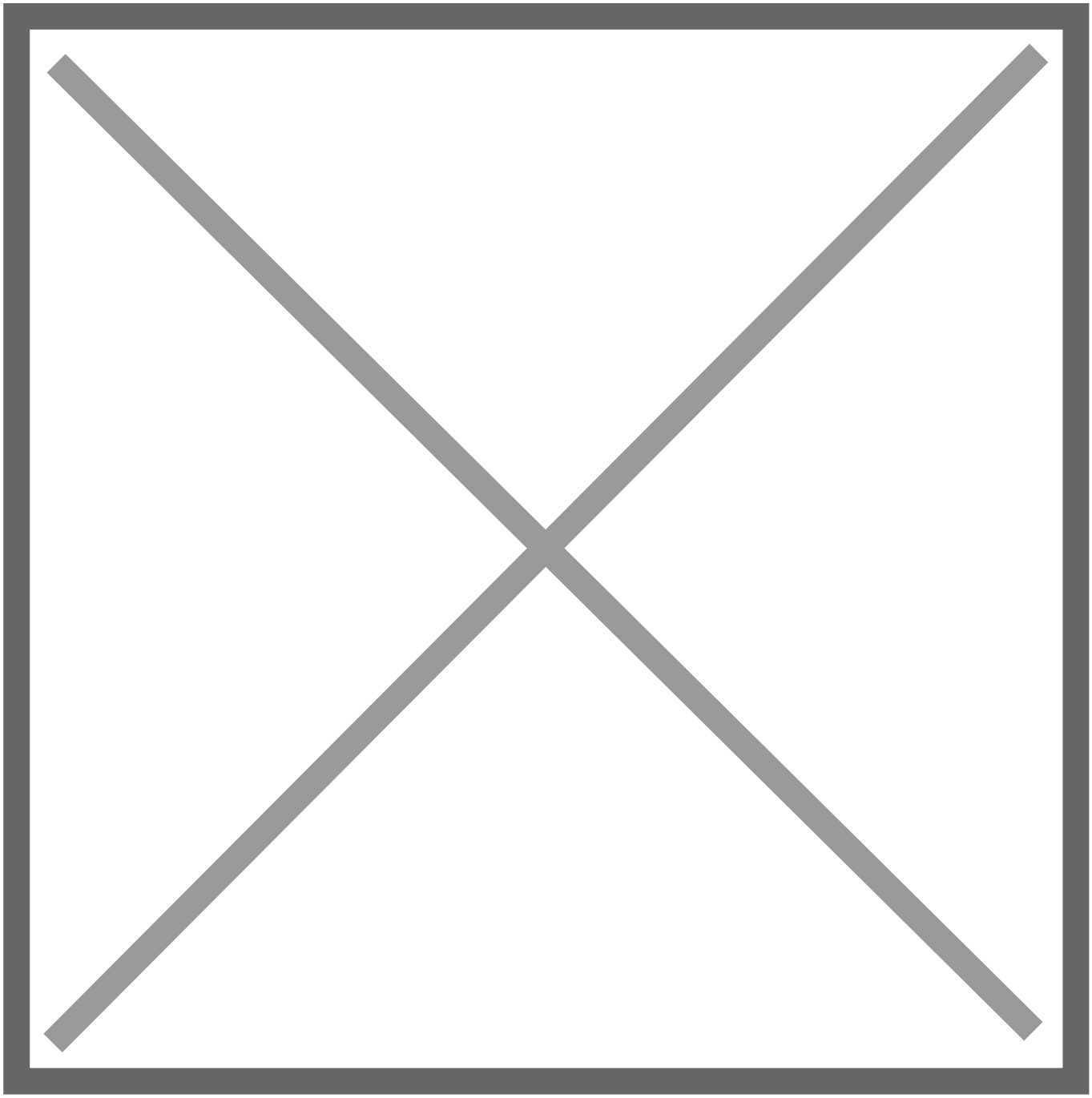
?????????????? ?????????? ??????????????

Уязвимость - любая слабость в системе, например ошибка, которая обеспечивает злоумышленнику физический или цифровой доступ к системе.

Угроза - человек, ситуация или прочее воздействие, способное воспользоваться уязвимостью

Задача отдела ИБ — закрыть уязвимости, соответствующие вашим угрозам, а НЕ победить саму угрозу.

Управление уязвимостями позволяет снижать вероятность компрометации путем их своевременного обнаружения и устранения. Важно определение приоритетов для исправления уязвимостей на основе их серьезности и вероятности эксплуатации. Набор техник направлен на определение и классификацию уязвимостей в системах и приложениях до использования и анализ потенциального воздействия уязвимостей. Включает в себя шаги:



1. Определение активов и требований. Сначала реестр данных, активов и ПО. Определить внешние требования, например соответствие стандартам или сертификациям. Учитывается несколько факторов:

- Требования регуляторов — ФСТЭК, PCI-DSS, SOC2, ISO и прочие.
- Корпоративную политику — компания должна иметь зафиксированные процедуры для обеспечения процесса.
- Классификацию данных — нужно знать, какие данные мы обрабатываем и где эти данные находятся. Также, данные должны быть классифицированы по уровню чувствительности, например публичные, для внутреннего использования и секретные данные.

2. Приоретизация активов. При создании (обновлении) реестра активов, каждому активу проставляется "вес", учитывающийся при подготовке отчета и приоретизации уязвимостей для устранения. Зависит от типа компании. Уязвимости должны устраняться в первую очередь на активах, имеющих больший вес.

3. Поиск уязвимостей. Процесс должен учитывать критичность активов. Учитываемые параметры при настройке ПО:

- Частота сканирования. Нет смысла проверять каждый день, если IT отдел может устранить найденные проблемы в течении квартала. Учитываются требования регуляторов; аппетит бизнеса к рискам; потенциальное нарушение доступности сервисов из-за сканирования; наличие или отсутствие необходимых ресурсов (людей, серверов, времени).
- Аутентификация. Сканеру уязвимостей можно предоставить учетные данные для доступа на проверяемый актив. Проверка уязвимостей с аутентификацией точнее. Сканирование без аутентификации показывает картину глазами атакующего и позволяет точнее определить уязвимости(если они будут подтверждены), которые следует закрывать в первую очередь.
- "Разрушительность" сканирования При настройке сканера как правило можно выбрать инвазивность сканирования (проверять ли DoS, пытаться ли подбирать пароли для учетных записей и пр.). Некоторые проверки могут привести к нарушению работоспособности сервиса или блокировке учетных записей. Уникален для каждой компании.
- Обновление базы уязвимостей и подготовка профиля для сканирования. Например, если в компании не используется Linux, нет смысла искать уязвимости для этой ОС, их можно исключить из профиля сканирования и значительно сократить время и интенсивность проверок.
- Серверное сканирование и сканирование с использованием агентов. Существует опция установки агента на проверяемые машины. Полезно для серверов, находящихся в DMZ, или рабочих станций пользователей, работающих удаленно. Сканирование со стороны сервера, в свою очередь, может выявить новые и/или не задокументированные устройства в сети.

4. Оценка уязвимостей и отчет Например, критическая уязвимость на сервере, не подключенном к интернету и находящемся в бункере будет иметь приоритет ниже, чем подобная уязвимость на публичном веб-сервере. Полученный отчет отправляется владельцу актива для устранения.

5. Исправление уязвимости Владелец актива устраняет уязвимости в отчете (установка обновлений безопасности, отключение уязвимых сервисов, замену устаревшего оборудования). Затем подтверждается, что уязвимость действительно устранена. Сотрудник, ответственный за устранение уязвимости определяет следующие аспекты:

- Порядок устранения — сортировка по критичности, сложности устранения и стоимости устранения.

- В какое время это будет происходить — процесс устранения уязвимостей должен проводиться в соответствии с политикой изменений компании. В процессе устранения уязвимостей сервис может быть какое-то время недоступен, что может повлиять на SLA компании. Само исправление уязвимости может также нести в себе риски для компании, поэтому обычно исправления сначала обкатываются в тестовой среде, и только после этого попадают в продакшн.
- Каким образом это будет происходить — нужно ли нотифицировать об изменении партнеров, менеджмент, соседние отделы, и т.д.

6. Проверка эффективности процесса Для стабильной работы необходимо постоянно оценивать его эффективность (сбор метрик, статистики или аудит). Поиск, адресация и устранение уязвимостей могут генерировать значительную нагрузку на технический персонал, бюджет, и, в зависимости от выстроенных процессов, административную нагрузку на согласование заявок на устранение. Частота/глубина сканов и скорость устранения уязвимостей зависит от множества факторов и будет уникальной для каждой компании.

Единственная универсальная рекомендация — этот процесс должен быть регулярным. Независимо от частоты сканирования (для кого-то и частота сканирований раз в год будет подходящим вариантом). Наличие регулярного процесса помогает заранее планировать время и ресурсы, повышает общую осведомленность сотрудников IT, и позволяет оценивать изменения уровня рисков за наблюдаемый период времени.

Управление уязвимостями внутри периметра

Как правило внутренние сканы бывают следующих видов:

- Discovery scan (сканирование для обнаружения активов) — легкий скан, как правило используются ICMP ping или TCP SYN, предназначенный в первую очередь для обнаружения новых хостов в сети.
- Authentication check — проверка аутентификации сканера на сканируемых машинах. Сканы с аутентификацией дают более точную информацию об уязвимостях, поэтому имеет смысл проверять, может ли сканер успешно авторизоваться на исследуемых узлах сети.
- SCA(Security Configuration Asessment) scan — проверка конфигурации машины на предмет соответствия стандартам или внутренним требованиям.
- Inventory scan — сбор информации об установленном ПО/железе и их версий.
- Specific CVE scan — проверка сети на наличие конкретных уязвимостей. Обычно настраивается профилем сканирования, используется инженерами для ускорения процесса поиска уязвимостей, как правило свежеанонсированных 0-day. Например, после анонса уязвимости в пакете xz utils имеет смысл создать профиль, проверяющий только версию пакета и с помощью быстрого скана оценить количество затронутых этой проблемой хостов.

Управление патчами(заплатками).

Несмотря на то, что термины управления уязвимостями и управления патчами часто используются взаимозаменяемо, это разные, хоть и взаимодополняющие процессы.

Процесс управления патчами (Patch management) отвечает за поиск, тестирование и своевременное применение обновлений безопасности на ИТ оборудовании компании. Отдельно стоит отметить важность тестирования выкатки патчей - ведь вместе с исправлением одних проблем производитель может анонсировать другие, более критичные для бизнеса.

Пассивная проверка на наличие уязвимостей

Еще одним способом проверить машину на наличие уязвимостей является сбор информации об установленных приложениях и проверка этого списка в открытых источниках или специализированных сервисах. Пример сервиса vulners.com:

1. Перейдем на сайт https://vulners.com/scanner/audit и выберем соответствующую ОС(для примера будем использовать Oracle Linux).

2. Скопируем предложенную команду и выполним ее на нашем сервере:

rpm -qa --qf '%{NAME}-%{VERSION}-%{RELEASE}.%{ARCH}\n'

3. Вставим полученный результат обратно на сайт:

4. И после нажатия кнопки "Next" получим результат — список известных уязвимостей в установленых пакетах!

Плюсы

- Проверки подобного рода не нагружают исследуемый сервер и занимают очень мало времени.
- Информацию об установленном ПО можно собирать в одно время(например, раз в день), а проверять на наличие уязвимостей в другое.
- Подобные сервисы как правило поддерживают API, что позволяет автоматизировать проверки.

Минусы

- Информация об уязвимостях может быть не очень корректной. Например, для некоторых linux дистрибутивов секьюрити патчи для приложений не меняют их отображаемую версию, что приводит к выдаче неверных результатов.
- Проверяются только уязвимости, связанные с версиями приложений. По факту, уязвимости могут быть связаны с конфигурацией (тогда такая проверка их не обнаружит), или же наоборот, некоторые уязвимости могут быть устранены с помощью каких-либо настроек(в этом случае мы получим flase positive, ложнопозитивный результат).

Для полноценной интеграции подобных сервисов в процесс управления уязвимостями необходимо эту интеграцию самостоятельно разработать. Выгрузка списка ПО из инвентори, запросы в API сервиса, выгрузка результатов в тикет-систему и прочие детали могут потребовать значительное время на разработку и поддержку автоматизации.

Стандарт CIS benchmark: проверка конфигурации

Для проверки конфигурации ПО на узлах сети используются либо встроенные средства сканера, либо специализированное ПО. Золотым стандартом в индустрии является CIS(Center Internet Security) benchmark. Ознакомиться с ним можно на сайте: https://www.cisecurity.org/cis-benchmarks. Также в интернете можно найти неофициальные скрипты от сторонних разработчиков, например: https://github.com/finalduty/cis-benchmarks-audit.

Нам понадобится сервер с установленным python3. Склонируем репозиторий:

```
# git clone https://github.com/finalduty/cis-benchmarks-audit.git
```

Перейдем в каталог с бенчмарком:

```
# cd cis-benchmarks-audit/
```

И запустим скрипт со следующими параметрами:

```
# python3 ./cis_audit.py --level 1 --server
```

--level 1 определяет уровень "требовательности" к настройкам конфигурации, уровень 1 более требовательный, чем уровень 2;
--server говорит о том, что мы проверяем конфигурацию сервера, а не рабочей станции.

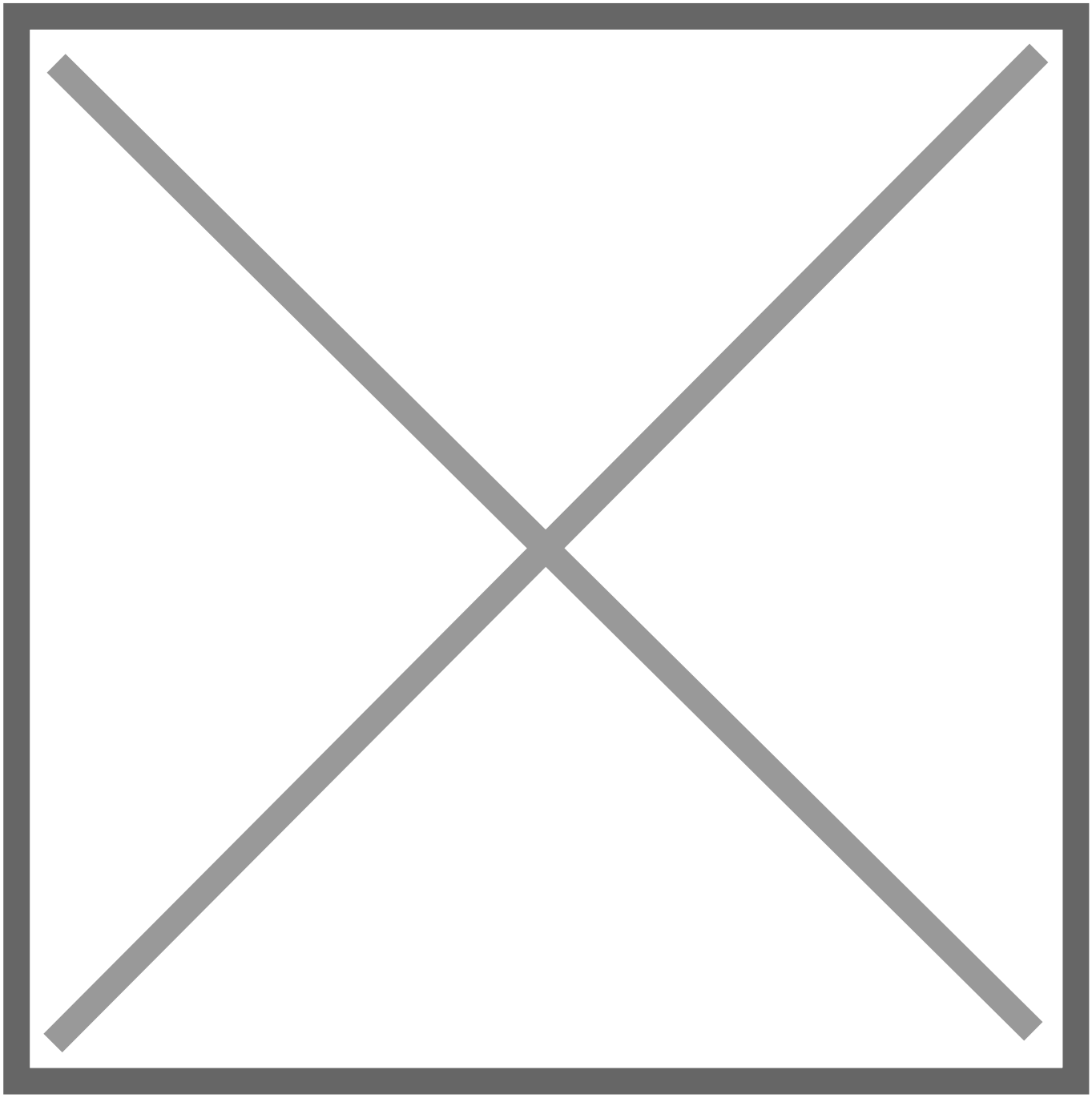
Анализируя результаты необходимо иметь в виду как внешние требования к компании (например, требования соответствия каким-либо стандартам), так и задачи бизнеса.

Например, в некоторых проверках бенчмарк требует, чтобы на сервере не были установлены определенные приложения, например HTTP server. Разумеется, это требование не имеет смысла для серверов, задачей которых как раз является обеспечение работы веб-приложений.

Анализ конфигурационных файлов системы аудита

Программы аудита отслеживают огромное количество событий системы, которые впоследствии можно использовать для отслеживания инцидентов ИБ. Первый вопрос, на который нужно ответить при составлении списка правил для аудита это "Какие есть требования к мониторингу событий? Какие события нам нужно отслеживать?"

Если специфических требований нет, то можно обратиться к матрице MITRE.



Как мы видим, самыми полезными событиями для мониторинга по статистике являются выполнение команд, создание процессов, модификация и создание файлов и события, связанные с сетевым трафиком.

GreenBone Vulnerability Management

GVM состоит из демона Greenbone Vulnerability Manager (gvmd), Greenbone Security Assistant (GSA), Greenbone Security Assistant Daemon (gsad) и сканера OpenVAS. Вариации:

- Greenbone Security Manager (GSM) — коммерческий, в форме физического или виртуального устройства с доступом к базе Greenbone.

- Greenbone Community Edition (GCE) — виртуальная машина, бесплатная пробная версия GSM с менее полным каналом сообщества Greenbone.
- Greenbone Source Edition (GSE) — GVM с открытым исходным кодом. Kali APT включают упакованную версию GSE.

У всех есть Greenbone Security Assistant (GSA) — веб-интерфейс для настройки сканирования и получения информации об уязвимостях.

Настройка GreenBone

Сначала определяется цель (Target) сканирования - совокупность сканируемых хостов. Настраивается через:

- файл CSV с именами хостов или IP-адресами.
- сканирование Host Discovery для обнаружения целей в сети.
- CIDR или IP-адреса вручную.

Digital Forensic исследует и анализирует цифровые устройства и данные для выявления преступных действий, инцидентов безопасности или других аномалий.

Анализ позволяет выявить новые вредоносные объекты или даже техники, пополнив индикаторы компрометации (IOC) и сигнатуры для EDR решений. Таким образом, подобные угрозы будут своевременно предотвращать в будущем.

В случае, если инцидент происходит в данный момент, то по результатам проведенного анализа вы сможете начать противодействие. Первыми простыми шагами противодействия могут оказаться блокировка обнаруженных IP-адресов, доменных имен или вредоносных исполняемых файлов.

Анализ оперативной памяти

Исследование оперативной памяти позволяет увидеть активные процессы, открытые сетевые соединения, пароли, ключи шифрования и многое другое. Сбор дампа оперативной памяти (memory dump) — первый этап анализа оперативной памяти.

Сбор дампа оперативной памяти с ВМ

Техники сбора дампов через средства виртуализации В VMware и Fusion (для MacOS) для получения дампа памяти достаточно поставить ВМ на паузу и скопировать два файла: `.vmem` и `.vmss` из директории ВМ.

Сбор дампа оперативной памяти с "железного" ПК

WinPmem: Старый, открытый исходный код, для Windows. Раньше был в Rekall, сейчас отдельный репозиторий.

DumpIt: упрощенный, Windows и Linux. В Windows объединяет 32-битную и 64-битную память в один выходной файл.

MemDump: бесплатная и простая утилита командной строки

Belkasoft RAM Capturer: мощный, бесплатный. Может захватывать ОП работающего ПК Windows даже при активной защите от отладки или защиты от дампа.

Magnet RAM Capture: бесплатный и простой.

LiME (Linux Memory Extractor): это загружаемый модуль ядра (LKM), прозрачный для целевой системы.

Режим гибернации (сон) в Windows

ОП перед отключением питание сохраняет состояние в файл C:\hiberfil.sys, в нем есть дамп ОП.

Strings Linux дефолтное приложение, пример:

Поиск IP адресов

```
strings win_52a.vmem | grep -E "\b([0-9]{1,3}\.){3}[0-9]{1,3}\b"
```

Поиск почтовых адресов (email)

```
strings win_52a.vmem | grep -oE "\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}\b"
```

Поиск артефактов командной строки

```
strings win_52a.vmem | grep -E "(cmd|powershell|bash)[^\s]+"
```

Volatility

Volatility де-факто главный инструмент анализа ОП.

- volatility 2: https://github.com/volatilityfoundation/volatility
- volatility 3: https://github.com/volatilityfoundation/volatility3

Volatility2 vs Volatility3

- Volatility 2 дамп памяти Windows ранее 10 build 19041.
- Volatility 2 на python2
- В Volatility 2 указывается профайл версии операционной системы для дампа.
- список основных команд в сравнении для версий https://blog.onfvp.com/post/volatility-cheatsheet/.

Используемые модули:

- `pslist`: покажет список запущенных процессов;
- `pstree`: покажет список запущенных процессов, выстроенных в виде дерева родительских отношений;
- `psscan`: поиск процессов, может показать чуть больше чем обычный **`pslist`**;
- `cmdline`: покажет процессы и их аргументы;
- `netscan`: покажет сетевые конекты и связанные с ними процессы;
- `malfind`: покажет процессы, содержащие потенциально зловредный код;
- `svcscan`: покажет список сервисов Windows;
- `dlllist`: покажет список загруженных DLL в указанный процесс.

Вывод списка процессов

```
# vol -f win_52a.vmem windows.pslist
Volatility 3 Framework 2.5.2
Progress:  100.00□□PDB scanning finished
PID□PPID□ImageFileName□Offset(V)□Threads□Handles□SessionId□Wow64□CreateTime□ExitTime□File outpu

4□0□System□0xa80934e81040□120□-□N/A□False□2023-11-07 06:34:32.000000 □N/A□Disabled
92□4□Registry□0xa80934ebd080□4□-□N/A□False□2023-11-07 06:34:25.000000 □N/A□Disabled
316□4□smss.exe□0xa80935b93040□2□-□N/A□False□2023-11-07 06:34:32.000000 □N/A□Disabled
...
```

Вывод дерева процессов

```
# vol -f win_52a.vmem windows.pstree
Volatility 3 Framework 2.5.2
Progress:  100.00□□PDB scanning finished
PID□PPID□ImageFileName□Offset(V)□Threads□Handles□SessionId□Wow64□CreateTime□ExitTime

4□0□System□0xa80934e81040□120□-□N/A□False□2023-11-07 06:34:32.000000 □N/A
* 1676□4□MemCompression□0xa80939902080□26□-□N/A□False□2023-11-07 06:34:41.000000 □N/A
508□428□wininit.exe□0xa8093608e300□1□-□0□False□2023-11-07 06:34:39.000000 □N/A
* 644□508□services.exe□0xa80936ed0180□9□-□0□False□2023-11-07 06:34:39.000000 □N/A
```

```
** 1048 644 svchost.exe 0xa8093b5e8340 4 - 0 False 2023-11-07 06:35:43.000000  N/A
** 1564 644 svchost.exe 0xa809398a5340 3 - 0 False 2023-11-07 06:34:41.000000  N/A
...
```

Сканирование процессов

```
# vol -f win_52a.vmem windows.psscan
Volatility 3 Framework 2.5.2
Progress:  100.00  PDB scanning finished
PID PPID ImageFileName Offset(V) Threads Handles SessionId Wow64 CreateTime ExitTime File outpu

2992 644 svchost.exe 0xa60000081080 8 - 0 False 2023-11-07 06:34:44.000000  N/A Disabled
3224 644 svchost.exe 0xa600001b7080 6 - 0 False 2023-11-07 06:34:44.000000  N/A Disabled
4 0 System 0xa80934e81040 120 - N/A False 2023-11-07 06:34:32.000000  N/A Disabled
92 4 Registry 0xa80934ebd080 4 - N/A False 2023-11-07 06:34:25.000000  N/A Disabled
528 2976 cmd.exe 0xa80934f44080 0 - 0 False 2023-11-07 08:49:34.000000  2023-11-07 08:49:34.000
Disabled
2340 644 spoolsv.exe 0xa8093542f080 9 - 0 False 2023-11-07 06:34:42.000000  N/A Disabled
316 4 smss.exe 0xa80935b93040 2 - N/A False 2023-11-07 06:34:32.000000  N/A Disabled
516 500 csrss.exe 0xa80935fc3080 12 - 1 False 2023-11-07 06:34:39.000000  N/A Disabled
508 428 wininit.exe 0xa8093608e300 1 - 0 False 2023-11-07 06:34:39.000000  N/A Disabled
436 428 csrss.exe 0xa80936c9a080 10 - 0 False 2023-11-07 06:34:38.000000  N/A Disabled
...
```

Сканирование процессов (`psscan`) может показать больше информации, чем обычное отображение списка процессов (`pslist`), если у каких-то процессов использовались техники сокрытия.

Поиск зловредной активности

```
# vol -f win_52a.vmem windows.malfind
Volatility 3 Framework 2.5.2
Progress:  100.00  PDB scanning finished
PID Process Start VPN End VPN Tag Protection CommitCharge PrivateMemory File output Hexdump Dis

...
2216 cybered_beacon 0x1b2017f0000 0x1b20183dfff VadS PAGE_EXECUTE_READWRITE 78 1 Disabled
4d 5a 41 52 55 48 89 e5 MZARUH..
48 81 ec 20 00 00 00 48 H......H
```

```
8d 1d ea ff ff ff 48 89□......H.
df 48 81 c3 cc 60 01 00□.H...`..
ff d3 41 b8 f0 b5 a2 56□..A....V
68 04 00 00 00 5a 48 89□h....ZH.
f9 ff d0 00 00 00 00 00□........
00 00 00 00 f8 00 00 00□........□4d 5a 41 52 55 48 89 e5 48 81 ec 20 00 00 00 48 8d 1d ea ff ff
ff 48 89 df 48 81 c3 cc 60 01 00 ff d3 41 b8 f0 b5 a2 56 68 04 00 00 00 5a 48 89 f9 ff d0 00
00 00 00 00 00 00 00 00 f8 00 00 00
8620□rundll32.exe□0x1f182510000□0x1f18255dfff□VadS□PAGE_EXECUTE_READWRITE□78□1□Disabled□
4d 5a 41 52 55 48 89 e5□MZARUH..
48 81 ec 20 00 00 00 48□H......H
8d 1d ea ff ff ff 48 89□......H.
df 48 81 c3 cc 60 01 00□.H...`..
ff d3 41 b8 f0 b5 a2 56□..A....V
68 04 00 00 00 5a 48 89□h....ZH.
f9 ff d0 00 00 00 00 00□........
00 00 00 00 f8 00 00 00□........□4d 5a 41 52 55 48 89 e5 48 81 ec 20 00 00 00 48 8d 1d ea ff ff
ff 48 89 df 48 81 c3 cc 60 01 00 ff d3 41 b8 f0 b5 a2 56 68 04 00 00 00 5a 48 89 f9 ff d0 00
00 00 00 00 00 00 00 00 f8 00 00 00


...
```

Вывод информации о сетевых взаимодействиях процессов

```
# vol -f win_52a.vmem windows.netscan.NetScan
Volatility 3 Framework 2.5.2
Progress:  100.00□□PDB scanning finished
Offset□Proto□LocalAddr□LocalPort□ForeignAddr□ForeignPort□State□PID□Owner□Created

0xa80935d55050□TCPv4□0.0.0.0□49664□0.0.0.0□0□LISTENING□664□lsass.exe□2023-11-07 06:34:40.000000
0xa80935d555d0□TCPv4□0.0.0.0□49664□0.0.0.0□0□LISTENING□664□lsass.exe□2023-11-07 06:34:40.000000
0xa80935d555d0□TCPv6□::□49664□::□0□LISTENING□664□lsass.exe□2023-11-07 06:34:40.000000
0xa80935d55890□TCPv4□0.0.0.0□135□0.0.0.0□0□LISTENING□892□svchost.exe□2023-11-07 06:34:40.000000
0xa80935d559f0□TCPv4□0.0.0.0□135□0.0.0.0□0□LISTENING□892□svchost.exe□2023-11-07 06:34:40.000000
...
```

Поиск по Yara-правилам

```
# vol -f win_52a.vmem windows.vadyarascan.VadYaraScan --yara-file "./malware_rules.yar"
Volatility 3 Framework 2.5.2
Progress:  100.00□□PDB scanning finished
Offset□PID□Rule□Component□Value

0x18201b6bec9□92□spyeye_plugins□$o□72 64 70 2e 64 6c 6c
0x182020eb214□92□spyeye_plugins□$o□72 64 70 2e 64 6c 6c
0x182020eb494□92□spyeye_plugins□$o□72 64 70 2e 64 6c 6c
0x182025e8fb7□92□spyeye_plugins□$o□72 64 70 2e 64 6c 6c
0x18202e3f53f□92□spyeye_plugins□$o□72 64 70 2e 64 6c 6c
0x18202e4a83f□92□spyeye_plugins□$o□72 64 70 2e 64 6c 6c
0x182031bef0c□92□spyeye_plugins□$o□72 64 70 2e 64 6c 6c
0x182032bbf6f□92□spyeye_plugins□$o□72 64 70 2e 64 6c 6c
0x182032c293f□92□spyeye_plugins□$o□72 64 70 2e 64 6c 6c
0x182033015b4□92□spyeye_plugins□$o□72 64 70 2e 64 6c 6c
0x18204295fec□92□spyeye_plugins□$o□72 64 70 2e 64 6c 6c
0x182071b6479□92□Insta11Strings□$□42 31 32 41 45 38 39 38 2d 44 30 35 36 2d 34 33 37 38 2d 41 3
34 34 2d 36 44 33 39 33 46 45 33 37 39 35 36
0x182072a3a31□92□Insta11Strings□$□45 43 44 34 46 43 34 44 2d 35 32 31 43 2d 31 31 44 30 2d 42 3
39 32 2d 30 30 41 30 43 39 30 33 31 32 45 31
0x182079b80c9□92□Insta11Strings□$□42 31 32 41 45 38 39 38 2d 44 30 35 36 2d 34 33 37 38 2d 41 3
34 34 2d 36 44 33 39 33 46 45 33 37 39 35 36
0x18207a74911□92□Insta11Strings□$□45 43 44 34 46 43 34 44 2d 35 32 31 43 2d 31 31 44 30 2d 42 3
39 32 2d 30 30 41 30 43 39 30 33 31 32 45 31
0x255fc21709c□780□SharedStrings□$m4□00 00 00 00 45 00 52 00 00 00 00 00
...
```

Использование дополнительных плагинов

```
# vol -r pretty -f win_52a.vmem -p /opt/volatility3/volatility3/framework/plugins/
cobaltstrike
Volatility 3 Framework 2.5.2
Formatting...0.00               PDB scanning finished
  |  PID |        Process | Port | Sleep | Jitter |
Server |   POST_PATH |               x86 Install_Path |                x64 Install_Path | Pipe
| License ID
* | 2216 | cybered_beacon |  443 | 60000 |      0 |              cybered-
c2.gopherz.ru,/activity | /submit.php | %windir%\syswow64\rundll32.exe |
```

```
%windir%\sysnative\rundll32.exe |      | 1580103824
* | 8620 |   rundll32.exe |  443 | 60000 |      0 | cybered-
c2.gopherz.ru,/IE9CompatViewList.xml | /submit.php | %windir%\syswow64\rundll32.exe |
%windir%\sysnative\rundll32.exe |      | 1580103824
```

В этом примере используется дополнительный плагин volatility 3 для поиска процесса, в рамках которого устанавливается связь с популярным Command&Control системой CobaltStrike.

Возможные ошибки YARA-библиотек

Работая с плагинами `cobaltstrike` и `yarascan`, вы, возможно, встретите ошибки загрузки этих плагинов. Введя флаг `-vvv` в конец команды vol, вы увидите, что приложение ругается на то, что не может выполнить команду `import yara`.

Причина в том, что `cobaltstrike` основан на `yarascan`, а он в свою очередь требует python-модуль `yara` версии 3.8.0 и выше. Поэтому нам надо удалить устаревший модуль `yara` и установить новый `yara-python`. А затем поправить некоторые ссылки на библиотеку.

В итоге, фиксим и проверяем себя так:

```
pip3 uninstall yara
pip3 install yara-python

# python3
Python 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import yara
Failed to import '/usr/lib/libyara.so'



# find / -name libyara.so
/usr/local/lib/python3.10/dist-packages/usr/lib/libyara.so

# ln -s /usr/local/lib/python3.10/dist-packages/usr/lib/libyara.so /usr/lib/libyara.so

# python3
Python 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import yara
```

```
>>> yara.__version__
'4.5.0'
>>>
```

Анализ HDD

Для групп реагирования на инциденты выделяются определенные функции:

- **Анализ структуры файлов**. Возможность перемещаться и видеть иерархию файлов на диске имеет решающее значение. Криминалистические инструменты должны отображать эту структуру, обеспечивая быстрый доступ к определенным файлам, особенно в известных местах подозрительной системы.
- **Hex Viewer**: в те моменты, когда вам требуется поближе познакомиться с вашими данными, просмотр файлов в шестнадцатеричном формате необходим. Эта возможность особенно удобна при работе с такими угрозами, как специализированное вредоносное ПО или уникальные эксплойты.
- **Анализ веб-артефактов**. Учитывая такой большой объем пользовательских данных, связанных с веб-активностью, выбранный инструмент должен эффективно анализировать и представлять эти данные. Это меняет правила игры, когда вы собираете воедино события, которые привели к тому, что пользователь попал на вредоносный веб-сайт.
- **Работа с электронной почты.** Иногда след ведет к внутренним угрозам. Может быть, это мошенник-сотрудник или просто кто-то ошибся. В таких случаях электронные письма часто имеют ключевое значение. Инструмент, который может извлекать и представлять эти данные, упрощает процесс, упрощая соединение точек.
- **Средство просмотра изображений**. Иногда изображения, хранящиеся в системах, могут рассказать собственную историю. Будь то проверка политики или более глубокое погружение, наличие встроенного средства просмотра является благом.
- **Анализ метаданных.** Такие детали, как временные метки создания файлов, хеши и расположение диска, могут иметь неоценимое значение. Рассмотрим сценарий, в котором вы пытаетесь сопоставить время запуска приложения с предупреждением о вредоносном ПО. Такие корреляции могут стать стержнем вашего расследования.

Все перечисленные выше критерии отлично реализованы в бесплатном программном обеспечении **Autopsy**.

Autopsy

Функции: построение временной диаграммы, поиск ключевых слов, извлечение артефактов из Интернета и электронной почты, а также возможность фильтровать результаты на основе известных хешей вредоносных файлов.

Loki / Thor

Основываясь на YARA-правилах, будучи запущенными непосредственно на зараженном ПК, они помогут вам определить подозрительные файлы и процессы.

Loki — бесплатная, но устаревшая и не поддерживаемая версия продукта, написанная на python.

Thor Lite — бесплатная новая версия с 4000 YARA правилами, идущими в комплекте + свои.

Thor — платная версия с 25000 YARA правилами.

Разработчик: https://www.nextron-systems.com/compare-our-scanners/

Revision #9
Created 28 October 2025 13:36:55 by Admin
Updated 30 October 2025 14:10:52 by Admin