??????? ?????? ? ??????? ????????

Системы аудита

Настройка Sysmon

1. Скачиваем Sysmon с официального сайта: https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon

2. Готовим файл конфигурации или берем готовый:

https://github.com/bakedmuffinman/Neo23x0-sysmon-config/blob/main/sysmonconfig-export.xml

Распаковываем архив, копируем файл конфига.

3. Открываем командную строку Powershell от имени администратора и запускаем команду:

.\Sysmon64.exe -i .\sysmonconfig-export.xml -accepteula

Ключ -i(install) отвечает за установку, опционально указывается файл конфигурации, а ключ -accepteula автоматически принимает лицензионное соглашение, что удобно при установке Sysmon через GPO, ansible или другие способы автоматизации.

4. Проверяем наличие логов.

Откроем остнастку mmc для просмотра событий системы:

PS> eventvwr.msc

Перейдем в раздел "Журналы приложений и служб" -> Microsoft -> Windows -> Sysmon -> Operational и убедимся, что события, описанные в файле конфигурации записываются.

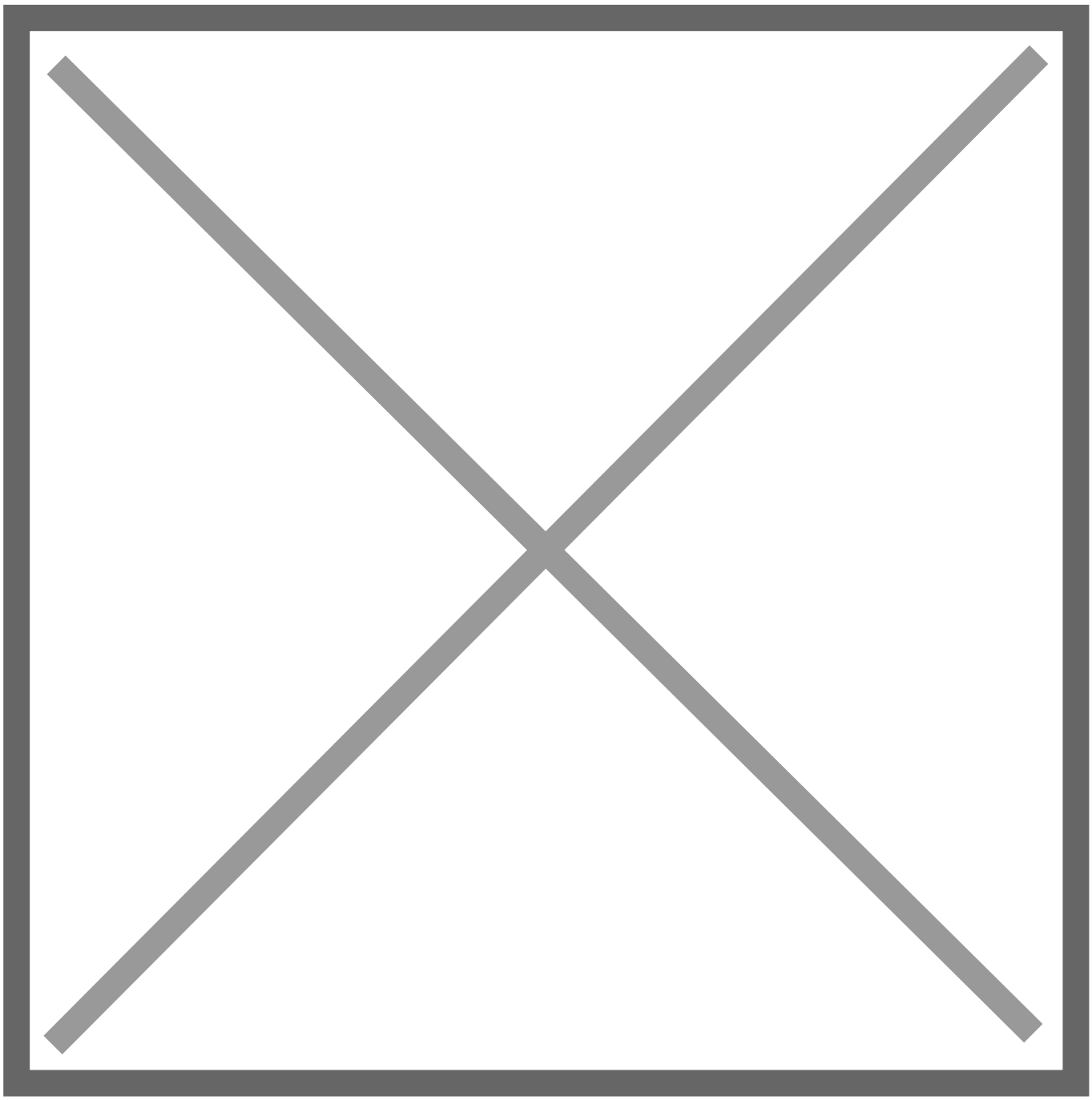
Стандартная конфигурация Sysmon

Давайте разберем мониторинг событий с помощью Sysmon в ОС Windows. Мы будем рассматривать конфиг Sysmon на примере популярного:
https://github.com/bakedmuffinman/Neo23x0-sysmon-config/blob/main/sysmonconfig-export.xml.

> При желании, можно присмотреться в дальнейшем и к этому конфигурационному файлу: https://github.com/SwiftOnSecurity/sysmon-config/blob/master/sysmonconfig-export.xml.

Конфигурация Sysmon описывается в формате XML и указывается при запуске программы.

Пропустим стандартное описание XML-схемы и начнем с разбора групп и правил.



Мы видим описание группы, в которой будет логироваться любое из описанных в группе событий.

`GroupRelation ="and"` в свою очередь будет записывать только те события, котрые попадают под все фильтры в группе.

Итак, внутри группы правил мы видим описание правил `ProcessCreate`. Эти правила будут создавать события с **eventID=**1 при создании системой процесса. Ключ `onmatch` говорит, что делать, в случае если совпало какое-либо условие. В данном примере, в случае совпадения, событие не будет записано (`onmatch = exclude`). То есть, в системный журнал Sysmon будут записаны все события о создании процессов кроме тех, которые указаны в нашей группе. Грубо говоря, мы исключаем неинтересные и часто создаваемые процессы, чтобы уменьшить количество записей в журнале.

Давайте разберем часто встречающиеся фильтры:

- `Image` — имя исполняемого файла;
- `CommandLine` — командная строка;
- `ParentCommandLine` — командная строка родительского процесса;
- `ParentImage` — образ исполняемого файла родительского процесса.

Возможные условия:

- `condition="is"` — жесткое совпадение, например:

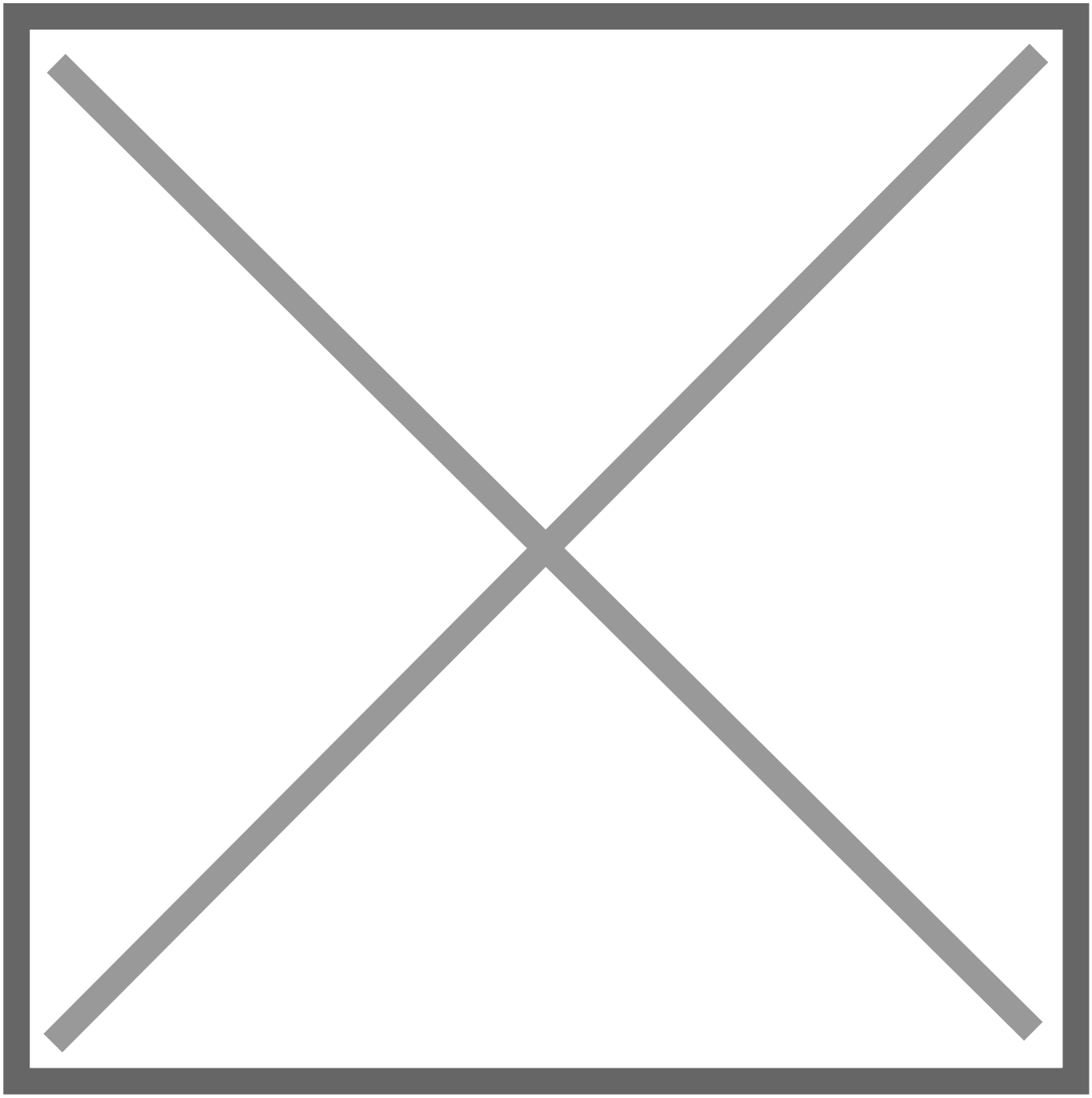
  ```
  <CommandLine condition="is">C:\Windows\System32\usocoreworker.exe -
  Embedding</CommandLine>
  ```

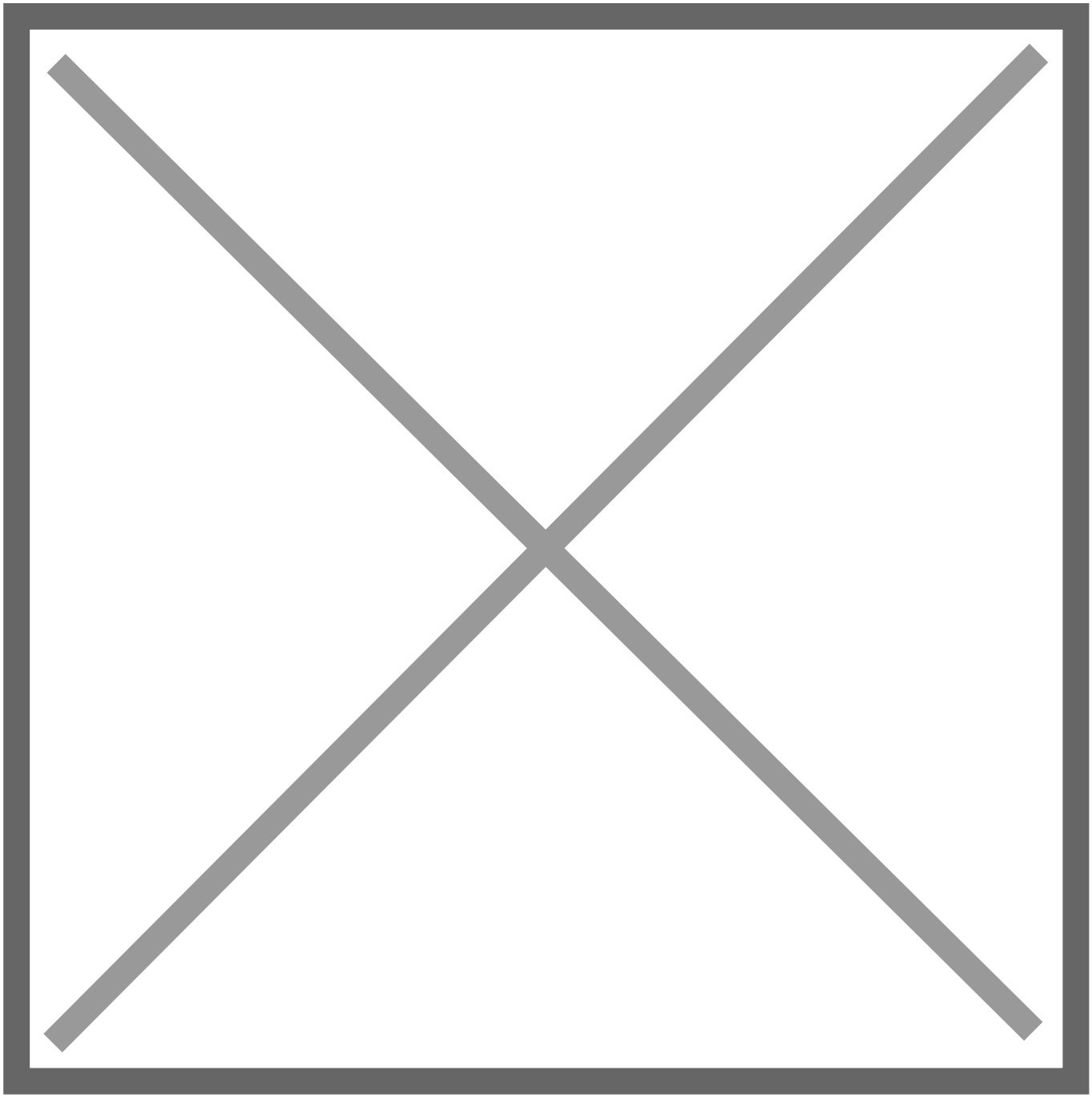
  . В данном примере будет записано(или проигнорировано) событие запуска процесса `usocoreworker.exe` именно с ключом `-Embedding`.
- `condition="contains"` — позволяет фильтровать события по имени файла, процесса, пути.
- `condition="begin with"/ "end with"` — позволяет фильтровать поля, которые начинаются или заканчиваются определенными символами.
- `condition="contains any"/condition="contains all"` — позволяет искать события, которые содержат одно из, либо все условия, описанные в правиле. Список разделяется символом ";", например:

  ```
  <PipeName condition="contains all">MSSE-;-server</PipeName>
  ```

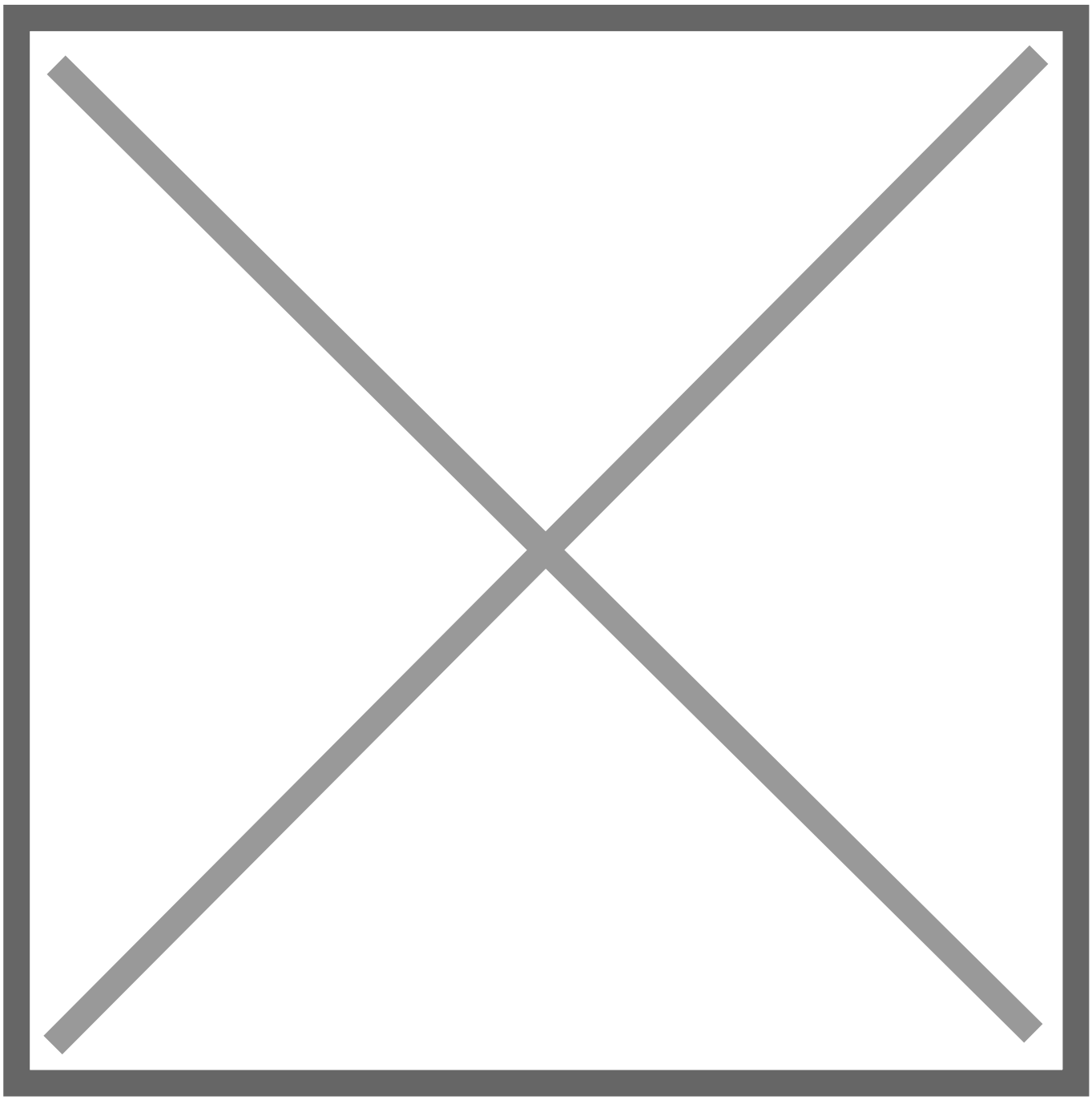


Следующая группа мониторит изменение временной метки создания файлов(**Sysmon eventID=2**). В данном случае мы видим `onmatch = include`, то есть будут записаны только те события, которые описаны в этой группе, а именно: изменение файлов в каталоге "C:\Users", изменения .exe файлов, и `HarddiskVolumeShadowCopy`.

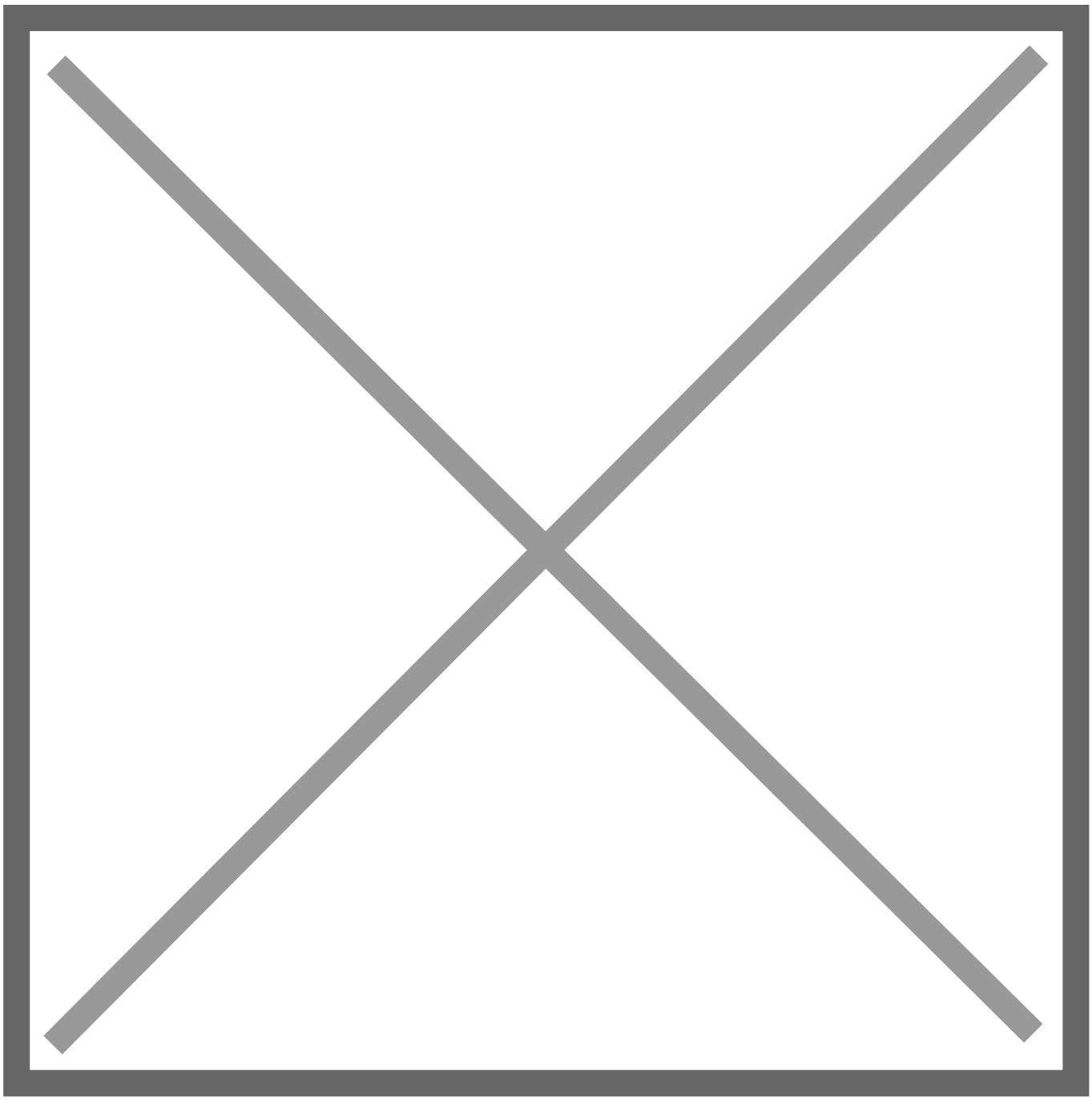


Дальше мы видим группу для мониторинга сетевых соединений(**Sysmon eventID=3**). Как и в случае с файлами, система постоянно инициирует огромное количество сетевых соединений, поэтому мы будем отслеживать только те соединения, которые описаны в этой группе.

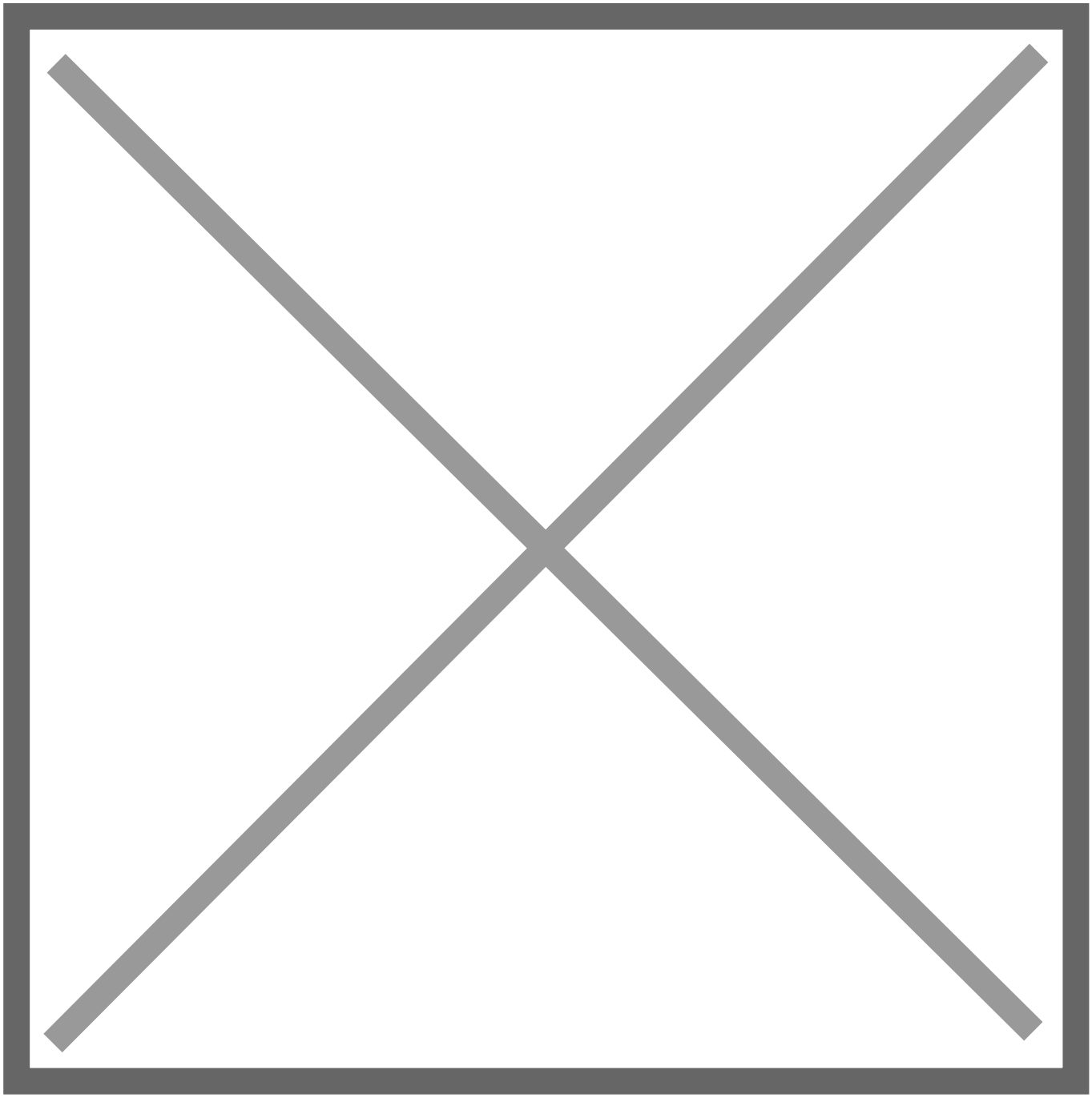
События, описанные в этой группе могут фильтроваться по имени исполняемого файла(`Image`), по IP/FQDN, либо по портам.



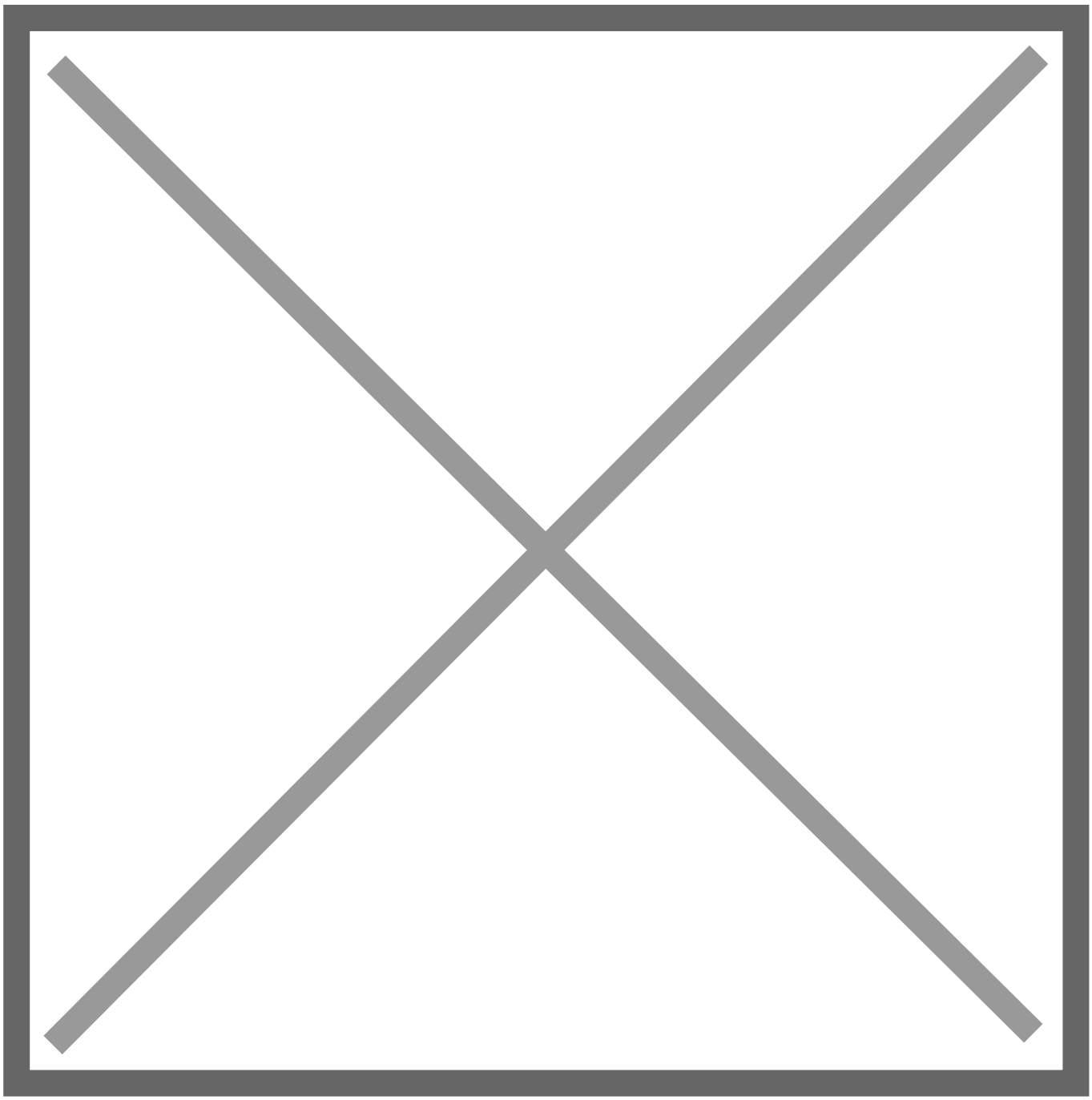
Исключение мониторинга событий коннектов к microsoft.com.



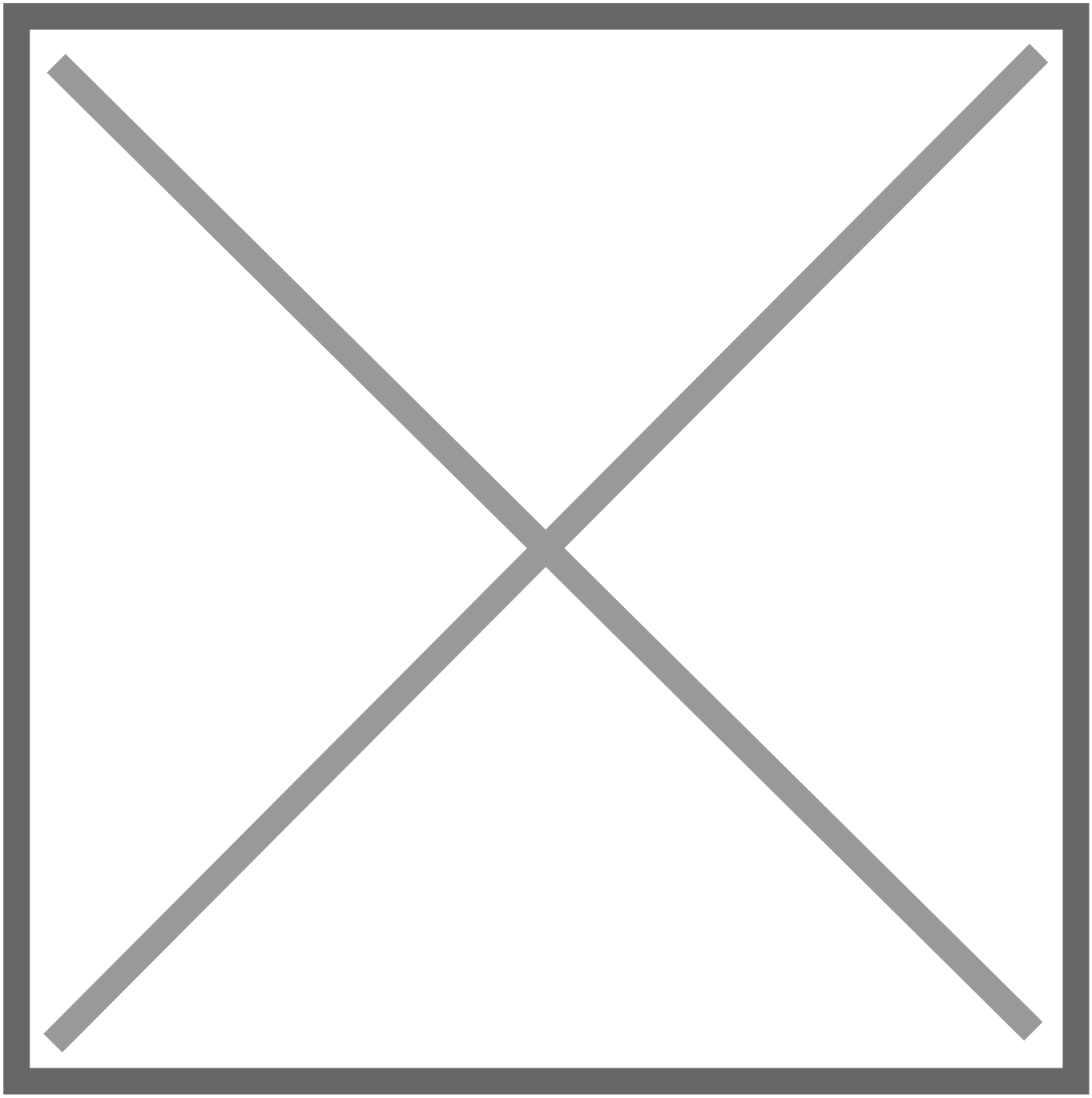
Отслеживание подключений по портам 22,23,143,3389.



Поскольку `onmatch="exclude"` и список пустой будут записываться все события окончания работы процесса(**Sysmon eventID=5**).

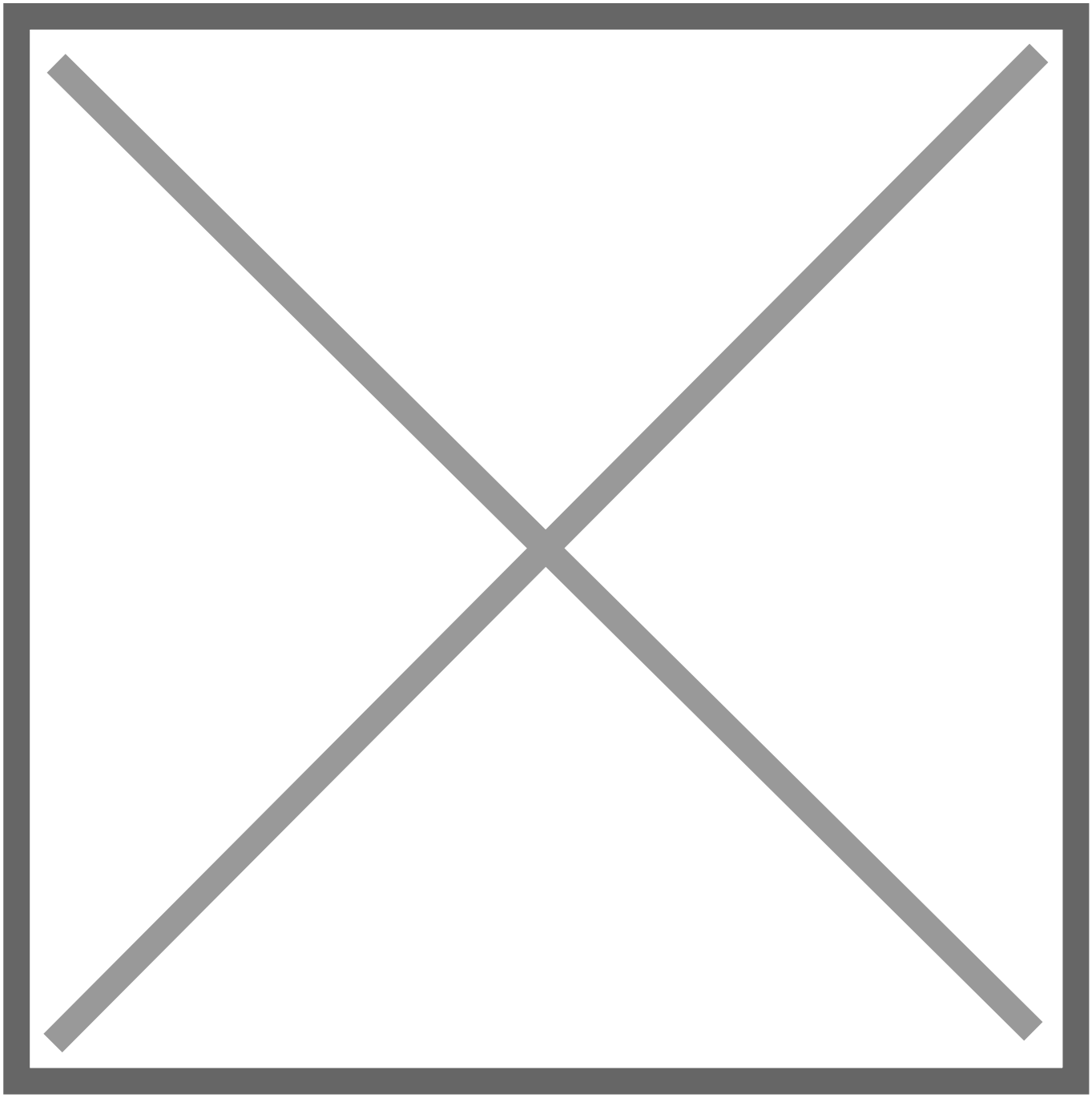


Sysmon eventID=7, загрузка библиотеки процессом. Очень "шумное" правило, следует использовать с крайней осторожностью, чтобы не перегрузить систему. В нашем примере правило выключено(`onmatch="include"` и пустой список.

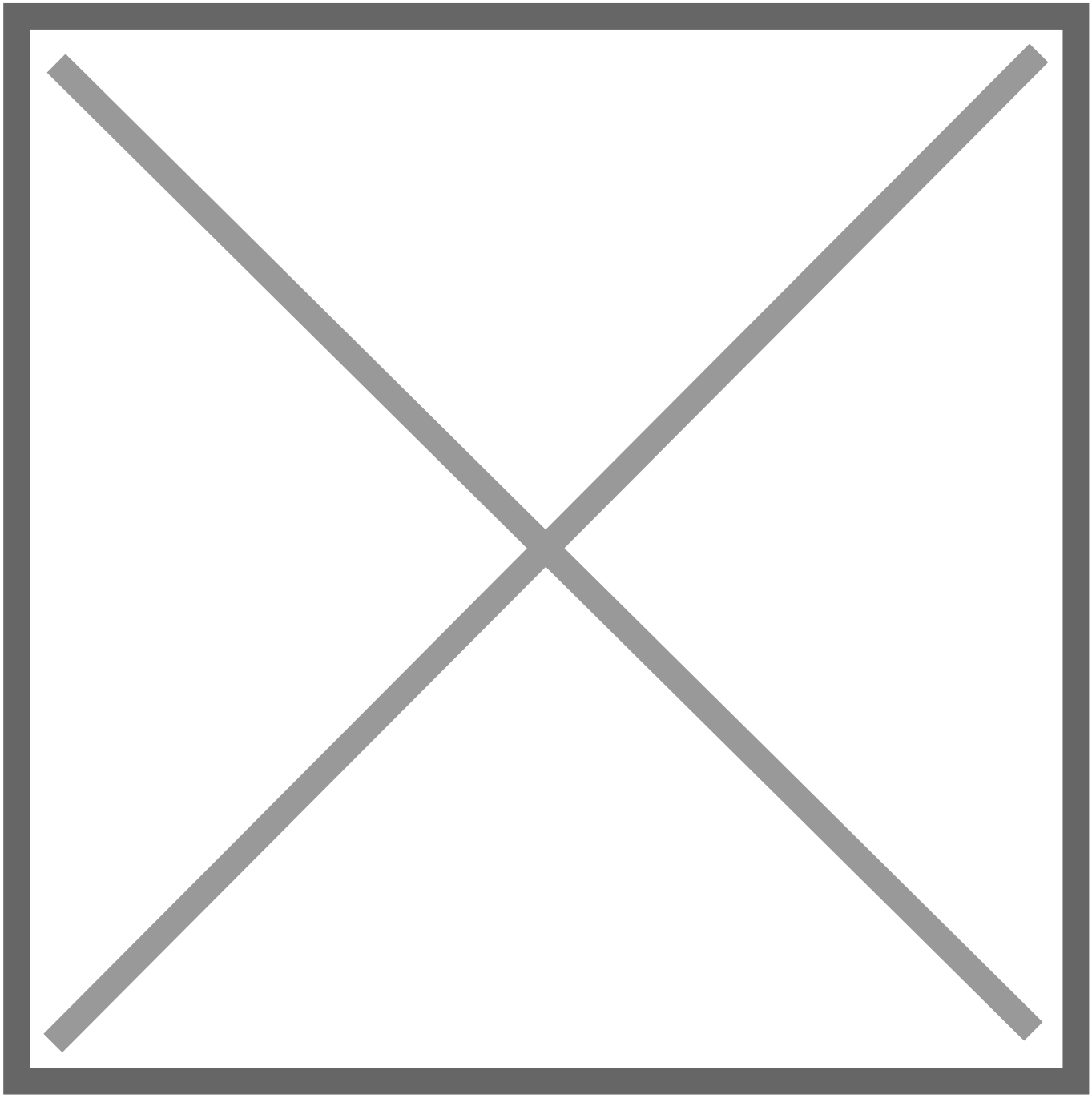


Sysmon eventID=8, создание удаленного потока. Помогает отслеживать process injection, когда один процесс запускает свой код в области памяти другого процесса.

Sysmon eventID=9(`RAW DISK ACCESS`, прямой доступ к диску) и **Sysmon eventID=10** (`NTER-PROCESS ACCESS`, доступ одного процесса к другому) как правило исключаются из мониторинга по причине большого количества событий и их малой полезности.



Sysmon eventID=11 отвечает за создание файлов. Крайне полезное правило, но опять же, очень "шумное". На скриншоте выше мы видим условия, отслеживающие создание файлов в каталоге "Downloads", файлов с определенными расширениями(.bat, .cmd и т.д.) и файлов в меню "Пуск" и автозагрузке.



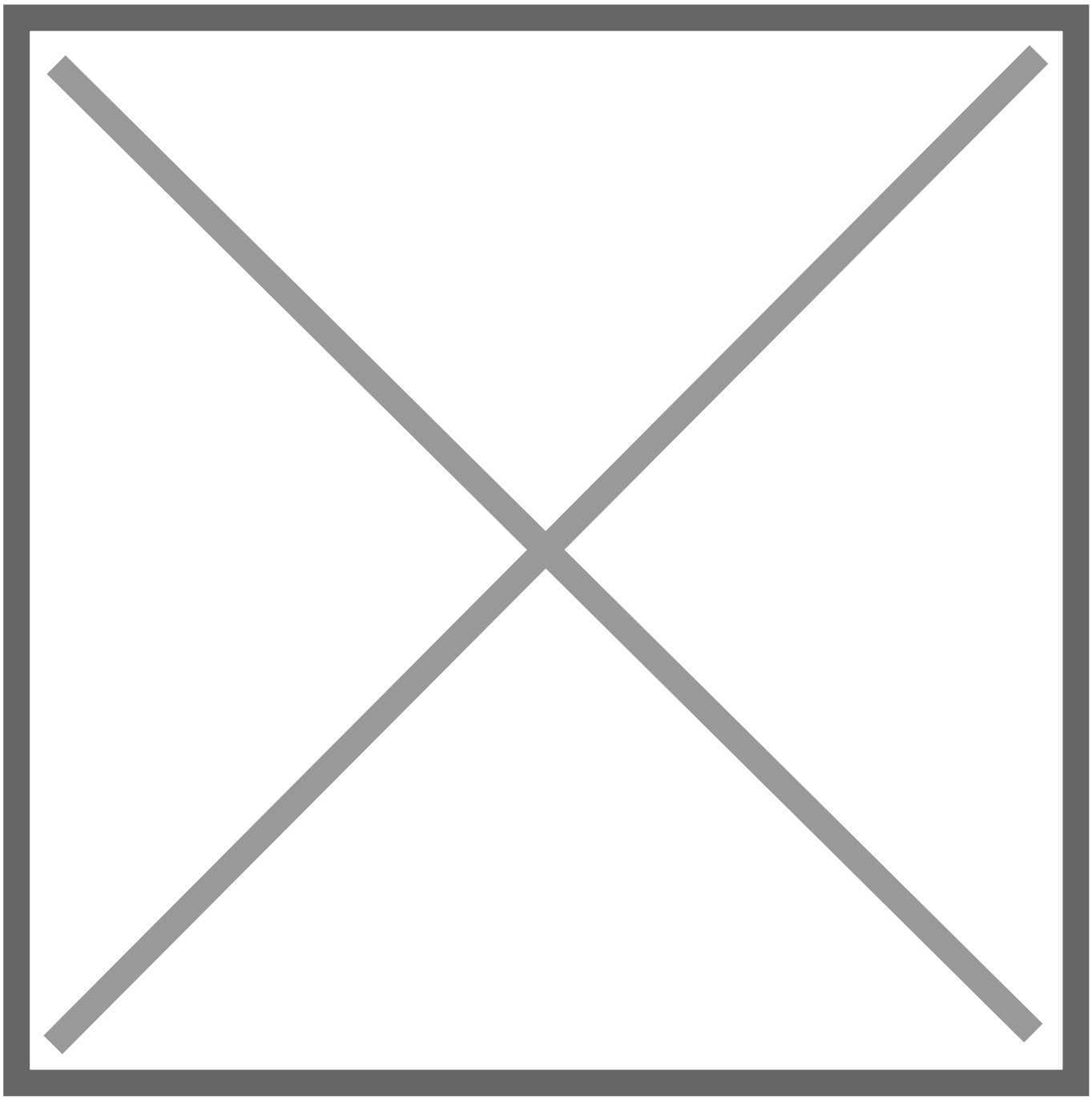
События Sysmon eventID=12,13,14 описывают изменения в реестре Windows:

- **EVENT 12**: "Объект реестра добавлен или удален"
- **EVENT 13**: "Установлено значение"
- **EVENT 14**: "Объект переименован".

На примере выше мы видим правило мониторинга веток Run, отвечающих за автозапуск при старте системы.

- **Sysmon eventID=15** отвечает за мониторинг альтернативных потоков данных (Alternative data streams) в NTFS.
- **Sysmon eventID=16** описывает события изменения конфигурации Sysmon.

- **Sysmon eventID=17**(Pipe Created) и **Sysmon eventID=18**(Pipe Connected) мониторят создание и использование pipes - область памяти используемая для коммуникации между процессами.
- **Sysmon eventID=19, 20** и **21** описывают события, связанные с WMI(Windows management interface).



Sysmon eventID=22 позволяет логировать все DNS-запросы. Эти события могут быть крайне полезными, однако система генерирует огромное количество DNS-запросов каждую секунду.

Начиная с версии 14 Sysmon умеет блокировать исполняемые файлы по хэшу.

Настройка auditd

1. Установим auditd на ОС Ubuntu:

apt install auditd

2. Подготовим или скачаем готовый файл конфигурации:

wget https://raw.githubusercontent.com/Neo23x0/auditd/master/audit.rules

3. Скопируем файл с правилами в каталог /etc/audit/rules.d:

mv audit.rules /etc/audit/rules.d/audit.rules

4. Загрузим правила командой:

augenrules --load

5. Убедимся, что правила добавились с помощью команды:

auditctl -l

6. И проверим, что демон auditd записывает события:

tail /var/log/audit/audit.log

Настройка OSquery

Мы будем использовать OSquery в составе elastic agent. Хотя OSquery можно установить отдельно, интеграция с ELK дает хорошую масштабируемость, позволяя запускать запросы сразу на множестве агентов одновременно. Для работы OSQuery необходим настроенный Fleet server (см. тему 6.2).

Перейдем в раздел Management -> OSQuery.

Нам предложат добавить интеграцию Osquery в имеющуюся политику Fleet Server. Выберем используемую на агентах политику и нажмем "Save and continue".

После того как политика обновится на агентах, можно начинать пользоваться Osquery. Для этого перейдем в раздел Management -> OSQuery -> Live queries и нажмем на "New live query".

Выберем агент(один или несколько), на котором выполним тестовый запрос OSQuery:

SELECT u.username, u.directory, u.uuid, g.groupname, g.group_sid, u.type FROM users AS u JOIN groups AS g USING(gid);

Ниже несколько полезных ресурсов с практическими рекомендациями по OSQuery:

1. https://speakerdeck.com/will03/practical-threat-hunting-with-osquery

2. https://pberba.github.io/security/2021/11/22/linux-threat-hunting-for-persistence-sysmon-auditd-webshell/

3. https://osquery.readthedocs.io/en/latest/

Защита внешнего периметра

DDoS атаки

У крупных компаний со своей AS есть возможность построить BGP peering с ISP, предоставляющим услуги защиты от DDoS. В обычное время этот дополнительный канал никак не используется, и у компании используются ее основные ISP. А во время атаки основные каналы "отключаются" и весь трафик заходит только от ISP оказывающие услуги фильтрации.

Атака на канал
Шаг 1

Коммуникация с ISP с целью ограничения пропускной полосы для протоколов, подверженным Amplification атакам, например 53 (dns), 11211 (memcache), 123 (ntp). Полный перечень протоколов можно изучить тут. Не используемые протоколы можно вообще заблокировать, на оставшиеся установить лимит. Однако, стоит понимать, что если выставить лимиты на UDP, где source port 53 - то под это правило попадут и ответы на DNS запросы, которые исходят, например, из вашего офиса. Т.е. канал и внешние сервисы вы спасете, но работа офиса в Интернете в этот момент может быть затруднена.

Шаг 2

Если вы недостаточно крупный клиент для вашего ISP, для компаний не обладающим свой IP подсетью, которой он может свободно распоряжаться в плане маршрутизации, решение может быть чуть более "интересным". Во-первых, подумайте о возможном размещении своих сервисов в облаках. В этом случае, защитой канала от крупных атак будет уже озадачен непосредственно облачный провайдер, вам же останется решать вопросы защиты только от атак на L7.

Шаг 3

Если по какой-то причине перенести свои сервисы полностью вы не можете, то небольшой шанс остаться защищенным все еще есть — не светите свой настоящий IP в интернете. Нигде. Никакая DNS запись не должна вести на ваш настоящий IP адрес, даже в истории DNS, которую можно найти в открытом доступе в интернете. А ваши офисные сотрудники должны выходить в интернет через NAT IP не имеющим отношения к вашим сервисам. Идея защиты заключается в том, что услугами облачной инфраструктуры мы воспользуемся только с целью проксирования трафика (L3/4).

Проблемой в использовании прокси все еще может стать величина вашего собственного канала в офисе. Если он 1Гбитс, и вас атакуют ровно на эту величину, то лимиты облачного провайдера могут не сработать, и защищаться вам все еще потребуется самостоятельно.

Пример проксирования с помощью iptables

```
iptables -I INPUT -p tcp -m tcp --dport 80 -j ACCEPT
iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination <REMOTE-HOST-IP-ADDRESS>:80
iptables -I INPUT -p udp -m udp --dport 123 -j ACCEPT
iptables -t nat -A PREROUTING -p udp --dport 123 -j DNAT --to-destination <IP-GOES-HERE>:123
iptables -t nat -A POSTROUTING -j MASQUERADE
iptables -I FORWARD -j ACCEPT
iptables -P FORWARD ACCEPT
sysctl net.ipv4.ip_forward=1
```

Шаг 4.1 (TCP)

Для защиты TCP-трафика используем возможности iptables на уровне нашего прокси сервера в облаке. Однако применять фильтры требуется не в FORWARD chian'e в связке с стандартной таблицей "filter", а в PREROUTING chain'e в связке с таблицей "mangle". Это очень сильно снизит нагрузку на вычислительные ресурсы прокси-сервера. Конкретные конструкции по защите от DDoS с помощью iptables можно найти здесь.

Шаг 4.2 (UDP)

Если вы публикуете UDP сервисы (например телефония), то для проксирования и блокировок возможно так же продолжать использовать iptables. Однако сложность SIP трафика заключается в том, что он использует динамические порты. В таком случае потребуется блокировать непосредственно UDP сервисы, подверженные усилению (53, 123, 389, 11211 и т.п.), а также попробовать максимально сузить диапазон динамически выделяемых портов (например 40000-45000) и разрешить пропускать только их.

Шаг 5

Если вы решитесь на такую необычную конфигурацию, располагая прокси сервер в облаке, следует иметь ввиду, что весь трафик вы теперь будете получать только с этого единственного IP адреса. Его и следует разрешить на сетевом оборудовании вашей собственной инфраструктуры, а все остальные прямые подключения запретить.

Однако, перед вами может встать вопрос прозрачности логов. Теперь весь интернет-трафик к вам будет приходить из единственного источника и возможно какая то аналитика, основанная на IP адресах клиентах нарушится. Для веб-приложения, с целью сохранения IP адресов клиентов в логах на уровне прокси стоит воспользоваться Nginx. Проставляя дополнительный заголовок X-Forwarder-For вы сможете сохранить информацию о настоящем

IP источника.

Ограничиваем доступ по IP-адресам

Основываясь на гео IP баз данных, можно разработать автоматизацию, которая 1 раз в день будет обновлять правила iptables либо ACL на вашем сетевом оборудовании и разрешать доступ только из RU региона.

Такая конфигурация отлично работала бы со связкой облачного L3/4-прокси, где на самом прокси вы реализуете доступность по IP гео региона, а на стороне офиса, где непосредственно развернуты сервисы, разрешен входящий трафик только с IP адреса этого прокси сервера.

Интересный реальный пример из жизни, из сферы госзаказов: государственная больница, в одном из многочисленных регионов РФ, имеет множество филиалов, маленьких офисов. Технологический прогресс идет и связью надо оснастить каждый, даже самый маленький филиал, дать возможность даже в самой глубинке пользоваться централизованными сервисами. Организация VPN сети остается на стороне медицинского учреждения, а вот контракты и сами каналы связи предоставляют "сверху", в каждую глубинку идет белый IP адрес. Зачем?

Там, где нет сервисов, нет причины использовать белые IP. Любая неверная конфигурация сетевого оборудования, слабый пароль, использование старых версий прошивки — все это лишний риск ИБ и ИТ специалистам. Даже межофисный VPN легко справляется с подключениями из-за NAT.

Конечно, идеальный сценарий, когда вы знаете конкретные IP адреса клиентов того или иного сервиса и разрешаете доступ только с этих источников, блокируя все остальные. Зачастую это, к сожалению, невозможно. Да и нужды бизнеса могут диктовать необходимость быть открытым всему миру. Однако, и для такого сценария есть рекомендации:

- заблокируйте подключение из Tor-сети;
- заблокируйте подключение из облаков, если вы ожидаете, что вашими сервисами должны пользоваться только люди;
- заблокируйте подключение из источников с плохой репутацией (пример Cisco Talos).

Примеры автоматизаций
iptables: https://github.com/trick77/ipset-blacklist;
CiscoASA: https://gist.github.com/RaceFPV/6f67e7cf4a99dfa3d473de5da325bb0f;
MikroTick: https://github.com/pwlgrzs/Mikrotik-Blacklist.

Помните, что примеры с github лишь примеры. Всегда проверяйте на содержимое скрипты, скаченные из интернета. Стоит так же учитывать, что из-за уникальности вашей собственной инфраструктуры они могут не работать как есть, и может потребоваться доработка.

Ограничиваем по IP в динамике (fail2ban)

В условиях, когда мы не можем заранее предоставить доступ только для доверенных IP, а ожидание обновления репутационных баз может стоить нам драгоценного времени, fail2ban приходит на помощь! Это очень мощный инструмент, который управляет динамическими списками iptables, следуя заранее заданной логике, которую можете создавать вы сами.

Есть множество стандартных профайлов, идущих в комплекте с приложением, которые, например, будут блокировать на 30 минут ip адрес, с которого было совершено 5 неудачных попыток подключения к ssh за последние 10 минут. Такие события логируются в файле /var/log/secure. Т.е. если есть необходимость разработать кастомный профайл — лог файл, это минимальное требование.

Из коробки доступен анализ логов множества приложений, таких как sshd, asterisk, apache, phpmyadmin и другие. Давайте попробуем создать свой собственный темплейт для OpenVPN.

1. Создаем файл /etc/fail2ban/filter.d/openvpn.conf со следующим конфигом:

```
[INCLUDES]
before = common.conf

[Definition]
_daemon = ovpn-server
failregex =%(__prefix_line)s<HOST>:[0-9]{4,5} TLS Auth Error:.*
           %(__prefix_line)s<HOST>:[0-9]{4,5} VERIFY ERROR:.*
           %(__prefix_line)s<HOST>:[0-9]{4,5} TLS Error: TLS handshake failed.*
           %(__prefix_line)sTLS Error: cannot locate HMAC in incoming packet from
\[AF_INET\]<HOST>:[0-9]{4,5}
maxlines = 1
```

2. Создаем файл /etc/fail2ban/jail.d/openvpn.conf со следущим конфигом:

```
[openvpn]
enabled  = true
port     = 1194   ; Change this if your OpenVPN is using a different port
protocol = udp
filter   = openvpn
logpath  = /var/log/openvpn.log  ; Change this if your OpenVPN log path is different
maxretry = 3
```

3. Рестарт службы:

```
systemctl restart fail2ban
```

Условие поиска событий в логе задаем мы. Т.е. fail2ban может быть не только инструментом борьбы против брутфорса, но, например, и от перебора существующих каталогов и страниц на сайте. Если количество строк в логах с кодом ответа 404 превышает, например, 100 за 10 минут — IP в блокировку. Но стоит быть уверенным, что нет ошибок на стороне кода веб-приложения и нет ссылок на несуществующие элементы.

Мы могли бы подсчитывать даже количество валидных GET/POST запросов (код ответа 200). Однако, тут надо быть осторожным, т.к. есть риск заблокировать и настоящих пользователей, которые просто сидят за одним NAT IP адресом в сети большого оператора связи. Важно понимать, что источником данных для принятия решений может быть лог файл любого приложения — и vpn сервер, и почтовый сервер, и сервер телефонии и многие другие.

Глубокая инспекция трафика

Когда блокировка IP адреса недопустима из-за риска блокировать настоящих пользователей, остается принимать решение о блокировке каждого отдельного пакета или веб-запроса.

С одной стороны, к нам на помощь может вновь прийти iptables. Даже, если речь идет о веб-приложении, где весь трафик шифруется (HTTPS), то в связке nginx + apache декрипт проходит на уровне nginx, а apache хостится на localhost (127.0.0.1), т.е. правило iptables может быть применено на этом интерфейсе.

Пример правила iptables с блокировкой, основанной на контенте TCP пакета:

```
iptables -A INPUT -p tcp --dport 8080 -m string --algo kmp --string "../../../../" -j DROP
iptables -A INPUT -p tcp --dport 10556 -m string --algo kmp --hex-string "|3e0000|" -j DROP
```

Очень важное замечание было сделано в первом абзаце данной главы: "пакета или веб-запроса". Если у нас используется система, которая анализирует каждый отдельный пакет, как в разобранном выше примере iptables, то он с легкостью пропустит веб-запрос, содержащий попытку LFI типа GET /script.php?page=../../../../../../../../etc/passwd, если злоумышленник разобьет его на несколько TCP пакетов, в пейлоде которых будет содержаться только один символ. Клиент отправляет 100 пакетов по одному символу, наш фильтр это пропускает, а сервер, используя заложенные алгоритмы в протокол TCP восстанавливает запрос целиком, и готов "ответить" злоумышленнику.

Из-за описанных выше особенностей для гарантированной защиты наших приложений нам потребуются инструментарии типа WAF (для веб-приложений) и IPS (для любых других). Они собирают TCP сессию целиком и только потом принимают решение о блокировке. В третьем видео темы "2.6 Противодействие на периметре" мы описывали особенности работы WAF и быструю установку одного из бесплатных решений. Работа с популярные IPS/IDS такими, как Suricata и Snort была также описана в уроке "2.5 Анализ логов сетевых средств защиты (WAF / NGFW / IDS)".

Крайне редко в своей практике я встречал IPS развернутые в Prevent mode, с целью защиты внешних сервисов. Уж очень большой риск влияния на работу сервиса из-за задержек в анализе, ведь он, кроме того, крайне требователен к ресурсам железа. Поэтому чаще их все-таки ставят сбоку, отливают для анализа копию трафика и это уже получается IDS (Detect mode). Но о том, что такая возможность защиты внешних сервисов существует стоит иметь ввиду.

Hardening сервисов

"hardening" - "укрепление" сервера с точки зрения ИБ.

Например, представьте, что вы используете WordPress CMS на вашем сайте, он хорошо позиционировал в сети, закрыт WAF'ом. Но, к сожалению, уязвимости в нем самом и его модулях обнаруживаются постоянно. И вот, появился очередной "0-day", который начинает активно использоваться злоумышленниками. WAF — не 100% гарант , и в нашем примере он не смог защитить ваш сайт, ваш сервер и злоумышленник реализует RCE (удаленное исполнение кода) на вашем сервере...

Рекомендуют включить и настроить SELinux. Он способен ограничить пользователя www-data, из-под которого работает наш сайт, и не дать ему возможность исполнять код на системе, а лишь читать файлы, что лежат в /var/www/html.

Стандарты помогут вам проверить каждый хост, каждую систему в отдельности, не забыли ли вы чего настроить. Самым распространенным является CIS Benchmark'и. CIS (Center for Internet Security) опубликовал стандарты для множества различных систем:

Кроме стандартов в интернете можно найти и различные автоматизации по их применению и проверке.

Интересную коллекцию подобных автоматизаций вы можете найти по ссылке.

Альтернативы CIS Benchmark

CIS хоть и является дефакто стандартом и самым популярным источником фреймворков, но не всегда в их коллекции можно найти интересующую нас технологию. На помощь, как обычно приходит Google. Будем искать, желательно на официальных сайтах, "hardening" или "best practice" под необходимый продукт. Например:

рекомендации на MikroTik:
https://mum.mikrotik.com/presentations/KH17/presentation_4162_1493374113.pdf
рекомендации на OpenVPN: https://openvpn.net/community-resources/hardening-openvpn-security/
рекомендации на Zabbix:
https://www.zabbix.com/documentation/current/en/manual/installation/requirements/best_practices

Revision #3
Created 17 October 2025 10:33:59 by Admin
Updated 30 October 2025 14:22:36 by Admin