват трафика и поиск в пакетах данных последовательностей данных, эталонных параметров и целых строк наборов событий, указанных в шаблонах или строки символов в упорядоченном во времени потоке информации.

Механизм поиска определяется способом описания атаки. Описывается атака с помощью правила и набора правил (условий).

Все СОВ, использующие метод обнаружения атак по сигнатуре, содержат базу структур (обычно правил), описывающих параметры известных атак (сигнатуры). При отсутствии описания (набора параметров) атаки, система обнаружения вторжений не обнаружит вредоносное воздействие, поскольку операция сопоставления полученного трафика и эталонного (присевающего в базе) не покажет совпадение. Соответственно, требуется постоянное обновление базы сигнатур (возможно с модификацией последних). Однако, при таком подходе к обнаружению вторжений исключаются ложные подтверждения атак, т. е. система однозначно интерпретирует входящий трафик, что повышает точность системы.

Для преодоления недостатков сигнатурного подхода в СОВ используется методика выявления вторжений на основе методики выявления аномалий, суть указанного сводится к определению нелегитимного трафика или состояния системы при несовпадении данного с разрешённым (штатным). Для того чтобы реализовать подобную систему обнаружения вторжений требуется:

– определить способы описания, параметризации системы и трафика (в данной нормальном состоянии);

– поскольку трафик меняется, требуется определить шаблон описания состояния системы и/или параметризованного трафика таким образом, чтобы были учтены границы допустимого и недопустимого отклонения от заданного эталона легитимного состояния системы или трафика;

– определить способы сравнения эталонных значений и получаемых с учетом допустимы и недопустимых отклонений.

Рассмотрим ситуацию сетевой разведки. Предполагается что при исследовании внешним нарушителем системы количество TCP-сеансов резко возрастет, как правило, отклоняемых будет боле, чем при штатной ситуации обмена данными в сети. Для распознавания подобной атаки можно применить следующее:

– статистический критерий равенства средних значений двух случайных величин (для данного использования, правда, необходимо сделать два достаточно неоднозначных предположения: о нормальности распределений случайных величин и о равенстве указанных дисперсий) [7].

– статистические методы и «методы обнаружения разладки». Требуются для коррекции вероятности ошибочных подтверждений и легитимном изменении трафика (в том числе сеансов TCP.)

– математические методы распознавания образов. Основываются на обучении системы для выявления свойств трафика, динамических параметров данного легитимного изменения.

Методы анализа аномального состояния трафика или состояния системы позволяет обнаруживать новые, неописанные сигнатурно атаки. Основный принцип данного метода следующий: все, что отличается от параметров штатного режима работы системы или стандартного легитимного трафика считается атакой.

Возможны два класса ошибок при использовании метода анализа аномального состояния:

– ложное подтверждение легитимности трафика;

– ложная срабатывание.

СОВ также могут классифицироваться по режиму обработки данных трафика:

– системы реального времени (входящий поток данных от сетевых датчиков анализируется сразу, без предварительной записи в логи и т. д.);

– системы отложенной обработки (выявления атак производится при анализе содержимого журналов регистрации событий, логов и/или банк записанного трафика).

Главная задача СОВ реального времени – обнаружение вторжений в момент указанных активности, при данном анализируются только заголовки пакетов. Обычно фильтрация пакетов данных подобных СОВ производится сетевом (протокол IP) и транспортном уровнях (протоколы TCP, UDP) OSI (таблица 2) [1].

Таблица 2 – Список параметров СОВ Snort

Параметр	**Описание**
-c <rules>	Использовать файл правил <rules>.
-E	Вносить предупреждения (alerts) в журнал регистрации событий Windows NT (не создавая log-файла)
-h <hn>	Указать внутреннюю (домашнюю сеть (homenetwork) <hn>)
-i <if>	Указать номер сетевого интерфейса по данного номер <if> (список интерфейсов можно получить при помощи команды snort -W)
-I	Добавлять к предупреждению наименование интерфейса
-K <mode>	Указание формата регистрации предупреждений: pcap (по умолчанию) – в двоичном формате, ascii– в текстовом формате, none– без регистрации
-l <ld>	Сохранять результаты регистрации в каталог <ld>
-L <file>	Сохранять результаты регистрации в указанный файл формата tcpdump (будет располагаться в каталоге, предварительно указанном параметром -l)
-n <cnt>	Завершить работу программы после получения <cnt> пакетов
-N	Не сохранять в файлах регистрации содержимого пакетов (сохраняется лишь текст предупреждений)
-p	Анализировать только пакеты, отправленные на локальный адрес, и широковещательные пакеты
-r <tf>	Прочитать и обработать файл <tf>, записанный в формате tcpdump

Параметр	Описание
-S <n=v>	Установить значение переменной n файла правил, равной v
-U	Записывать временные метки в универсальном скоординированном времени (UTC)
-v	Отображать на экране заголовки всех перехваченных пакетов
-V	Вывести версию СОВ Snort
-W	Показать доступные сетевые интерфейсы
-X	Сохранять в файле журнала регистрации событий содержимое перехваченных пакетов в не отформатированном виде, начиная с уровня link модели OSI
Параметр	Описание
-y	Добавить месяц, день и год в отображаемые и сохраняемые временные отметки
-?	Справка

Способы описания правил СОВ Snort.

Описание языка правил приведено в документации к системе, в файле snort_manual.pdf (c:\snort\doc\snort_manual.pdf).

Правила записываются в одну строку, если возникает необходимость перенести текст правила на следующую строку, необходимо добавить в конце строки символ обратной косой черты «\».

Синтаксис правила подразумевает наличие двух частей:

1) Атрибуты. Описание любого атрибута указывается в скобках, каждая пара атрибут – значение имеет вид: <атрибут>: <значение>;. Значения строковых атрибутов записываются в кавычках.

Рассмотрим пример простого правила (одна строка) [17]:

– строка правила следующая: alert<протокол><адрес_подсети1>[/маска_подсети1] <порт1><направление><адрес_подсети2>[/маска_подсети2]

<порт2> ([msg:"Текст сообщения";] [другие_атрибуты]), где:

− alert – действие, указанное необходимо выполнить при обнаружении пакета, удовлетворяющего данному правилу, и указанное заключается в генерации «предупреждения» – записи в журнале регистрации

−<протокол> – наименование протокола (tcp, udp, icmp, ip);

−<адрес_подсети>[/маска_подсети] – IP-адрес и маска подсети;

− <порт> – номер порта либо диапазон портов в формате 1:1024 для обозначения диапазона портов от 1 до 1024, 1024: – с номерами больше или равными 1024, или:1024 – меньше или равными 1024;

− <направление> – обозначение направления в виде ->, <- или <>.

2) Заголовок. Заголовок состоит из:

1. Указания действия, указанное необходимо выполнить (alert (генерация записи предупреждающего события, запись содержимого пакета данных помеченного как нелегитимный), log (простая запись содержимого пакета), pass (пропуск о записи) и др.) (таблица 3);

2. протокол (tcp, udp, icmp, ip);

3. IP-адрес, маска подсети источника и приемника информации, а также информации о портах источника и приемника.

В случае неизвестности точных значений параметров IP-адреса и номера портов могут подменяться ключевым словом «any», что указывает на любые значения в задаваемом параметре.

Существует четыре категории атрибутов:

−meta-data– предоставляют информацию о правиле;

−payload– атрибуты данного типа предназначены для поиска информации в «полезной нагрузке» (содержимом) пакета;

−non-payload– предназначены для поиска информации в заголовках пакетов;

−post-detection– определяют поведение системы после обнаружения пакета, удовлетворяющего правилу.

В таблице приведен неполный список атрибутов, могущие быть использованы при написании правил [14].

Для ограничения области поиска используют ключевые слова offset и depth, так как указанное существенно сократит время, затрачиваемое на анализ пакета.

Таблица 3 – Список атрибутов СОВ Snort

Атрибут	**Описание**
meta-data	
msg: "<текст>";	Сообщение, указанное добавляется в журнал регистрации при активации правила

Атрибут	Описание
sid: <идентификатор>;	Уникальный номер, используемый для идентификации правил. Идентификаторы от 100 до 1 000 000 используются для правил, включенных в дистрибутив Snort. Для локальных правил следует использовать значения больше 1 000 000
rev: <номер_редакции>;	Целое число, служащее для обозначения номера редакции правила
classtype: <имя_класса>;	Используется для обозначения класса атаки. Полный список классов приведен в документации по Snort
priority: <приоритет>;	Целое число, используемое для переопределения приоритета, задаваемого указанным ранее классом атаки, или для назначения приоритета новому правилу. Наивысший приоритет – 1, типичное значение атрибута составляет от 1 до 4
payload	
content: [!]"<строка>";	Позволяет искать заданную подстроку в содержимом полезной нагрузки пакета. Восклицательный знак означает отсутствие указанной информации в пакете. По умолчанию данный атрибут является чувствительным к регистру. Для обозначения двоичных данных следует использовать шестнадцатеричные значения, отделенные вертикальными чертами: |00 5C|. Атрибут content имеет несколько модификаторов, могущие располагаться следом за указанным элементом
nocase;	Модифицирует стоящий ранее атрибут content, делая данного нечувствительным к регистру
depth: <число_байт>;	Значение атрибута (в байтах) определяет, как далеко в полезной нагрузке пакета должен производиться поиск

Атрибут	Описание
offset: <число_байт>;	Значение атрибута определяет, сколько байтов полезной нагрузки следует пропустить. Поиск будет вестись, начиная с число_байт+1-го байта
distance: <число_байт>;	Атрибут похож на depth, но указывает, сколько байт необходимо пропустить после предыдущей совпавшей подстроки перед продолжением поиска
within: <число_байт>;	Атрибут указывает системе искать совпадения лишь в первых число_байт, начиная с конца предыдущей совпавшей подстроки
non-payload	
dsize: <размер>;	Сравнить размер полезной нагрузки с заданным. Возможно указание диапазонов значений с использованием знаков >, < и <>. Например: >128, 300<>500 (от 300 до 500)
flags: [!|*|+]<флаги>	Проверить, установлены ли указанные флаги в принятом TCP-пакете. Флаги записываются подряд без пробелов и обозначаются следующим образом: F– FIN (LSB в байте флагов) S– SYN R– RST P– PSH A– ACK U– URG 1 – Резерв 1 (MSB в байте флагов) 2 – Резерв 2 0– флаги не установлены Могут быть дополнительно использованы следующие модификаторы: ! – указанные флаги не установлены * – установлен хотя бы один из указанных + – установлены указанные и любые другие
itype: <тип>;	Сравнить тип ICMP сообщения с указанным. Возможно указание диапазонов значений с использованием знаков >, < и <> (см. Выше)
icode: <код>;	Сравнить код ICMP сообщения с указанным

Комментарии указываются знаком # (все, что после указанного символа, не читается системой до конца строки).

Синтаксис языка описания сигнатурных параметров позволяет вместо точных IP-адресов использовать специальные переменные переменные, указанные можно применять в последующих описаниях. Синтактис описания переменной следующий:

var<имя_переменной><значение_переменной>

Для того, что указать переменную, используется знак $.

Методические указания и порядок выполнения работы. Задание

В рамках лабораторной работы выполняются следующие задачи:

Изучение принципов сигнатурного анализа и методов построения сигнатур атак.

Создание правил Snort для выявления различных типов сканирования портов (SYN, FIN, Xmas, Null).

Создание правил Snort для выявления атак на сетевые службы (SSH-брутфорс, FTP-брутфорс).

Создание правил Snort для выявления атак на веб-приложения (SQL-инъекции, XSS, pathtraversal).

Изучение принципов обнаружения аномалий и методов построения моделей нормального поведения.

Анализ возможностей гибридного подхода, сочетающего сигнатурный анализ с обнаружением аномалий.

Сравнительный анализ эффективности сигнатурного анализа и обнаружения аномалий при выявлении различных типов атак.

1.Изучить принципы сигнатурного анализа

2. Произвести сигнатурный анализ

Задание 1. Создание правил для выявления сканирования портов

1. Откройте файл локальных правил Snort: sudonano /etc/snort/rules/local.rules.

2. Создайте правило для выявления SYN-сканирования. SYN-сканирование характеризуется отправкой большого количества SYN-пакетов без завершения трёхстороннего рукопожатия: alerttcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SYN scanattempt"; flags:S,12; detection_filter:trackby_src, count 10, seconds 5; sid:2000001; rev:1;). Опция flags:S,12 проверяет установку флага SYN и отсутствие флагов ACK и RST. Опция detection_filter генерирует предупреждение при получении 10 SYN-пакетов от одного источника за 5 секунд.

3. Создайте правило для выявления FIN-сканирования. FIN-сканирование использует пакеты с флагом FIN для определения состояния портов: alerttcp $EXTERNAL_NET any -> $HOME_NET any (msg:"FIN scanattempt"; flags:F,12; detection_filter:trackby_src, count 10, seconds 5; sid:2000002; rev:1;).

4. Создайте правило для выявления Xmas-сканирования. Xmas-сканированиеустанавливаетфлаги FIN, PSH и URG одновременно: alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"Xmas scan attempt"; flags:FPU,12; detection_filter:trackby_src, count 10, seconds 5; sid:2000003; rev:1;).

5. Создайте правило для выявления Null-сканирования. Null-сканированиеотправляетпакетыбезустановленныхфлагов: alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"Null scan attempt"; flags:0; detection_filter:trackby_src, count 10, seconds 5; sid:2000004; rev:1;).

6. Протестируйте корректность конфигурации: sudo snort -T -c /etc/snort/snort.conf. Запустите Snort врежиме IDS: sudo snort -A console -q -c /etc/snort/snort.conf -i<интерфейс>.

7. Выполните сканирование портов с применением nmapдлякаждогометода: sudonmap -sS<IP-адрес_цели>; sudonmap -sF<IP-адрес_цели>; sudonmap -sX<IP-адрес_цели>; sudonmap -sN<IP-адрес_цели>. Зафиксируйте срабатывания правил в консоли Snort. Сопоставьте количество срабатываний с количеством отправленных пакетов.

Задание 2. Создание правил для выявления атак на сетевые службы

1. Создайте правило для выявления SSH-брутфорс. Атака характеризуется множественными попытками аутентификации за короткий период: alert tcp $EXTERNAL_NET any -> $HOME_NET 22 (msg:"SSH brute force attempt"; flow:to_server,established; content:"SSH-"; nocase; detection_filter:trackby_src, count 5, seconds 60; sid:2000005; rev:1;). Опция flow:to_server,established ограничивает правило пакетами в направлении сервера в установленных соединениях. Опция content:"SSH-" проверяет наличие баннера SSH в полезной нагрузке (Таблица 3).

2. Создайтеправилодлявыявления FTP-брутфорс: alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP brute force attempt"; flow:to_server,established; content:"USER"; nocase; detection_filter:trackby_src, count 5, seconds 60; sid:2000006; rev:1;).

3. Создайте правило для выявления попыток подбора пароля RDP (Event ID 4625 в Windows): alert tcp $EXTERNAL_NET any -> $HOME_NET 3389 (msg:"RDP brute force attempt"; flow:to_server; detection_filter:trackby_src, count 10, seconds 60; sid:2000007; rev:1;).

4. Запустите Snort и сгенерируйте брутфорс-атаки с применением hydra: hydra -l admin -P /usr/share/wordlists/rockyou.txt ssh://<IP-адрес_цели>; hydra -l

admin -P /usr/share/wordlists/rockyou.txt ftp://<IP-адрес_цели>. Зафиксируйте срабатывания правил.

Задание 3. Создание правил для выявления атак на веб-приложения

1. Создайте правило для выявления SQL-инъекций. SQL-инъекциихарактеризуютсяналичием SQL-ключевыхсловв HTTP-запросах: alert tcp any any -> $HOME_NET 80 (msg:"SQL Injection attempt - UNION SELECT"; flow:to_server,established; content:"UNION"; nocase; content:"SELECT"; nocase; distance:0; within:20; sid:2000008; rev:1;). Опции distance и within задают относительное положение второго content относительно первого (Таблица 1).

2. Создайте правило для выявления XSS. XSS-атаки характеризуются наличием JavaScript-кодав HTTP-запросах: alert tcp any any -> $HOME_NET 80 (msg:"XSS attempt - script tag"; flow:to_server,established; content:"<script"; nocase; sid:2000009; rev:1;).

3. Создайте правило для выявления PathTraversal. Path Traversal характеризуется наличием последовательности ../ в HTTP-запросах: alert tcp any any -> $HOME_NET 80 (msg:"Path Traversal attempt"; flow:to_server,established; content:"../"; sid:2000010; rev:1;).

4. Создайте правило для выявления Command Injection: alert tcp any any -> $HOME_NET 80 (msg:"Command Injection attempt"; flow:to_server,established; pcre:"/(\|\|\|\|\&\&|;|\|\n|\$\(|\`)/Ui"; sid:2000011; rev:1;). Опция pcre использует регулярное выражение для выявления метасимволов командной оболочки.

5. Запустите Snort и сгенерируйте атаки на тестовое веб-приложение (DVWA или аналогичное): отправьте SQL-инъекцию через форму входа; отправьте XSS через комментарий; отправьте PathTraversal через параметр URL. Зафиксируйте срабатывания правил.

Задание 4. Сравнение сигнатурного анализа и обнаружения аномалий

1. Изучите принципы обнаружения аномалий. УстановитеZeek (ранееBro): sudoaptinstallzeek. НастройтеZeekдляанализасетевоготрафика: sudozeekctldeploy.

2. Запустите Zeek для мониторинга трафика: sudozeekctlstart. Сгенерируйте нормальный трафик (веб-браузение, SSH, DNS) и зафиксируйте метаданные, формируемые Zeek в файлах conn.log, http.log, dns.log.

3. Сгенерируйте аномальный трафик (сканирование, брутфорс, веб-атаки) и сравните возможности сигнатурного анализа (Snort) с метаданными Zeek. Определите, какие атаки выявляются каждым методом.

4. Сравните эффективность методов: сигнатурный анализ (Snort) – высокая точность выявления известных атак, невозможность выявления неизвестных; обнаружение аномалий (Zeek) – возможность выявления неизвестных атак, высокий уровень ложных срабатываний.

5. Сформируйте итоговый отчёт по результатам выполнения всех заданий. Отчёт должен содержать: созданные правила с пояснениями, результаты тестирования, сравнительный анализ методов обнаружения, выводы по эффективности применения.

Таблица 4 – Правила СОВ

№ п/п	Описание	Правило
1	Обнаружение входящих ECHO-запросов (ping’ов)	alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg: "Incoming ECHO REQUEST"; itype: 8;)
2	Обнаружение исходящих ECHO-ответов	alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg: "Outgoing ECHO REPLY"; itype: 0;)
3	Обнаружение больших ICMP-пакетов (атака «PingofDeath»)	alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg: "Incoming large ICMP packet"; dsize: >800;)
4	DoS-атака Winnuke	alert tcp $EXTERNAL_NET any -> $HOME_NET 135:139 (msg: "DoS Winnuke attack"; flags: U+;)
5	Запрос на подключение к 139 порту (служба SMB) из внешней сети (два варианта)	alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg: "NETBIOS SMB IPC$ share access"; flags: A+; content: "|00|"; offset: 0; depth: 1; content: "|FF|SMB|75|"; offset: 4; depth: 5; content: "\\IPC$|00|"; nocase;) alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS SMB IPC$ share access (unicode)"; flags: A+; content: "|00|"; offset: 0; depth: 1; content: "|FF|SMB|75|"; offset: 4; depth: 5; content: "|5c00|I|00|P|00|C|00|$| 00|"; nocase;)

№ п/п	Описание	Правило
6	Запрос на подключение к 445 порту (служба SMB) из внешней сети (два варианта)	alert tcp $EXTERNAL_NET any -> $HOME_NET 445 (msg: "NETBIOS SMB IPC$ share access"; flags: A+; content: "|00|"; offset: 0; depth: 1; content: "|FF|SMB|75|"; offset: 4; depth: 5; content: "\\IPC$|00|"; nocase;) alert tcp $EXTERNAL_NET any -> $HOME_NET 445 (msg:"NETBIOS SMB IPC$ share access (unicode)"; flags: A+; content: "|00|"; offset: 0; depth: 1; content: "|FF|SMB|75|"; offset: 4; depth: 5; content: "|5c00|I|00|P|00|C|00|$|00|"; nocase;)
7	Обнаружение сканирования портов методом NULL.	alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"NULL port scanning"; flags:!FSRPAU;)
8	Обнаружение сканирования портов методом XMAS.	alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"XMAS port scanning"; flags:FPU+;)

Индивидуальное задание

Студент самостоятельно выбирает инфраструктуру организации для рассмотрения при решении лабораторных задач.

Требования к отчету и защите

Защита предполагает опрос по материалу, изложенному в отчете.

Отчет должен быть оформлен: требуется наличие титульного листа, введение (содержит: указание цели лабораторной работы, задачи, краткие теоретические сведения), последовательное изложение разработанного материала, подписанные скриншоты хода работы, если использовалась компьютерная техника, заключение (выводы, анализ полученных результатов).

Проверяется знание теоретического материала с учетом знаний ответов на **Контрольные вопросы**, приведенные в пункте «**Контрольные вопросы**».

Задаются вопросы по пунктам, приведенным в пункте «Методические указания и порядок выполнения работы».

Критерии к оценке защиты отчета студентом приведены в таблице 1 «Шкала оценок уровня».

Контрольные вопросы

1. Что такое сигнатурный анализ? Каковы его преимущества и недостатки?
2. Что такое обнаружение аномалий? Каковы его преимущества и недостатки по сравнению с сигнатурным анализом?
3. Опишите принципы построения сигнатур атак. Какие требования предъявляются к качеству сигнатур?
4. Какие методы обнаружения аномалий существуют? Опишите их особенности и область применения.
5. Что такое гибридный подход к обнаружению атак? В чём его преимущества перед отдельными методами?
6. Каковы признаки SYN-сканирования в сетевом трафике? Какое правило Snort выявляет такую активность?
7. Каковы признаки SSH-брутфорс-атаки? Какие методы выявления применяются?
8. Что такое SQL-инъекция? Какие признаки SQL-инъекции могут быть выявлены в HTTP-запросах?
9. Чтотакое XSS (Cross-Site Scripting)? Какие признаки XSS выявляются в HTTP-запросах?
10. Какие ограничения имеет сигнатурный анализ при выявлении атак zero-day? Как обнаружение аномалий решает эту проблему?
11. Какие системы обнаружения вторжений реализуют гибридный подход? Опишите их возможности.
12. Как интегрируются системы обнаружения вторжений с SIEM-системами? Какие преимущества обеспечивает корреляция событий?

ЛАБОРАТОРНАЯ РАБОТА № 9. АУДИТ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ ВЕБ-ПРИЛОЖЕНИЙ

Общие сведения

Цель: получение навыков обследования организации при проведении аудита информационной безопасности веб-приложений, обучение методам и средствам сбора информации об анализируемом веб-приложении, обучение методам и средствам тестирования защищенности служб SSL/TLS, тестирования защищенности механизма управления доступом в веб-приложениях

Программное обеспечение: среда виртуализации, дистрибутивы BacktrackLinux или KaliLinux, среда выявления и эксплуатации уязвимостей MetasploitFramework, среда тестирования защищенности веб-приложений BurpSuite, среда эксплуатации уязвимостей веб-приложений BeEF,

небезопасные веб-приложения проекта WebGoat, образы небезопасных веб-приложений проекта PentesterLab, современные веб-браузеры (InternetExplorer, GoogleChrome, MozillaFirefox), ПО Wireshark, программные средства инструментального анализа защищенности (XSpider, MaxPatrol ЗАО Позитив Текнолоджис или аналогичных программ), а также специализированные сканеры уязвимостей веб-приложений (например, Acunetix, AppScan, BurpSuitePro, W3AF)

Методический материал по выполнению лабораторных заданий

Аудит безопасности веб-приложений представляет собой комплекс мероприятий, направленных на выявление уязвимостей в веб-приложениях и оценку их защищённости от актуальных угроз. Аудит включает анализ архитектуры приложения, проверку реализации механизмов аутентификации и авторизации, выявление уязвимостей ввода (инъекции), проверку корректности обработки сессий, анализ клиентской части (JavaScript) и конфигурации веб-сервера. Современные методики аудита опираются на стандарты OWASP TestingGuide и OWASP Top 10.

OWASP Top 10 - перечень наиболее критичных рисков безопасности веб-приложений, обновляемый OpenWebApplicationSecurityProject (OWASP) каждые три-четыре года. Актуальный перечень OWASP Top 10 2021 включает: A01 BrokenAccessControl (нарушение контроля доступа); A02 CryptographicFailures (ошибки криптографии); A03 Injection (инъекции); A04 InsecureDesign (небезопасный дизайн); A05 SecurityMisconfiguration (небезопасная конфигурация); A06 VulnerableandOutdatedComponents (устаревшие компоненты); A07 IdentificationandAuthenticationFailures (ошибки идентификации и аутентификации); A08 SoftwareandDataIntegrityFailures (ошибки целостности); A09 SecurityLoggingandMonitoringFailures (ошибки журналирования); A10 Server-SideRequestForgery (SSRF).

Современные инструменты аудита веб-приложений включают: BurpSuiteProfessional– комплексную платформу для тестирования безопасности веб-приложений, обеспечивающую проксирование трафика, активное и пассивное сканирование, ручной анализ запросов с применением Intruder и Repeater; OWASP ZAP (ZedAttackProxy) – open-source инструмент с возможностями, аналогичными BurpSuite; Acunetix и Netsparker – коммерческие сканеры, обеспечивающие автоматизированное выявление уязвимостей; sqlmap – специализированный инструмент для выявления и эксплуатации SQL-инъекций; Nikto – сканер веб-серверов, выявляющий устаревшие версии и небезопасные конфигурации.

Методики аудита веб-приложений включают: пассивное сканирование – анализ трафика приложение-браузер без отправки тестовых запросов; активное

сканирование – отправка тестовых запросов для выявления уязвимостей; ручное тестирование – детальный анализ конкретных функций приложения с применением специализированных техник; анализ исходного кода (SAST) – выявление уязвимостей на уровне исходного кода приложения; анализ компонентов (SCA) – проверка используемых библиотек и фреймворков на наличие известных уязвимостей.

Тестирование на проникновение веб-приложений проводится в следующей последовательности: сбор информации о приложении (картирование структуры, выявление технологий); анализ механизмов аутентификации и авторизации; проверка наличия уязвимостей ввода (SQL-инъекции, XSS, commandinjection); анализ управления сессиями (предсказуемость идентификаторов, корректность timeout); проверка бизнес-логики (доступ к данным других пользователей, обход проверок); анализ конфигурации веб-сервера и применяемых заголовков безопасности (CSP, HSTS, X-Frame-Options).

С позиции информационной безопасности регулярный аудит веб-приложений является обязательным элементом программы защиты информации, требуемым стандартами PCI DSS (для приложений, обрабатывающих данные платёжных карт), ФСТЭК (для значимых объектов КИИ) и ISO 27001. Выявленные в процессе аудита уязвимости классифицируются по критичности (CVSS), и формируется план корректирующих мероприятий с указанием сроков устранения. Особое внимание следует уделять интеграции проверок безопасности в процесс CI/CD для выявления уязвимостей на ранних этапах разработки (DevSecOps).

Одним из первых этапов анализа защищенности любой компьютерной системы является сбор информации. В зависимости от используемой методологии анализа защищенности вебприложения могут применяться различные методы и средства сбора информации. Стоит отметить, что сбор информации, как правило, не характерен для методологии инструментального анализа защищенности (сканирования), а характерен для методологии тестирования на возможность проникновения.

Методы сбора информации делятся на активные и пассивные. Активные методы требуют непосредственного взаимодействия с исследуемым приложением путем отправки ему запросов и анализа соответствующих ответов, а пассивные методы используют информацию, отправляемую сервером веб-приложения данного клиентам (например, HTTP-заголовки X-Frame-Options, Strict-TransportSecurity и т. д.) без отправки запросов. При анализе вебприложений, как правило, используются только активные методы.

Активные методы делятся на методы с подключением к приложению (например, идентификация веб-сервера с помощью сканера Httprint) и методы

без подключения (например, сбор информации о приложении поисковыми роботами, сканерами Интернет, и т.д.).

В результате проведения сбора информации о веб-приложении могут быть получены:

– имена и IP-адреса сетевых узлов, на указанных размещены вебприложение и данного компоненты;

– логины и пароли технологических учетных записей;

– комментарии разработчиков;

– данные о системном и прикладном ПО, применяемых средствах защиты и конфигурации веб-приложения;

– адреса электронной почты разработчиков приложения;

– исходный код серверной части веб-приложения;

– конфиденциальные файлы.

– Программными средствами получения необходимой информации являются:

– поисковые системы (например, Google, Shodan, Bing);

– специализированные сканеры уязвимостей Интернет (например, http://un1c0rn.net/);

– инструментальные средства анализа защищенности сетей общего назначения (Nmap, Xprobe2, XSpider);

– инструментальные средства анализа защищенности сетей вебприложений (AppScan, Acunetix, BurpSuite, ZAP, W3AF и т. д.).

Защита транспортного уровня веб-приложения основана на использовании протоколов семейства SSL/TLS, имеющих значительное количество механизмов, функций и параметров защиты, реализация и конфигурация указанных определяет в конечном итоге уровень защищенности веб-приложения. Несмотря на то, что в настоящее время известно много автоматизированных средств тестирования защищенности SSL/TLS (например, сервис www.ssllabs.com, программы SSLscan и SSLyze), детали указанных реализации, как правило, неизвестны или требуют дополнительного исследования, что иногда не позволяет полностью доверять результатам указанных работы. Одним из низкоуровневых и надежных средств тестирования защищенности служб SSL/TLS является клиент OpenSSL.

Одним из основных механизмов защиты современных вебприложений является механизм управления доступом. Обычно выделяют следующие этапы управления доступом [9]: − идентификация – установление идентификационных данных;

− аутентификация – подтвержденное установление идентификационных данных;

− авторизация – назначение прав идентификационным данным. При входе в веб-приложение (signin, login) пользователь идентифицируется (сообщает соответствующ идентификатор) и аутентифицируется (доказывает, что данный объект именно указанный пользователь, чей идентификатор был сообщен).

Большинство веб-приложений используют аутентификацию по паролю. В веб-приложениях с высоким уровнем защищенности (например, в Интернет-банках) также применяются протоколы двухфакторной аутентификации. Очевидным недостатком аутентификации по паролю является возможность использования паролей с плохими статистическими характеристиками. Хранение пароля или данного передача по каналам связи в открытом или даже зашифрованном виде потенциально несет угрозу раскрытия пароля.

Тем не менее, современные защищенные веб-приложения в большинстве случаев используют передачу пароля в зашифрованном виде с помощью протоколов семейства SSL/TLS, а хранение пароля в хешированном виде. При данном для хранения паролей пользователей рекомендуется использовать не криптографические хэш-функции общего назначения (например, SHA или MD5), а специализированные функции PBKDF2, bcrypt, scrypt и т. п. Также для хранения паролей необходимо использовать «соль», предназначенную для затруднения проведения атак по словарям и радужным таблицам.

Авторизация в веб-приложениях может быть определена как процесс проверки того, разрешен или запрещен запрос на получения доступа пользователя к ресурсу в соответствии с заданной политикой безопасности. Как правило, в веб-приложениях реализуется ролевая (RBAC) или атрибутная (ABAC) политика логического управления доступом.

Одним из методов тестирования возможности получения привилегий другого пользователя является дифференциальный анализ. Данного идея заключается в идентификации всех возможных запросов и соответствующих данным URL, указанные может выполнить данный пользователь. Затем все полученные запросы выполняются от имени другого пользователя веб-приложения.

Механизм авторизации рекомендуется реализовывать на уровнях представления, бизнес-логики и данных веб-приложения. Уровень представления – не отображает функционал (например, формы, фреймы, ссылки, кнопки), на указанный пользователь не имеет прав доступа. Уровень бизнес-логики обеспечивает выполнение проверки наличия соответствующих прав доступа до выполнения запроса в веб-приложении, т. е. никакие функции не могут быть выполнены до авторизации (например, если пользователь отправляет запрос на удаление учетной записи, то веб-приложение должно убедиться, что пользователь имеет право на удаление учетной записи и не

выполнять никаких функций до того, как будет установлено). Уровень данных обеспечивает проверку наличия прав доступа пользователя к данным, а не только к функционалу обработки данных (например, пользователь, используя URL вида /delete?record=1, должен удалять только указанные записи в базе данных, на указанные данный объект имеет право доступа DELETE).

Традиционно DoS-атаки являются сетевыми (используют недостатки сетевых технологий) и могут быть классифицированы по уровням модели ISO/OSI. Например, атаки ICMP Flood (L3) и DNS/NTP Amplification (L7) приводят к отказу канала связи, а атаки Ping of Death (L3), SYN Flood (L4), SSL RenegotiationDoS (L5/L6), HTTP Flood (L7), Slow HTTP (L7) воздействуют на платформу веб-приложения (операционная система, веб-сервер, фрэймворк и т. д.).

Для достижения отказа в обслуживании с помощью атак прикладного уровня (L7) атакующему может потребоваться существенно меньшее количество ресурсов. Например, если для успешной реализация атаки SYN Flood данная должна быть распределенной (DDoS) и использовать ботнет, то для реализации атак класса Slow HTTP DoS (Slowloris, Slow HTTP Post, Slow HTTP Read) обычно достаточно одного компьютера [10–13].

Вместе с тем для веб-приложений также характерны DoS-атаки уровня приложения, возможные из-за наличия уязвимостей в данного коде [9]. Например, возможность проведения атаки типа SQL injection может позволить злоумышленнику удалить базу данных с учетными записями пользователей или выполнить запрос вида selectbenchmark(100000000, now()) для израсходования ресурсов системы. Также примерами атак уровня приложения являются атаки XML BillionLaughs (XML Bomb), XML QuadraticBlowupAttack, ZIP ofDeath (ZIP Bomb).

Методические указания и порядок выполнения работы. Задание

Цель лабораторной работы предполагает освоение методик аудита информационной безопасности веб-приложений, изучение современных инструментов тестирования безопасности и приобретение навыков выявления уязвимостей согласно OWASP Top 10.

В рамках лабораторной работы выполняются следующие задачи:

Изучение актуального перечня OWASP Top 10 и категорий уязвимостей веб-приложений.

Настройка прокси-сервера BurpSuite или OWASP ZAP для перехвата и анализа HTTP-трафика.

Анализ структуры тестового веб-приложения (например, DVWA, WebGoat, JuiceShop) и выявление функциональных возможностей.

Выявление и эксплуатация уязвимостей категории A03 Injection (SQL-инъекции, commandinjection).

Выявление и эксплуатация уязвимостей категории A01 BrokenAccessControl (IDOR, обход авторизации).

Выявление и эксплуатация уязвимостей XSS (Reflected, Stored, DOM-based).

Анализ управления сессиями и проверка корректности механизмов аутентификации.

Анализ конфигурации веб-сервера и проверка наличия заголовков безопасности (CSP, HSTS).

Формирование итогового отчёта по результатам аудита с классификацией выявленных уязвимостей по CVSS и рекомендациями по устранению.

Комплекс заданий включает следующие лабораторные работы по основам анализа защищенности веб-приложений:

1. Сбор информации о веб-приложении.
2. Тестирование защищенности транспортного уровня служб SSL/TLS вебсервера www.test.app.com.
3. Тестирование защищенности механизма управления доступом.
4. Тестирование защищенности механизма управления сессиями.
5. Тестирование на устойчивость к атакам отказа в обслуживании.

Методические указания и порядок выполнения работы. Задание

Задание 1. Настройка BurpSuite и анализ структуры приложения

1. Запустите Burp Suite Community Edition. Создайте новый проект: File → NewProject. Перейдите во вкладку Proxy → Options и проверьте настройки прокси: адрес 127.0.0.1, порт 8080.

2. Настройте браузер для проксирования через BurpSuite. В Firefox откройте Settings → Network Settings → Manual proxy configuration, укажите HTTP Proxy: 127.0.0.1, Port: 8080. Установитефлажок Also use this proxy for HTTPS.

3. Импортируйтесертификат CA Burp Suite. В Burp Suite откройте Proxy → Options → Import / Export CA certificate → Certificate in DER format. Сохранитефайлиимпортируйтееговбраузер: Firefox → Settings → Privacy & Security → Certificates → View Certificates → Authorities → Import.

4. Убедитесь, что Burp Suite перехватываеттрафик: вкладка Proxy → Intercept → Intercept is on. Откройте тестовое веб-приложение (например, DVWA, OWASP JuiceShop) в браузере. BurpSuite перехватит все HTTP/HTTPS-запросы.

5. Отключите перехват (Interceptisoff) и переключитесь во вкладку HTTP History. Проанализируйте перехваченные запросы: определите применяемые

технологии (сервер, фреймворк, CMS); изучите механизм управления сессиями (Cookie, JWT); определите параметры запросов.

6. Перейдите во вкладку Target → Sitemap. Изучите структуру приложения: разделы, страницы, API-эндпоинты. Зафиксируйте карту сайта для последующего анализа.

Задание 2. Выявление уязвимостей SQL-инъекций

1. В тестовом приложении найдите форму ввода, параметр которой передаётся в SQL-запрос (например, форма поиска, фильтрации, авторизации).

2. Введите в поле ввода одинарную кавычку (') и отправьте форму. Если приложение уязвимо, сервер вернёт ошибку SQL, указывающую на некорректный синтаксис. Зафиксируйте сообщение об ошибке.

3. Перехватите запрос во вкладке BurpSuiteProxy → Intercept. Отправьте запрос во вкладку Repeater (Ctrl+R). Во вкладке Repeater проанализируйте структуру запроса и попробуйте внедрить SQL-код: ' OR '1'='1' -- , ' UNION SELECT NULL, NULL, NULL -- , ' AND 1=1 -- , ' AND 1=2 -- .

4. Если уязвимость подтверждена, определите количество колонок в SQL-запросе методом ORDER BY: ' ORDER BY 1 -- , ' ORDER BY 2 -- , ... до получения ошибки. Затемопределитевыводимыеколонки: ' UNION SELECT 1, 2, 3 -- .

5. Извлекитеданныеизбазы: ' UNION SELECT table_name, NULL, NULL FROM information_schema.tables -- ; ' UNION SELECT column_name, NULL, NULL FROM information_schema.columns WHERE table_name='users' -- ; ' UNION SELECT username, password, NULL FROM users -- .

6. Если ручной анализ затруднён, примените sqlmap для автоматизированной эксплуатации: sqlmap -u "http://<URL>/?id=1" --dbs. sqlmap автоматически определит тип инъекции, извлечёт базы данных, таблицы и данные. Зафиксируйте результаты.

7. Сформулируйте рекомендации по устранению уязвимости: применение preparedstatements (параметризованных запросов); валидация и санитизация ввода; применение ORM вместо прямых SQL-запросов; настройка WAF для блокировки SQL-инъекций.

Задание 3. Выявление уязвимостей XSS

1. В тестовом приложении найдите поле ввода, значение которого отображается на странице без санитизации (форма комментариев, поиск, профиль пользователя).

2. Введите тестовую последовательность для выявления Reflected XSS: <script>alert('XSS')</script>. Если приложение уязвимо, при отправке формы в браузере отобразится диалоговое окно с сообщением XSS.

3. Если базовая полезная нагрузка фильтруется, попробуйте обход фильтров: <imgsrc=x onerror=alert(1)>– использование обработчика событий;

<svgonload=alert(1)>– использование SVG; <bodyonload=alert(1)>– использование body; javascript:alert(1) – использование протокола javascript: ; <ScRiPt>alert(1)</ScRiPt>– смешанный регистр; <scr<script>ipt>alert(1)</scr</script>ipt>– вложенные теги.

4. Для выявления Stored XSS введите полезную нагрузку в поле, значение которого сохраняется на сервере (комментарий, профиль). Откройте страницу с сохранённым значением в другом браузере или инкогнито-режиме – если XSS сработает, уязвимость подтверждена.

5. Для выявления DOM-based XSS проанализируйте JavaScript-код приложения на наличие использования источников данных (location.hash, location.search, document.referrer) без санитизации. Введите полезную нагрузку в URL-фрагмент: http://<URL>/page.html#<img src=x onerror=alert(1)>.

6. Сформулируйте рекомендации по устранению: outputencoding (преобразование специальных символов в HTML-сущности); применение CSP (Content-Security-Policy) для ограничения источников выполнения скриптов; валидация ввода на стороне сервера; использование фреймворков с автоматической санитизацией.

Задание 4. Анализ управления сессиями и аутентификацией

1. Авторизуйтесь в тестовом приложении. В BurpSuite перейдите во вкладку HTTP History и найдите запрос, устанавливающий Cookie сессии. Зафиксируйте значение идентификатора сессии (например, PHPSESSID, JSESSIONID, session_id).

2. Проверьте предсказуемость идентификатора сессии. Выполните несколько авторизаций и сравните значения идентификаторов. Если значения предсказуемы (последовательные числа, основанные на времени), сессии могут быть перехвачены.

3. Проверьте наличие атрибутов безопасности у Cookie. В ответе сервера найдите заголовок Set-Cookie и проверьте наличие атрибутов: HttpOnly (запрещает доступ к Cookie через JavaScript, защита от XSS); Secure (передача только по HTTPS); SameSite=Strict/Lax (защита от CSRF).

4. Проверьте корректность механизма logout. Авторизуйтесь, скопируйте Cookie сессии, выполните logout, затем отправьте запрос с сохранённым Cookie. Если сервер принимает запрос, сессия не была корректно завершена на сервере.

5. Проверьте политику паролей. Попробуйте зарегистрировать пользователя со слабым паролем (123456, password, admin). Проверьте наличие требований к минимальной длине, сложности и отсутствию в списках скомпрометированных паролей.

6. Проверьте наличие защиты от брутфорс-атак. Выполните 10 неудачных попыток входа с неверным паролем. Если блокировка аккаунта или

ограничение частоты запросов не срабатывает, приложение уязвимо к брутфорс-атакам.

7. Проверьте возможность обхода аутентификации. Попробуйте SQL-инъекцию в форме входа: admin' -- ,admin' OR '1'='1' -- . Проверьте доступ к защищённым страницам без авторизации (direct URL access).

Задание 5. Анализ конфигурации веб-сервера и проверка заголовков безопасности

1. Проанализируйте HTTP-заголовки ответов сервера. В BurpSuite откройте любой ответ сервера и изучите заголовки. Проверьте наличие заголовков безопасности: Content-Security-Policy (ограничивает источники выполнения скриптов); Strict-Transport-Security (HSTS – принудительное использование HTTPS); X-Frame-Options (защита от clickjacking); X-Content-Type-Options (предотвращение MIME-sniffing).

2. Проверьте доступность служебных файлов. Попробуйте обратиться к: /robots.txt – может содержать пути к закрытым разделам; /.git/ – может содержать исходный код приложения; /.env– может содержать пароли и ключи; /backup.zip, /backup.sql– могут содержать резервные копии; /phpinfo.php– может раскрывать конфигурацию сервера.

3. Проверьте обработку ошибок. Обратитесь к несуществующей странице (/nonexistent_page_12345) и проанализируйте сообщение об ошибке. Если ошибка содержит технические подробности (версия сервера, путь к файлам, стек вызовов), это облегчает атаки.

4. Проверьте версию веб-сервера и применяемых библиотек. Заголовок Server может содержать версию (Apache/2.4.41, nginx/1.18.0). Сопоставьте выявленные версии с базой известных уязвимостей (CVE).

5. Выполните активное сканирование с применением BurpSuite. Во вкладке Target → Sitemap выберите целевой хост, нажмите правой кнопкой мыши → Activescan. Дождитесь завершения сканирования и проанализируйте выявленные уязвимости.

6. Сформируйте итоговый отчёт по результатам аудита. Отчёт должен содержать: описание структуры приложения, выявленные уязвимости с классификацией по OWASP Top 10 и CVSS, доказательства эксплуатации, рекомендации по устранению, приоритизированный план корректирующих мероприятий.

Индивидуальное задание

Студент самостоятельно выбирает инфраструктуру организации для рассмотрения при решении лабораторных задач.

Требования к отчету и защите

Защита предполагает опрос по материалу, изложенному в отчете.

Отчет должен быть оформлен: требуется наличие титульного листа, введение (содержит: указание цели лабораторной работы, задачи, краткие теоретические сведения), последовательное изложение разработанного материала, подписанные скриншоты хода работы, если использовалась компьютерная техника, заключение (выводы, анализ полученных результатов).

Проверяется знание теоретического материала с учетом знаний ответов на **Контрольные вопросы**, приведенные в пункте «**Контрольные вопросы**».

Задаются вопросы по пунктам, приведенным в пункте «Методические указания и порядок выполнения работы».

Критерии к оценке защиты отчета студентом приведены в таблице 1 «Шкала оценок уровня».

Контрольные вопросы

1. Что такое OWASP Top 10? Какова роль данного перечня в обеспечении безопасности веб-приложений?
2. Опишитекатегорию A01 Broken Access Control. Какие типичные уязвимости входят в данную категорию?
3. Опишите категорию A03 Injection. Какие виды инъекций существуют и как они выявляются?
4. Что такое SQL-инъекция? Какие методы предотвращения SQL-инъекций существуют?
5. Чтотакое XSS (Cross-Site Scripting)? Опишите виды XSS и методы предотвращения.
6. Чтотакое CSRF (Cross-Site Request Forgery)? Чем CSRF отличается от XSS?
7. Чтотакое SSRF (Server-Side Request Forgery)? Какие риски связаны с данной уязвимостью?
8. Опишитекатегорию A05 Security Misconfiguration. Какие типичные ошибки конфигурации выявляются при аудите?
9. Чтотакое CSP (Content Security Policy)? Как CSP предотвращает атаки XSS?
10. Чтотакое HSTS (HTTP Strict Transport Security)? Какова роль HSTS в обеспечении безопасности?
11. Какие инструменты применяются для аудита безопасности веб-приложений? Опишите возможности BurpSuite и OWASP ZAP.
12. Что такое DevSecOps? Как интеграция проверок безопасности в процесс CI/CD снижает риски, связанные с уязвимостями веб-приложений?

ЛАБОРАТОРНАЯ РАБОТА № 10. OSSEC HIDS

Общие сведения

Цель: получение навыков обследования организации при проведении аудита информационной безопасности

Программное обеспечение: ПК, ОС, OSSEC HIDS, текстовый редактор

Методический материал по выполнению лабораторных заданий

OSSEC (OpenSourceHost-basedIntrusionDetectionSystem) представляет собой open-source систему обнаружения вторжений на уровне хоста, обеспечивающую анализ журналов, контроль целостности файлов, выявление руткитов и мониторинг безопасности операционных систем. OSSEC поддерживает операционные системы Linux, Windows, macOS, Solaris, AIX, HP-UX, что позволяет применять единое решение для защиты разнородной инфраструктуры. Система построена по клиент-серверной архитектуре с центральным сервером, агрегирующим данные от агентов.

Архитектура OSSEC включает следующие основные компоненты: сервер (ossec-server) – центральный элемент, выполняющий обработку данных от агентов и регистрацию событий; агент (ossec-agentd) – программа, устанавливаемая на контролируемые системы и обеспечивающая сбор журналов, контроль целостности и выявление руткитов; база данных – хранилище данных о состоянии контролируемых систем и зарегистрированных событиях; веб-интерфейс – модуль формирования отчётов и управления системой.

Основные возможности OSSEC включают: анализ журналов (loganalysis) – выявление подозрительной активности в журналах операционной системы, служб и приложений по заданным правилам; контроль целостности файлов (fileintegritymonitoring, FIM) – регулярная проверка контрольных сумм критичных файлов и выявление несанкционированных изменений; выявление руткитов (rootkitdetection) – проверка системы на наличие признаков установки руткитов и скрытых процессов; мониторинг процессов – отслеживание запуска и завершения критичных процессов; реакция на события (activeresponses) – автоматизированное выполнение заданных действий при выявлении подозрительной активности.

Правила OSSEC формируются в формате XML и обеспечивают выявление подозрительных событий в журналах. Правила поддерживают декодеры (decoders), обеспечивающие извлечение полей из записей журналов различных форматов (syslog, WindowsEventLog, Apacheaccesslog, IIS logs). Система содержит более 3000 встроенных правил, выявляющих атаки и аномалии в журналах различных операционных систем, служб и приложений.

Пользовательские правила позволяют расширить систему под специфику конкретной инфраструктуры.

Современной альтернативой OSSEC является Wazuh–форк OSSEC, обеспечивающий расширенные возможности и активное развитие. Wazuh включает интеграцию с ElasticStack (ELK) для визуализации и анализа данных, поддержку интеграции с SIEM-системами, расширенные возможности контроля целостности и выявления уязвимостей. Wazuh применяется в корпоративных инфраструктурах для мониторинга безопасности серверов, рабочих станций и облачных ресурсов, а также для обеспечения соответствия требованиям стандартов PCI DSS, HIPAA, GDPR, ФСТЭК.

С позиции информационной безопасности OSSEC и Wazuh применяются для выявления атак, которые не могут быть обнаружены сетевыми IDS/IPS: попыток повышения привилегий, изменений критичных системных файлов, установки бэкдоров и руткитов, аномалий в работе служб, подозрительной активности пользователей. Интеграция хостовых IDS с сетевыми IDS/IPS и SIEM-системами обеспечивает комплексный мониторинг безопасности и корреляцию событий из различных источников, что особенно важно для выявления сложных целенаправленных атак (APT).

HIDS (хостовая система обнаружения вторжений) – имеется система для обнаружения вторжений, данная наблюдает и анализирует события, указанные происходят внутри системы (в отличие от сетевой СОВ, указанная отслеживает в первую очередь сетевой трафик).

OSSEC – имеется хостовая система обнаружения вторжений (HIDS), свободная и с открытым исходным кодом. Данная ведёт анализ системных логов, проверку целостности, наблюдение за реестром ОС Windows, обнаружение руткитов, оповещение в заданное время и если будет обнаружено какое-либо событие. Данная предоставляет функцию обнаружения вторжений для большинства операционных систем, включая Linux, OpenBSD, FreeBSD, Mac OS X, Solaris и Windows.

«Система обнаружения вторжений (СОВ) (англ. IntrusionDetectionSystem (IDS)) – программное (как в рассматриваемом случае) или аппаратное средство, предназначенное для выявления фактов неавторизованного доступа (вторжения или сетевой атаки) в компьютерную систему или сеть.

Цели, достигаемые при помощи IDS:

1. Самая главная – обнаружить вторжение или сетевую атаку;

2. Прогнозирование возможного вторжения – злоумышленник обычно выполняет ряд действий при прощупывании исследуемой безопасности, указанное действие оставит след;

3. Обеспечить централизованный контроль над безопасностью сети (особенно, большой и распределенной);

4. Многие другие цели, указанные допускается придумать сами.

Системы IDS делятсяна network-based и host-based.

NetworkBased IDS определяют атаки, захватывая и анализируя сетевые пакеты. Слушая сетевой сегмент, NIDS может просматривать сетевой трафик от нескольких хостов, указанные присоединены к сетевому сегменту, и таким образом защищать данные хосты.

HostBased IDS имеют дело с информацией, собранной внутри единственного компьютера. Подобное выгодное расположение позволяет HIDS анализировать деятельность с большой достоверностью и точностью, определяя только указанные процессы и пользователей, имеющие отношение к конкретной атаке в ОС. Более того, в отличии от network-based IDS, host-based IDS могут «видеть» последствия предпринятой атаки, так как данные могут иметь непосредственный доступ к системной информации, файлам данных и системным процессам, являющимся целью атаки» [6].

HIDS в основном используют информационные источники двух типов: результаты аудита ОС и системные логи. Результаты аудита ОС обычно создаются на уровне ядра ОС и, следовательно, являются более детальными и лучше защищенными, чем системные логи. Однако системные логи намного меньше и не столь многочисленны, как результаты аудита, и, следовательно, легче для понимания. Некоторые hostbased IDS разработаны для поддержки централизованной инфраструктуры управления и получения отчетов IDS, что может допускать единственную консоль управления для отслеживания многих хостов. Другие создают сообщения в формате, указанный совместим с системами сетевого управления.

OSSEC – имеется хостовая система обнаружения вторжений (HIDS), свободная и с открытым исходным кодом. Данная ведёт анализ системных логов, проверку целостности, наблюдение за реестром ОС Windows, обнаружение руткитов, оповещение в заданное время и если будет обнаружено какое-либо событие. Данная предоставляет функцию обнаружения вторжений для большинства операционных систем, включая Linux, OpenBSD, FreeBSD, Mac OS X, Solaris и Windows.

OSSEC помогает клиентам соответствовать определенным требованиям, таким как PCI и HIPAA. Данный объект позволяет клиентам обнаруживать и предупреждать о несанкционированных изменениях файловой системы и

вредоносном поведении, встроенных в файлы журналов коммерческих продуктов, а также пользовательских приложений. Для PCI данный объект охватывает разделы мониторинга целостности файлов (PCI 11.5, 10.5), проверки и мониторинга журналов (раздел 10), а также применения / проверки политик.

Мультиплатформенность. OSSEC позволяет клиентам реализовать комплексную систему обнаружения вторжений на основе хоста с детализированными политиками для конкретных приложений / серверов на нескольких платформах, таких как Linux, Solaris, Windows и Mac OS X.

Настраиваемые оповещения в реальном времени. OSSEC позволяет клиентам настраивать инциденты, о указанных данные хотят получать уведомления, и позволяет данным сосредоточиться на повышении приоритета критических инцидентов над обычным шумом в любой системе. Интеграция с smtp, sms и syslog позволяет клиентам быть в курсе предупреждений, отправляя указанных на устройства с поддержкой электронной почты. Также доступны варианты активного ответа для немедленного блокирования атаки.

Интеграция с существующей инфраструктурой. OSSEC будет интегрироваться с текущими инвестициями клиентов, такими как продукты SIM / SEM (SecurityIncidentManagement / SecurityEventsManagement) для централизованной отчетности и корреляции событий.

Централизованное управление. OSSEC предоставляет упрощенный централизованный сервер управления для управления политиками в нескольких операционных системах. Кроме того, данный объект также позволяет клиентам определять специфичные для сервера переопределения для более тонких политик.

4. Агентный и безагентный мониторинг

5. OSSEC предлагает гибкость агентного и безагентного мониторинга систем и сетевых компонентов, таких как маршрутизаторы и межсетевые экраны. Безагентный мониторинг позволяет клиентам, у указанных есть ограничения наустановку программного обеспечения в системах (например, одобренные FDA системы или устройства), обеспечивать безопасность и соблюдение нормативных требований» [3].

6. Проверка целостности файлов

7. Все произведенные атаки на сеть и компьютеры объединяет одно: данные изменяют указанные системы. Необходимость проверки целостности файлов (или FIM – мониторинга целостности файлов) в том, чтобы обнаружить все изменения и предупредить при выполнении, когда данные произойдут. Может быть атака, злоупотребление со стороны сотрудника или даже опечатка со стороны администратора, при выполнении будет сообщено о любом изменении файла, каталога или реестра.

8. Хакеры-преступники хотят скрыть соответствующие действия, но, используя обнаружение руткитов, допускается получать уведомления об изменении системы обычным для руткитов способом.

9. Активный ответ позволяет OSSEC предпринимать немедленные действия при срабатывании определенных предупреждений. Имеется может предотвратить распространение инцидента до того, как администратор сможет принять меры.

Методические указания и порядок выполнения работы. Задание

В рамках лабораторной работы выполняются следующие задачи:

1. Установка и настройка сервера OSSEC на операционной системе Linux.

2. Установка и настройка агента OSSEC на контролируемой системе (Linux или Windows).

3. Изучение структуры конфигурационных файлов и принципов работы декодеров.

4. Настройка контроля целостности критичных файлов и проверка работы механизма FIM.

5. Настройка правил выявления подозрительной активности в журналах операционной системы.

6. Генерация тестовых событий и анализ зарегистрированных предупреждений.

7. Настройка активных реакций (activeresponses) на выявленные события.

8. Изучение возможностей интеграции OSSEC с SIEM-системами и формирование итоговых отчётов.

9. Рассмотреть работу OSSEC HIDS

10. Рассмотреть работу OSSEC HIDS с применением Apache и Web-интерфейса.

Задание 1. Установка и настройка сервера OSSEC

1. Загрузите архив OSSEC сервера: wget https://github.com/ossec/ossec-hids/releases/download/3.6.0/ossec-hids-3.6.0.tar.gz. Распакуйте архив: tarxzf ossec-hids-3.6.0.tar.gz &&cd ossec-hids-3.6.0.

2. Установитезависимости: sudo apt install build-essential libssl-dev libpcre2-dev zlib1g-dev libevent-dev. Запустите установочный скрипт: sudo ./install.sh. Выберите тип установки server, язык en, примите стандартные настройки установки. При запросе на настройку электронной почты укажите параметры SMTP-сервера (можно пропустить).

3. После завершения установки запустите сервис: sudo /var/ossec/bin/ossec-controlstart. Проверьте статус: sudo /var/ossec/bin/ossec-controlstatus. Убедитесь, что все компоненты (ossec-monitord, ossec-logcollector, ossec-analysisd, ossec-maild, ossec-execd, ossec-dbd) запущены.

4. Откройте файл конфигурации: sudonano /var/ossec/etc/ossec.conf. Изучите основные разделы: global– глобальные настройки (email, syslog); rules– пути к базам правил; syscheck– настройки контроля целостности файлов; rootcheck– настройки выявления руткитов; localfile– мониторинг дополнительных журналов.

5. Настройте переменные сетевых сегментов. Добавьте в раздел global: <global><email_notification>yes</email_notification><email_to>admin@example.com</email_to><smtp_server>localhost</smtp_server></global>. При необходимости настройте отправку уведомлений в SIEM через syslog: <global><logall>yes</logall></global>.

Задание 2. Настройка контроля целостности файлов (FIM)

1. В файле ossec.conf найдите раздел syscheck. Изучитеформатдиректив: <directories check_all="yes" realtime="yes">/etc,/usr/bin,/usr/sbin,/bin,/sbin</directories>. Параметр check_all включает все проверки (хэш, размер, права, владелец), realtime– мониторинг в реальном времени через inotify.

2. Добавьте в список контролируемых каталоги: <directoriescheck_all="yes" realtime="yes">/var/www,/var/lib/mysql,/etc/nginx</directories>. Эти каталоги содержат конфигурацию веб-сервера, базы данных и приложений.

3. Настройте игнорирование изменений в несущественных файлах. Добавьте: <ignore>/etc/mtab</ignore> (динамический файл монтирования); <ignore>/etc/hostname</ignore> (может изменяться при DHCP); <ignore>/var/log/ossec</ignore> (собственные журналы OSSEC);

<ignoretype="sregex">.log$|.swp$</ignore> (игнорирование файлов логов и swap-файлов).

4. Перезапустите OSSEC: sudo /var/ossec/bin/ossec-control restart. Проверь тежурнал: sudo tail -f /var/ossec/logs/ossec.log.

5. Внесите тестовые изменения в контролируемые файлы: sudo touch /etc/test_file (создание нового файла); sudo echo "test content" | sudo tee -a /etc/passwd (изменение существующего файла); sudochmod 777 /etc/shadow (изменение прав доступа).

6. Проверьте журнал предупреждений: sudo tail -f /var/ossec/logs/alerts/alerts.log. Зафиксируйтесобытияизмененияфайлов. Для каждого события определите: время, тип изменения (создание, модификация, удаление), путь к файлу, характер изменения.

Задание 3. Настройка агента OSSEC и подключение к серверу

1. На контролируемой системе (Linux) установите агент OSSEC. Загрузите и распакуйте архив, запустите установщик: sudo ./install.sh, выберите тип agent.

2. На сервере добавьте агента: sudo /var/ossec/bin/manage_agents. Выберите опцию A (Addanagent), укажите имя агента (например, lab-agent), IP-адрес агента, идентификатор (можно использовать auto).

3. Извлеките ключ агента: в manage_agents выберите опцию E (Extractkeyforanagent), укажите идентификатор. Скопируйте строку ключа в буфер обмена.

4. На системе агента импортируйте ключ: sudo /var/ossec/bin/manage_agents. Выберите опцию I (Importkeyfromserver), вставьте ключ, подтвердите добавление.

5. На системе агента укажите IP-адрес сервера. Откройте файл конфигурации: sudonano /var/ossec/etc/ossec.conf. В разделе client задайте: <client><server-ip>192.168.1.10</server-ip></client> (укажите IP-адрес сервера OSSEC).

6. Запустите агент: sudo /var/ossec/bin/ossec-control start. Проверьтестатус: sudo /var/ossec/bin/ossec-control status.

7. На сервере проверьте подключение агента. В журнале: sudotail -f /var/ossec/logs/active-responses.log. Также выполните: sudo /var/ossec/bin/agent_control -lc (список подключённых агентов) и sudo /var/ossec/bin/agent_control -i <ID_агента> (информация об агенте).

Задание 4. Настройка правил выявления подозрительной активности

1. Создайте пользовательское правило для выявления SSH-брутфорс. Откройтефайл: sudo nano /var/ossec/rules/local_rules.xml. Добавьте: <group name="local,syslog,sshd,"><rule id="100100" level="5"><if_sid>530</if_sid><match>Failed

password</match><description>SSH: Failed password attempt</description></rule><rule id="100101" level="10" frequency="5" timeframe="60"><if_matched_sid>100100</if_matched_sid><same_source_ip /><description>SSH: Multiple failed password attempts (brute force)</description></rule></group>. Правило 100100 регистрирует каждую неудачную попытку, правило 100101 генерирует предупреждение при 5 попытках за 60 секунд.

2. Создайте правило для выявления изменения критичных файлов: <rule id="100102"
level="7"><if_group="syscheck"<match>/etc/passwd|/etc/shadow|/etc/sudoers</match><description>Critical system file modified</description></rule>.

3. Перезапустите OSSEC: sudo /var/ossec/bin/ossec-control restart.

4. Сгенерируйте тестовые события. На системе с агентом выполните: несколько неудачных попыток входа по SSH (sshnonexistent_user@localhost с неверным паролем); изменение файла /etc/passwd (sudouseraddtestuser).

5. Проверьте срабатывание правил на сервере: sudotail -f /var/ossec/logs/alerts/alerts.log. Зафиксируйте срабатывания правил 100100, 100101 и 100102.

Задание 5. Настройка активных реакций

1. Настройте активную реакцию блокировки IP-адреса при выявлении SSH-брутфорс. Вфайлеossec.confнасервередобавьтевраздел command: <command><name>firewall-drop</name><executable>firewall-drop.sh</executable><expect>srcip</expect><timeout_allowed>yes</timeout_allowed></command>.

2. Вразделactive_responsedобавьте: <active-response><command>firewall-drop</command><location>local</location><rules_id>100101</rules_id><timeout>60</timeout></active-response>. При срабатывании правила 100101 IP-адрес атакующего будет заблокирован на 60 секунд.

3. Перезапустите OSSEC и сгенерируйте SSH-брутфорс с атакующего хоста. Проверьте блокировку: sudoiptables -L -n | grep DROP. Убедитесь, что IP-адрес атакующего заблокирован.

4. Сформируйте итоговый отчёт по результатам выполнения всех заданий. Отчёт должен содержать: описание установки и настройки сервера и агента OSSEC, настройку контроля целостности, созданные правила, результаты тестирования, настройку активных реакций, выводы по эффективности обнаружения атак.

Индивидуальное задание

Студент самостоятельно выбирает инфраструктуру организации для рассмотрения при решении лабораторных задач.

Требования к отчету и защите

Защита предполагает опрос по материалу, изложенному в отчете.

Отчет должен быть оформлен: требуется наличие титульного листа, введение (содержит: указание цели лабораторной работы, задачи, краткие теоретические сведения), последовательное изложение разработанного материала, подписанные скриншоты хода работы, если использовалась компьютерная техника, заключение (выводы, анализ полученных результатов).

Проверяется знание теоретического материала с учетом знаний ответов на **Контрольные вопросы**, приведенные в пункте «**Контрольные вопросы**».

Задаются вопросы по пунктам, приведенным в пункте «Методические указания и порядок выполнения работы».

Критерии к оценке защиты отчета студентом приведены в таблице 1 «Шкала оценок уровня».

Контрольные вопросы

1. Что такое хостовая система обнаружения вторжений (HIDS)? Чем HIDS отличается от сетевой IDS (NIDS)?
2. Опишите архитектуру OSSEC. Каковы функции основных компонентов системы?
3. Какие операционные системы поддерживает OSSEC? Какова роль агентов и сервера?
4. Что такое контроль целостности файлов (FIM)? Как данный механизм реализован в OSSEC?
5. Что такое декодеры в OSSEC? Какова их роль в обработке записей журналов?
6. Опишите структуру правил OSSEC. Какие возможности предоставляются для создания пользовательских правил?
7. Что такое активные реакции (activeresponses) в OSSEC? Какие типичные реакции применяются при выявлении подозрительной активности?
8. Какие методы выявления руткитов применяются в OSSEC? Каковы ограничения данных методов?
9. Чем Wazuh отличается от OSSEC? Какие дополнительные возможности обеспечивает Wazuh?
10. Как интегрируется OSSEC/Wazuh с SIEM-системами? Какие преимущества обеспечивает интеграция?
11. Какие стандарты безопасности (PCI DSS, HIPAA, ФСТЭК) требуют применения HIDS? Какие конкретные требования предъявляются?
12. Какие типы атак выявляются хостовыми IDS, но не выявляются сетевыми IDS? Приведите примеры.

ЛАБОРАТОРНАЯ РАБОТА № 11. НАСТРОЙКА РАБОТЫ IPSEC

Общие сведения

Цель: научиться выполнять базовую настройку IPsec в Linux.

Программное обеспечение: OCLinux.

Методический материал по выполнению лабораторных заданий

IPsec (InternetProtocolSecurity) представляет собой набор протоколов для обеспечения защиты сетевого трафика на сетевом уровне модели OSI. IPsec обеспечивает конфиденциальность (шифрование), целостность (HMAC), аутентификацию (цифровые подписи, сертификаты) и защиту от повторного воспроизведения (anti-replay) IP-пакетов. Протокол является обязательной частью IPv6 и опциональной частью IPv4, применяется для построения виртуальных частных сетей (VPN) site-to-site и client-to-site, а также для защиты служебного трафика между узлами инфраструктуры.

Архитектура IPsec включает следующие основные компоненты: протоколы AH (AuthenticationHeader) и ESP (EncapsulatingSecurityPayload), обеспечивающие аутентификацию и шифрование пакетов; протоколы IKE (InternetKeyExchange) для автоматического согласования параметров безопасности и обмена ключами; базы данных политик безопасности (SPD) и ассоциаций безопасности (SAD), определяющие правила обработки трафика; криптографические алгоритмы для шифрования (AES, ChaCha20) и аутентификации (HMAC-SHA, GMAC).

Протокол ESP обеспечивает шифрование и аутентификацию IP-пакетов и применяется в большинстве современных IPsec-развёртываний. Протокол AH обеспечивает только аутентификацию без шифрования и в современных реализациях применяется редко из-за ограничений при прохождении через NAT. ESP поддерживает два режима работы: транспортный режим – защита полезной нагрузки IP-пакета с сохранением исходного заголовка; туннельный режим –полная инкапсуляция исходного IP-пакета в новый IP-пакет с ESP-заголовком, применяемый для построения VPN site-to-site.

Протокол IKE обеспечивает автоматическое согласование параметров безопасности между узлами и выполняется в две фазы. В первой фазе (IKE SA, MainMode или AggressiveMode) формируется защищённый канал между узлами с применением алгоритма Diffie-Hellman для обмена ключами. Во второй фазе (IPsec SA, QuickMode) согласовываются параметры защиты конкретного трафика и формируются ассоциации безопасности. Современная версия IKEv2 обеспечивает улучшенную защиту от атак, поддержку мобильности (MOBIKE) и упрощённый процесс согласования.

Современные реализации IPsec включены в операционные системы Linux (strongSwan, Libreswan, OpenSwan), Windows (NativeIPsec, AlwaysOn VPN), macOS и мобильные платформы (iOS, Android). strongSwan является наиболее распространённой open-source реализацией IPsec для Linux, обеспечивающей поддержку IKEv2, современных криптографических алгоритмов (AES-GCM, ChaCha20-Poly1305), сертификатов X.509 и интеграцию с различными средствами аутентификации (RADIUS, LDAP, OTP).

С позиции информационной безопасности IPsec применяется для построения защищённых VPN-соединений между филиалами организаций (site-to-site VPN), обеспечения удалённого доступа сотрудников к корпоративным ресурсам (remoteaccess VPN), защиты трафика между облачными ресурсами и локальной инфраструктурой, а также для защиты служебного трафика (DNS, LDAP, SMB) внутри локальной сети. Преимуществами IPsec являются стандартизированность, высокая производительность (аппаратное ускорение в современных сетевых картах), поддержка различными устройствами и операционными системами. Альтернативными решениями для построения VPN являются OpenVPN (TLS-based), WireGuard (современный протокол с упрощённой реализацией) и SSTP (Microsoft).

IPsec (сокращение от IP Security) – набор протоколов для обеспечения защиты данных, передаваемых по межсетевому протоколу IP. Позволяет осуществлять подтверждение подлинности (аутентификацию), проверку целостности и/или шифрование IP-пакетов. IPsec также включает в себя протоколы для защищённого обмена ключами в сети Интернет. Используется для организации vpn-соединений [1].

IKE – протокол, связывающий все компоненты IPsec в работающее целое. В частности, IKE обеспечивает первоначальную аутентификацию сторон, а также указанных обмен общими секретными ключами.

В работе протоколов IPsec можно выделить пять этапов [2]:

1. Первый этап начинается с создания на каждом узле, поддерживающим стандарт IPsec, политики безопасности. На данном этапе определяется какой трафик подлежит шифрованию, какие функции и алгоритмы могут быть использованы.

2. Второй этап является по сути первой фазой IKE. Данной цель – организовать безопасный канал между сторонами для второй фазы IKE. На втором этапе выполняются:

– аутентификация и защита идентификационной информации узлов,

– проверка соответствий политик IKE SecurityAssociation (SA) узлов для безопасного обмена ключами,

– обмен Диффи-Хеллмана, в результате указанного у каждого узла будет общий секретный ключ,

– создание безопасного канала для второй фазы IKE.

– Третий этап – является второй фазой IKE. Данного задачей является создание IPsec туннеля. На третьем этапе выполняются следующие функции:

– согласуются параметры IPsec SA по защищаемому IKE SA каналу, созданному в первой фазе IKE,

– устанавливается IPsec SA,

– периодически осуществляется пересмотр IPsec SA, чтобы убедиться в данной безопасности,

– (Опционально) выполняется дополнительный обмен Диффи-Хеллмана.

4. Рабочий этап. После создания IPsec SA начинается обмен информацией между узлами через IPsec-туннель, используются протоколы и параметры, установленные в SA.

5. Прекращают действовать текущие IPsec SA. Имеется происходит при указанных удалении или при истечении времени жизни (определенное в SA в байтах информации, передаваемой через канал, или в секундах), значение указанного содержится в SAD на каждом узле. Если требуется продолжить передачу, запускается фаза два IKE (если требуется, то и первая фаза) и далее создаются новые IPsec SA. Процесс создания новых SA может происходить и до завершения действия текущих, если требуется непрерывная передача данных

Методические указания и порядок выполнения работы. Задание

Задание 1. Установка и настройка strongSwan для site-to-site VPN

1. Установите strongSwan и необходимые плагины: sudoaptinstallstrongswanstrongswan-pkilibcharon-extra-pluginslibcharon-extauth-pluginslibstrongswan-extra-pluginslibstrongswan-standard-plugins. Проверьте версию: ipsecversion.

2. Откройте файл конфигурации: sudonano /etc/ipsec.conf. Очистите существующее содержимое и добавьте базовую конфигурацию для site-to-site VPN. Укажите: configsetup– глобальные настройки; conn %default– параметры по умолчанию для всех соединений; connmyvpn– параметры конкретного соединения.

3. Настройте соединение site-to-site. На первом узле (left) добавьте: connmyvpnauto=startkeyexchange=ikev2 authby=secretleft=192.168.1.10 leftsubnet=192.168.1.0/24 right=192.168.2.10 rightsubnet=192.168.2.0/24 ike=aes256-sha256-modp2048! esp=aes256-sha256!. На втором узле (right) укажите зеркальную конфигурацию с теми же параметрами.

4. В файле /etc/ipsec.secrets задайте общий ключ: 192.168.1.10 192.168.2.10 : PSK "MySecretKey123!". Установите одинаковый ключ на обоих узлах.

5. Перезапустите strongSwan: sudosystemctlrestartstrongswan. Проверьте статус: sudoipsecstatusall. Убедитесь, что соединение установлено (State: ESTABLISHED).

6. Проверьте доступность удалённой подсети через туннель: ping 192.168.2.10 (с первого узла). Проанализируйте трафик: sudotcpdump -i <интерфейс>esporudpport 500 orudpport 4500. Зафиксируйте IKE (UDP 500/4500) и ESP (протокол 50) пакеты.

Задание 2. Настройка IPsec с аутентификацией по сертификатам

1. Создайте центр сертификации (CA) с применением strongswan-pki. Сгенерируйте приватный ключ CA: ipsecpki --gen --typersa --size 4096 --outformpem>ca-key.pem. Создайте сертификат CA: ipsecpki --self --ca --lifetime 3650 --inca-key.pem --typersa --dn "CN=MyVPN-CA" --outformpem>ca-cert.pem.

2. Сгенерируйте ключи и сертификаты для сервера и клиента. Для сервера: ipsecpki --gen --typersa --size 2048 --outformpem>server-key.pem; ipsecpki --pub --inserver-key.pem --typersa | ipsecpki --issue --lifetime 825 --cacertca-cert.pem --cakeyca-key.pem --dn "CN=vpn.example.com" --san vpn.example.com --flagserverAuth --outformpem>server-cert.pem. Аналогично для клиента.

3. Скопируйте сертификаты в каталоги strongSwan: sudocpca-cert.pem /etc/ipsec.d/cacerts/; sudocpserver-cert.pem /etc/ipsec.d/certs/; sudocpserver-key.pem /etc/ipsec.d/private/; sudochmod 600 /etc/ipsec.d/private/server-key.pem.

4. В файле ipsec.conf настройте аутентификацию по сертификатам: connmyvpnauto=startkeyexchange=ikev2 left=192.168.1.10 leftcert=server-cert.pemleftsendcert=alwaysleftsubnet=192.168.1.0/24 right=%anyrightcert=client-cert.pemrightsourceip=10.10.10.0/24 authby=pubkey.

5. В файле ipsec.secrets укажите приватный ключ: : RSA "server-key.pem". Перезапустите strongSwan и проверьте установление туннеля.

Задание 3. Настройка удалённого доступа (RoadWarrior) с IKEv2

1. На сервере IPsec настройте конфигурацию для удалённых клиентов. Bipsec.confдобавьте: conn roadwarrior auto=add keyexchange=ikev2 left=192.168.1.10 leftcert=server-cert.pemleftsendcert=always leftsubnet=0.0.0.0/0 right=%any rightsourceip=10.10.10.0/24 rightdns=8.8.8.8,8.8.4.4 authby=pubkeyeap_identity=%identity.

2. Создайте сертификат клиента и экспортируйте его вместе с приватным ключоми CA-сертификатомвформат PKCS#12: openssl pkcs12 -export -in client-cert.pem -inkey client-key.pem -certfile ca-cert.pem -out client.p12.

3. На клиентском устройстве (Windows 10/11) импортируйте сертификат клиента: certmgr.msc → Personal → Certificates → Import → выберитеclient.p12. Импортируйте CA-сертификат: Trusted Root Certification Authorities → Import → выберите ca-cert.pem.

4. Настройте VPN-подключениев Windows: Settings → Network & Internet → VPN → Add a VPN connection. Укажите: VPN provider – Windows (built-in); Connection name –MyVPN; Server name or address –<IP-адрес_сервера>; VPN type – IKEv2; Type of sign-in info – Certificate.

5. Установите VPN-соединение и проверьте доступность ресурсов через туннель. Проверьте назначенный IP-адрес: ipconfig /all. Проверьте маршрут по умолчанию: routeprint.

6. На сервере проанализируйте трафик: sudotcpdump -i <интерфейс>esporudpport 500 orudpport 4500. Зафиксируйте установление IKE SA (UDP 500) и IPsec SA (ESP).

Задание 4. Сравнение IPsec с OpenVPN и WireGuard

1. Установите OpenVPN: sudo apt install openvpn easy-rsa. Создайте PKI и сгенерируйте сертификаты сервера и клиента. Настройте конфигурацию сервера: /etc/openvpn/server.conf с параметрами protoudp, port 1194, devtun, аутентификация по сертификатам.

2. Запустите OpenVPN-сервер: sudosystemctlstartopenvpn@server. Настройте клиентский конфигурационный файл и установите соединение. Проверьте доступность ресурсов через туннель.

3. УстановитеWireGuard: sudo apt install wireguard. Сгенерируйтеключисервераиклиента: wggenkey | tee privatekey | wgpubkey>publickey. Настройтеконфигурациюсервера: /etc/wireguard/wg0.conf спараметрами Address, ListenPort, PrivateKey, PublicKeyклиента.

4. ЗапуститеWireGuard: sudowg-quick up wg0. Настройте клиентский конфигурационный файл и установите соединение. Проверьте доступность ресурсов через туннель: sudowgshow.

5. Сравните производительность трёх решений с применением iperf3. На сервере: iperf3 -s. На клиенте: iperf3 -c <IP-адрес_сервера> -t 30. Зафиксируйте пропускную способность для каждого VPN-решения.

6. Сравните характеристики решений: сложность настройки; поддерживаемые платформы; протоколы и алгоритмы шифрования; производительность; возможность обхода NAT; масштабируемость; удобство управления клиентами. Сформируйте сравнительную таблицу.

7. Сформируйте итоговый отчёт по результатам выполнения всех заданий. Отчёт должен содержать: описание настройки IPsecsite-to-site и RoadWarrior, настройку аутентификации по сертификатам, сравнение IPsec, OpenVPN и WireGuard, выводы по области применения каждого решения.

Индивидуальное задание

Студент самостоятельно выбирает инфраструктуру организации для рассмотрения при решении лабораторных задач.

Требования к отчету и защите

Защита предполагает опрос по материалу, изложенному в отчете.

Отчет должен быть оформлен: требуется наличие титульного листа, введение (содержит: указание цели лабораторной работы, задачи, краткие теоретические сведения), последовательное изложение разработанного материала, подписанные скриншоты хода работы, если использовалась компьютерная техника, заключение (выводы, анализ полученных результатов).

Проверяется знание теоретического материала с учетом знаний ответов на **Контрольные вопросы**, приведенные в пункте «**Контрольные вопросы**».

Задаются вопросы по пунктам, приведенным в пункте «Методические указания и порядок выполнения работы».

Критерии к оценке защиты отчета студентом приведены в таблице 1 «Шкала оценок уровня».

Контрольные вопросы

1. Что такое IPsec? Какие задачи решает данный набор протоколов?
2. Опишите архитектуру IPsec. Каковы функции протоколов AH и ESP?
3. Чем отличается транспортный режим IPsec от туннельного? В каких случаях применяется каждый из них?
4. Что такое ассоциация безопасности (SA)? Какова роль баз SPD и SAD?
5. Опишите протокол IKE. Какие фазы согласования существуют и какие задачи решаются на каждой из них?
6. Чем IKEv1 отличается от IKEv2? Какие преимущества обеспечивает IKEv2?
7. Какие криптографические алгоритмы применяются в IPsec? Каковы современные рекомендации по выбору алгоритмов?
8. Что такое Diffie-Hellman? Как данный алгоритм применяется в IPsec для обмена ключами?
9. Опишите возможности strongSwan. Каковы преимущества данной реализации IPsec для Linux?
10. Чтотакое Always On VPN в Windows? Какова роль данной технологии в обеспечении безопасности удалённого доступа?
11. Чем IPsec отличается от OpenVPN и WireGuard? Каковы преимущества и недостатки каждого решения?

12. Какие сценарии применения IPsec существуют в корпоративных инфраструктурах? Приведите примеры.

ЛАБОРАТОРНАЯ РАБОТА № 12. ПРОГРАММНОЕ ЛОГИРОВАНИЕ СЕТЕВЫХ ПАКЕТОВ В ФАЙЛ

Общие сведения

Цель: получит навыки создания программ на языке C++, которая в режиме непрерывного мониторинга осуществляет запись в файл некоторых атрибутов, выбранных пользователем сетевых пакетов.

Требуется написать программу, указанная в режиме непрерывного мониторинга осуществляет обнаружение заданной сетевой атаки, автоматически реагирует на атаку и сохраняет данные о подозрительной активности.

Программное обеспечение: Snort 2.6.1.4, сниффер WinDump 3.8.3, сканер NMap 4.20, библиотека WinPcap 4.0, Unicorn, XSpider, hping3.

Методический материал по выполнению лабораторных заданий

Программное логирование сетевых пакетов представляет собой процесс регистрации сетевого трафика в файлы или базы данных с применением программных средств. Данная возможность применяется при расследовании инцидентов информационной безопасности, мониторинге сетевой активности, отладке сетевых приложений и проведении криминалистического анализа. Современные системы логирования обеспечивают захват, фильтрацию, агрегацию и долгосрочное хранение сетевого трафика с возможностью последующего поиска и анализа.

Основные инструменты программного логирования сетевых пакетов включают: tcpdump – классическую утилиту командной строки для захвата и записи пакетов в формате pcap, поддерживающую фильтры BPF; dumpcap – утилиту захвата, применяемую в Wireshark и tshark; tshark – консольную версию Wireshark с возможностью декодирования протоколов; argus – систему анализа сетевых потоков с возможностью генерации метаданных; Zeek (ранее Bro) – платформу анализа сетевого трафика с поддержкой сценариев и генерацией структурированных логов.

Форматы хранения данных о сетевом трафике включают: pcap (libpcap) – классический формат захвата пакетов, поддерживаемый большинством инструментов; pcapng – расширенный формат с поддержкой нескольких интерфейсов захвата и аннотаций; NetFlow (v5, v9, IPFIX) – формат агрегированных данных о сетевых потоках, обеспечивающий компактное хранение метаданных без полезной нагрузки; JSON, CSV – текстовые форматы,

применяемые для хранения структурированных метаданных и интеграции с системами анализа.

Современные системы агрегации и анализа сетевых логов включают: ElasticStack (ELK) – комплекс из Elasticsearch, Logstash и Kibana для сбора, хранения и визуализации данных; Splunk – коммерческую платформу для анализа машинных данных; Graylog–open-source систему централизованного управления логами; ApacheKafka – платформу потоковой обработки данных. Данные системы обеспечивают приём логов из различных источников, их нормализацию, долгосрочное хранение, поиск с применением специализированных языков запросов и визуализацию через дашборды.

Программное логирование сетевых пакетов в файл на Python может быть реализовано с применением библиотеки scapy, обеспечивающей возможности захвата, фильтрации и анализа пакетов. Библиотека scapy поддерживает формирование и отправку пакетов, декодирование протоколов всех уровней, а также запись и чтение файлов формата pcap. Программный подход обеспечивает гибкую настройку фильтрации, возможность реализации пользовательской логики обработки и интеграцию с внешними системами.

С позиции информационной безопасности программное логирование применяется для: регистрации сетевой активности при расследовании инцидентов; выявления аномалий и подозрительной активности с применением методов анализа метаданных; контроля соблюдения политик безопасности; обеспечения соответствия требованиям регуляторов (ФСТЭК, PCI DSS), требующих хранения журналов сетевой активности в течение определённого периода. Современные системы полного захвата пакетов (fullpacketcapture, PCAP) обеспечивают хранение всего сетевого трафика за длительный период, что особенно важно при расследовании сложных инцидентов и целенаправленных атак (APT).

Методические указания и порядок выполнения работы. Задание

В рамках лабораторной работы выполняются следующие задачи:

1. Изучение инструментов командной строки для захвата и записи сетевых пакетов в файл (tcpdump, tshark).

2. Освоение форматов хранения данных о сетевом трафике (pcap, pcapng, NetFlow).

3. Установка и настройка библиотеки scapy для работы с сетевыми пакетами в Python.

4. Разработка пользовательского сценария логирования пакетов с применением scapy и запись данных в файл формата pcap.

5. Реализация фильтрации пакетов по заданным критериям (протокол, адреса, порты).

6. Реализация извлечения метаданных пакетов (размер, временные метки, направления) и запись в структурированный формат (CSV, JSON).

7. Анализ записанных данных с применением Python и формирование итогового отчёта по результатам анализа.

Согласовать с преподавателем атрибуты и типы сетевых пакетов, указанные будут логироваться в программе. Написать программу на языке C++, указанная в режиме непрерывного мониторинга осуществляет запись в файл выбранных атрибутов сетевых пакетов.

Задание 1. Захват пакетов с применением tcpdump

1. Установитетcpdump: sudo apt install tcpdump. Получите список сетевых интерфейсов: tcpdump -D. Выберите интерфейс, через который выполняется подключение к сети лаборатории.

2. Запустите захват пакетов с выводом в реальном времени: sudotcpdump -i <интерфейс> -nn. Опция -i задаёт интерфейс, -n отключает разрешение DNS-имён (ускоряет вывод), второй -n отключает разрешение портов в имена сервисов. Наблюдайте за форматом вывода: timestamp, протокол, адреса и порты, флаги, длина.

3. Запишите пакеты в файл формата pcap: sudotcpdump -i <интерфейс> -w capture.pcap -c 1000. Опция -w задаёт файл записи, -c ограничивает количество пакетов. Сгенерируйте тестовый трафик (ping, веб-запросы, DNS) и дождитесь записи 1000 пакетов.

4. Прочитайте записанный файл: tcpdump -r capture.pcap -nn. Примените фильтр при чтении: tcpdump -r capture.pcap 'tcpport 80' -nn. Проанализируйте захваченные пакеты и определите протоколы, наблюдаемые в трафике.

5. Примените фильтры BPF для выборочного захвата. Захватите только TCP-трафик: sudotcpdump -i <интерфейс> 'tcp' -w tcp_only.pcap. Захватите трафик с определённого хоста: sudotcpdump -i <интерфейс> 'host 192.168.1.1' -w host_traffic.pcap. Захватите трафик на определённый порт: sudotcpdump -i

<интерфейс> 'tcpport 443' -w https_traffic.pcap. Сгенерируйте соответствующий трафик для каждого фильтра.

6. Проанализируйте записанные файлы с применением Wireshark. Откройте каждый pcap-файл в Wireshark и изучите декодированные протоколы. Сравните возможности анализа tcpdump и Wireshark.

Задание 2. Захват пакетов с применением tshark

1. Установитетshark: sudo apt install tshark. Проверьте версию: tshark --version. Убедитесь, что поддерживается декодирование актуальных протоколов (HTTP/2, TLS 1.3, QUIC).

2. Запустите захват с детальным декодированием: sudotshark -i <интерфейс> -V. Опция -V включает подробный вывод с полным декодированием всех протоколов. Сравните объём выводимой информации с tcpdump.

3. Запишите пакеты с извлечением определённых полей в CSV-формат: sudotshark -i <интерфейс> -T fields -e frame.time_epoch -e ip.src -e ip.dst -e tcp.dstport -e tcp.flags -e http.request.uri -e frame.len -E header=y -E separator=, -E occurrence=f -w fields.csv. Опция -T fields включает режим извлечения полей, -e задаёт извлекаемые поля, -E header=y добавляет заголовок CSV.

4. Проанализируйте полученный CSV-файл: column -s, -t fields.csv | less. Также можно обработать файл с применением Python и pandas: importpandasaspd; df = pd.read_csv('fields.csv'); print(df.describe()); print(df.groupby('ip.src').size().sort_values(ascending=False).head(10)).

5. Примените фильтры отображения tshark для выборочного декодирования. Захватите только TLS-трафик: sudotshark -i <интерфейс> -Y 'tls' -w tls.pcap. Захватите DNS-запросы: sudotshark -i <интерфейс> -Y 'dns.flags.response == 0' -w dns_queries.pcap. Проанализируйте результаты.

Задание 3. Программное логирование с применением Python и scapy

1. Установите Python iscapy: sudo apt install python3 python3-pip; pip3 install scapy. Проверьтеустановку: python3 -c "from scapy.all import *; print('scapy ready')".

2. Создайте Python-скрипт log_packets.py для захвата и записи пакетов. Скрипт должен: захватывать пакеты с применением функции sniff; для каждого пакета извлекать метаданные (время, адреса, порты, протокол); выводить информацию в консоль; сохранять пакеты в файл pcap.

3. Скрипт должен содержать функцию process_packet, вызываемую для каждого пакета. Функция должна: извлекать IP-адреса источника и назначения; определять протокол; извлекать порты TCP/UDP; выводить информацию в консоль; накапливать пакеты в списке. После остановки захвата (Ctrl+C) скрипт должен сохранять пакеты в файл формата pcap с применением функции wrpcap.

4. Запустите скрипт от имени администратора: sudo python3 log_packets.py. Сгенерируйте тестовый трафик: выполните ping, откройте веб-страницы, выполните DNS-запросы. Остановите скрипт через 60 секунд.

5. Проанализируйте записанный файл capture_scapy.pcap с применением Wireshark. Убедитесь, что все пакеты корректно записаны. Сравните файл с результатами tcpdump.

6. Расширьте функциональность скрипта: добавьте фильтрацию пакетов по протоколу (только TCP, только UDP, только ICMP); добавьте подсчёт статистики (количество пакетов по протоколам, топ IP-адресов); добавьте запись метаданных в CSV-файл одновременно с pcap.

Задание 4. Извлечение метаданных и анализ трафика

1. Создайте Python-скрипт analyze_packets.py для анализа записанного файла pcap. Скрипт должен: читать pcap-файл с применением функции rdpcap; для каждого пакета извлекать метаданные; записывать метаданные в CSV-файл; вычислять статистические показатели.

2. Скрипт должен извлекать следующие метаданные для каждого пакета: временную метку (frame.time_epoch); IP-адрес источника (ip.src); IP-адрес назначения (ip.dst); порт источника (tcp.sport или udp.sport); порт назначения (tcp.dport или udp.dport); протокол (ip.proto); длину пакета (frame.len).

3. Скрипт должен вычислять и выводить статистику: общее количество пакетов; количество пакетов по протоколам (TCP, UDP, ICMP, другие); топ-10 IP-адресов источников по количеству пакетов; топ-10 портов назначения; средний, минимальный и максимальный размер пакета; распределение пакетов по времени.

4. Запустите скрипт: python3 analyze_packets.py. Проанализируйте полученные результаты. Сопоставьте статистику с ожидаемой сетевой активностью.

5. Создайте визуализацию статистики с применением matplotlib. Постройте графики: распределение пакетов по протоколам (круговая диаграмма); топ-10 IP-адресов источников (столбчатая диаграмма); распределение пакетов по времени (линейный график); распределение размеров пакетов (гистограмма).

6. Выявите аномалии в сетевой активности: необычно большое количество пакетов от одного IP-адреса; нестандартные порты назначения; аномальные размеры пакетов; необычные временные паттерны. Зафиксируйте выявленные аномалии и сформулируйте гипотезы об их причинах.

Задание 5. Интеграция с системой централизованного сбора логов

1. Установите ElasticStack (ELK) для централизованного сбора и анализа логов. Установите Elasticsearch, Logstash и Kibana: sudo apt install

elasticsearchlogstashkibana. Запуститесервисы: sudosystemctl start elasticsearchlogstashkibana.

2. Настройте Logstash для приёма данных из CSV-файла. Создайте конфигурационный файл /etc/logstash/conf.d/network.confc секциями input (чтение CSV), filter (парсингинормализация) и output (отправкав Elasticsearch).

3. Отправьте метаданные сетевого трафика в Elasticsearch. Используйте выходной файл CSV из предыдущего задания. Проверьте индексацию данных: curl -X GET 'localhost:9200/_cat/indices?v'.

4. Откройте Kibana в браузере: http://localhost:5601. Создайте indexpattern для сетевых данных. Настройте дашборд с визуализациями: распределение пакетов по протоколам; топ IP-адресов; временная шкала трафика; карта сетевых соединений.

5. Настройте правила выявления аномалий в Kibana: создайте watch или alert для выявления необычного количества пакетов от одного IP-адреса; настройте уведомления по электронной почте при выявлении аномалий.

6. Сформируйте итоговый отчёт по результатам выполнения всех заданий. Отчёт должен содержать: описание применения tcpdump и tshark для захвата пакетов, код Python-скриптов для логирования и анализа, результаты статистического анализа, настройку интеграции с ELK, выводы по эффективности применения программного логирования для расследования инцидентов.

Индивидуальное задание

Разработать блок-схему алгоритма выявления заданной сетевой атаки.

Модифицировать функцию logPacket, добавив в функцию код для обнаружения заданной сетевой атаки.

При определении атаки осуществить запись в файл данных о подозрительной активности и (в случае целесообразности) блокировать атакуемый узел по IP и/или MAC-адресу. Для добавления правила в межсетевой экран Netfilter из пользовательской программы следует использовать функцию system, например, system("iptables -I INPUT 1 -s 202.54.1.2 -j DROP").

Осуществить моделирование заданной сетевой атаки.

требуется написать программу, указанная в режиме непрерывного мониторинга осуществляет обнаружение заданной сетевой атаки, автоматически реагирует на атаку и сохраняет данные о подозрительной активности. Согласуйте с преподавателем один из следующих вариантов заданий:

ЗАДАНИЕ 1. Программная реализация СОВ для обнаружения TCP SYNflood атаки.

ЗАДАНИЕ 2. Программная реализация СОВ для обнаружения ICMP-flood атаки.

ЗАДАНИЕ 3. Программная реализация СОВ для обнаружения DDoS атаки, основанной на протоколе TCP.

ЗАДАНИЕ 4. Программная реализация СОВ для обнаружения атаки «Тунелирование».

ЗАДАНИЕ 5. Программная реализация СОВ для обнаружения TCP-flood атаки.

ЗАДАНИЕ 6. Программная реализация СОВ для обнаружения UDP-flood атаки.

ЗАДАНИЕ 7. Программная реализация СОВ для обнаружения LAND атаки. ЗАДАНИЕ 8. Программная реализация СОВ для обнаружения ICMP-smurf атаки.

ЗАДАНИЕ 9. Программная реализация СОВ для обнаружения атаки крошечными фрагментами (TinyFragmentAttack).

ЗАДАНИЕ 10. Программная реализация СОВ для обнаружения узлов, осуществляющих сканирование в режиме половинного открытия.

ЗАДАНИЕ 11. Программная реализация СОВ для обнаружения узлов, отправляющих пакеты с нестандартными протоколами, инкапсулированными в IP.

ЗАДАНИЕ 12. Программная реализация СОВ для обнаружения узлов, отправляющих пакеты с содержанием в поле данных заданных шаблонов.

Требования к отчету и защите

Защита предполагает опрос по материалу, изложенному в отчете.

Отчет должен быть оформлен: требуется наличие титульного листа, введение (содержит: указание цели лабораторной работы, задачи, краткие теоретические сведения), последовательное изложение разработанного материала, подписанные скриншоты хода работы, если использовалась компьютерная техника, заключение (выводы, анализ полученных результатов).

Проверяется знание теоретического материала с учетом знаний ответов на **Контрольные вопросы**, приведенные в пункте «**Контрольные вопросы**».

Задаются вопросы по пунктам, приведенным в пункте «Методические указания и порядок выполнения работы».

Критерии к оценке защиты отчета студентом приведены в таблице 1 «Шкала оценок уровня».

Контрольные вопросы

1. Что такое программное логирование сетевых пакетов? Какие задачи решает данная технология?

2. Какие инструменты командной строки применяются для захвата и записи сетевых пакетов? Опишите возможности tcpdump и tshark.

3. Чем формат pcap отличается от pcapng? Какие преимущества обеспечивает pcapng?

4. Что такое NetFlow и IPFIX? Чем данные форматы отличаются от pcap?

5. Чтотакое full packet capture (PCAP)? Какие преимущества обеспечивает полное сохранение трафика?

6. Опишите возможности библиотеки scapy для работы с сетевыми пакетами в Python.

7. Какие методы фильтрации пакетов поддерживает scapy? Приведите примеры фильтров.

8. Что такое Elastic Stack (ELK)? Какова роль компонентов Elasticsearch, Logstash и Kibana?

9. Как интегрируются системы логирования сетевого трафика с SIEM-системами?

10. Какие требования к хранению журналов сетевой активности предъявляют стандарты ФСТЭК и PCI DSS?

11. Какие методы анализа метаданных сетевого трафика применяются без декодирования полезной нагрузки?

12. Что такое сетевая криминалистика (networkforensics)? Какова роль программного логирования в расследовании инцидентов?

ЗАКЛЮЧЕНИЕ

Учебно-методическое пособие по дисциплине «Информационная безопасность открытых информационных систем» разработано в соответствии с профилем подготовки студентов специальности 10.05.03 «Информационная безопасность автоматизированных систем» (специализация «Безопасность открытых информационных систем») и ориентировано на формирование у обучающихся прикладных компетенций в сфере защиты цифровой инфраструктуры.

В ходе освоения лабораторного курса студенты последовательно отрабатывают ключевые направления обеспечения информационной безопасности: учатся выявлять и анализировать угрозы и уязвимости в распределённых системах, осваивают принципы работы систем аудита, приобретают навыки проектирования механизмов защиты вычислительных сетей и СУБД, а также методов противодействия исследованию трафика и деструктивным воздействиям злоумышленников. Достижение этих целей обеспечивается комплексом из 12 лабораторных работ, реализуемых в специализированной учебной лаборатории.

Важной составляющей практикума является работа с реальным инструментарием, позволяющим моделировать и анализировать инциденты информационной безопасности. В составе используемого программного обеспечения – как коммерческие продукты (MicrosoftDesktopEducation, KasperskyTotalSpaceSecurity), так и свободно распространяемые средства анализа и защиты (Snort, Ethereal/аналоги, NMAP, MySQL). Дополнительно задействовано специализированное ПО (VMWareWorkstation, «Страж-НТ», «Панцирь-К»), позволяющее отрабатывать сценарии защиты в виртуализированных средах и с применением отечественных решений.

Таким образом, пособие обеспечивает сбалансированное сочетание теоретической базы и практической отработки навыков, необходимых будущему специалисту по информационной безопасности для решения типовых задач на предприятиях: от сканирования и анализа сетевого трафика до построения эшелонированной защиты ресурсов и аудита безопасности. Представленный набор инструментов и лабораторных заданий соответствует современным требованиям к подготовке кадров в области защиты открытых информационных систем и позволяет сформировать у студентов целостное представление о методах и средствах обеспечения безопасности цифровой инфраструктуры.

ЛИТЕРАТУРА

1. Федеральный закон от 27.07.2006 № 149-ФЗ «Об информации, информационных технологиях и о защите информации».
2. Федеральный закон от 27.07.2006 № 152-ФЗ «О персональных данных».
3. ГОСТ Р 57580.1–2017. Безопасность финансовых (банковских) транзакций. Положения по безопасности финансовых транзакций.
4. NIST SP 800-30 Rev. 1. Guide for Conducting Risk Assessments. – Gaithersburg: NIST, 2012. – 95 p.
5. NIST SP 800-61 Rev. 2. Computer Security Incident Handling Guide. – Gaithersburg: NIST, 2012. – 79 p.
6. MITRE ATT&CK Framework [Электронный ресурс]. – URL: https://attack.mitre.org (дата обращения: 01.09.2026).
7. Климов, С. М. Экспериментальная оценка противодействия компьютерным атакам на стендовом полигоне / С. М. Климов, М. П. Сычёв, А. В. Астрахов. – Москва: Радиотехника, 2017. – 112 с.
8. Stallings, W. Cryptography and Network Security: Principles and Practice / W. Stallings. – 7th ed. – Boston: Pearson, 2017. – 768 p.
9. Bishop, M. Computer Security: Art and Science / M. Bishop. – Boston: Addison-Wesley, 2019. – 1104 p.
10. Дойникова, Е. В. Оценивание защищённости и выбор контрмер для управления кибербезопасностью: монография / Е. В. Дойникова, И. В. Котенко. – Москва: РАН, 2021. – 184 с.
11. Краковский, Ю. М. Методы и средства защиты информации: учеб. пособие для вузов / Ю. М. Краковский. – Санкт-Петербург: Лань, 2024. – 272 с.
12. Магазев, А. А. Противодействие сетевым атакам в локальных сетях: учеб. пособие / А. А. Магазев, М. В. Щерба, Е. В. Щерба. – Омск: ОмГТУ, 2021. – 119 с.
13. Подтопельный, В. В. Аудит информационной безопасности: учеб. пособие для студентов специальности 10.05.03 / В. В. Подтопельный. – Калининград: БГАРФ, 2023. – Ч. 1. – 171 с.
14. Баланов, А. Н. Кибербезопасность: учеб. пособие для вузов / А. Н. Баланов. – Санкт-Петербург: Лань, 2024. – 680 с.
15. Баланов, А. Н. Защита информационных систем. Кибербезопасность: учебное пособие для вузов / А. Н. Баланов. – 2-е изд., стер. – Санкт-Петербург: Лань, 2025. – 280 с.

Локальный электронный методический материал

Владислав Владимирович Подтопельный

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ ОТКРЫТЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

Редактор С. Кондрашова
Корректор Т. Звада

Уч.-изд. л. 8,5. Печ. л. 7,5.

Издательство федерального государственного бюджетного образовательного учреждения высшего образования «Калининградский государственный технический университет».
236022, Калининград, Советский проспект, 1