Превентивные средства защиты информации

Детективные сами по себе направлены на сбор информации о событии с целью дальнейшей отправки информации специалисту для определения, легитимна ли данная активность.

Наиболее интересны для нас именно **превентивные** способы, направленные непосредственно на недопущение нарушения. Нарушение может представлять собой запуск вредоносного файла, который будет средствами превентивной защиты заблокирован, несанкционированное подключение и многое другое. Превентивная защита направлена на недопущение факта нарушения.

В рамках данной главы основной упор мы сделаем на наиболее популярные способы защиты информации:

1. антивирусные программы,
2. песочницы (*sandbox*).

Средства антивирусной защиты

Средства антивирусной защиты предотвращают деятельность ВПО и выполнение вредоносного кода, могут обнаруживать компьютерные вирусы и восстанавливать заражённые файлы.

Сигнатурный анализ

Анализ на основе имеющихся в базе сигнатур.
Сигнатура представляет собой уникальную для каждого вируса строку, которая однозначно указывает на определённую вредоносную программу. При появлении нового вируса его код анализируется специалистами, и сигнатура заносится в базу антивируса. Этот способ самый эффективный, так как позволяет почти со стопроцентной вероятностью определить наличие известного вируса.
Минус такого способа: при отсутствии в базе данных сигнатуры вируса, данное ВПО не будет определяться сигнатурным способом. Также нередки случаи ошибочного выбора сигнатур, что приводит к ложному срабатыванию.

Эвристический анализ

Если мы говорим про обнаружение неизвестных вирусным базам вредоносных программ, то нам может помочь эвристический анализ. Эвристический анализ включает в себя два направления проверки:

- На виртуальной машине выполняются команды программы с целью моделирования действий подозрительного ПО. Далее анализируются команды по мере их выполнения, отслеживаются подозрительные действия, особенно репликация, перезапись и попытки скрытия файлов. Если что-то из перечня подозрительных действий было выявлено, это помечается как потенциальный вирус. Таким образом, все команды выполняются вне реальной машины, что позволяет не подвергаться лишней опасности.
- Другой подход подразумевает под собой декомпиляцию и анализ содержащегося машинного кода. После декомпиляции код сравнивают с известными вирусами и вирусными действиями. Если код совпадает с кодом вирусов и вирусоподобных действий, файл помечается.
  Эвристический метод анализа более эффективен в вопросе обнаружения ранее неизвестных вирусов, однако только в случае, если в коде обнаруживаются функции известных вирусов и код вирусных функций.
  В итоге

Метод	**+**	**-**
Сигнатурный	Позволяет почти со стопроцентной вероятностью определить наличие известного вируса.	- При отсутствии в базе данных сигнатуры вируса данное ВПО не будет определяться. - Нередки случаи ошибочного выбора сигнатур, что

		приводит к ложным срабатываниям. - Без обновления баз данных антивируса пользователь будет не защищён от наиболее актуальных угроз, даже если они уже были внесены в базу.
Эвристический	- Позволяет почти со стопроцентной вероятностью определить наличие известного вируса. - Более эффективен в вопросе обнаружения ранее неизвестных вирусов.	- Возможность ложных срабатываний на вполне обыкновенные действия. - Не помогает в лечении заражённых файлов.

Также у многих антивирусов есть иные технологии сканирования и обнаружения ВПО, например, анализ местоположения участков кода в ПО и другие. Однако сигнатурный и эвристический — основные методы, использующиеся всеми антивирусными решениями.

Чем отличаются корпоративные антивирусные решения от личных?

Последнее особенно актуально, если вы любите заходить на сайты с «бесплатным» ПО, где вас перед установкой данного «бесплатного» ПО просят выключить антивирус, иначе программа для обхода встроенных средств защиты скачанного вами ПО может не сработать, и вам придётся покупать данное ПО.
А теперь ответьте на вопрос: «Может ли организация позволить себе понести многомиллионные убытки из-за того, что кто-то из сотрудников на рабочее устройство скачал вот такую «бесплатную» программу и установил себе такое ВПО, чем скомпрометировал целый сегмент сети организации?»

Основная разница между личными и корпоративными антивирусными решениями как раз в том, что в корпоративных существуют встроенные решения для реализации контроля за соблюдением норм ИБ на рабочих устройствах сотрудников.

Dr.Web

https://www.drweb.ru/
В продуктах *Dr.Web* для обнаружения и обезвреживания неизвестного вредоносного ПО применяется множество эффективных несигнатурных технологий, сочетание которых позволяет обнаруживать новейшие (неизвестные) угрозы до внесения записи в вирусную базу. Остановимся на некоторых из них.

Технология FLY-CODE — обеспечивает качественную проверку упакованных исполняемых объектов, распаковывает любые (даже нестандартные) упаковщики методом виртуализации исполнения файла, что позволяет обнаружить вирусы, упакованные даже неизвестными антивирусному ПО *Dr.Web* упаковщиками.

Технология Origins Tracing — при сканировании исполняемого файла он рассматривается как некий образец, построенный характерным образом, после чего производится сравнение полученного образца с базой известных вредоносных программ. Технология позволяет с высокой долей вероятности распознавать вирусы, ещё не добавленные в вирусную базу *Dr.Web*.

Технология анализа структурной энтропии — обнаруживает неизвестные угрозы по особенностям расположения участков кода в защищенных криптоупаковщиками проверяемых объектах.

Технология ScriptHeuristic — предотвращает исполнение любых вредоносных скриптов в браузере и *PDF*-документах, не нарушая при этом функциональности легитимных скриптов. Защищает от заражения неизвестными вирусами через веб-браузер. Работает независимо от состояния вирусной базы *Dr.Web* совместно с любыми веб-браузерами.

Технология Dr.Web ShellGuard — закрывает путь в компьютер для эксплойтов — вредоносных объектов, пытающихся использовать уязвимости, в том числе ещё не известные никому, кроме вирусописателей (так

называемые уязвимости «нулевого дня»), с целью получения контроля над атакуемыми приложениями или операционной системой в целом, контролируя запущенные процессы «изнутри».

Традиционный эвристический анализатор — содержит механизмы обнаружения неизвестных вредоносных программ. Работа эвристического анализатора опирается на знания (эвристики) об определённых особенностях (признаках) вирусов — как характерных именно для вирусного кода, так и, наоборот, крайне редко встречающихся в вирусах. Каждый из таких признаков характеризуется своим «весом» — числом, модуль которого определяет важность, серьёзность данного признака, а знак, соответственно, указывает на то, подтверждает он или опровергает гипотезу о возможном наличии неизвестного вируса в анализируемом коде.

Модуль эмуляции исполнения — технология эмуляции исполнения программного кода необходима для обнаружения полиморфных и сложношифрованных вирусов, когда непосредственное применение поиска по контрольным суммам невозможно либо крайне затруднено (из-за невозможности построения надёжных сигнатур). Метод состоит в имитации исполнения анализируемого кода эмулятором — программной моделью процессора (и отчасти компьютера и ОС).

Песочница(sandbox). Что такое, как используется, какие бывают

Песочница представляет собой среду безопасного тестирования. Данное решение используется для запуска подозрительного ПО обособленно от основной системы с целью минимизации рисков заражения. Такая среда позволяет специалистам безопасно изучить работу кода в динамике и по совокупности признаков определить, является ли данное ПО безопасным или нет.

К признакам, по которым специалисты определяют степень доверия к приложению, относятся:

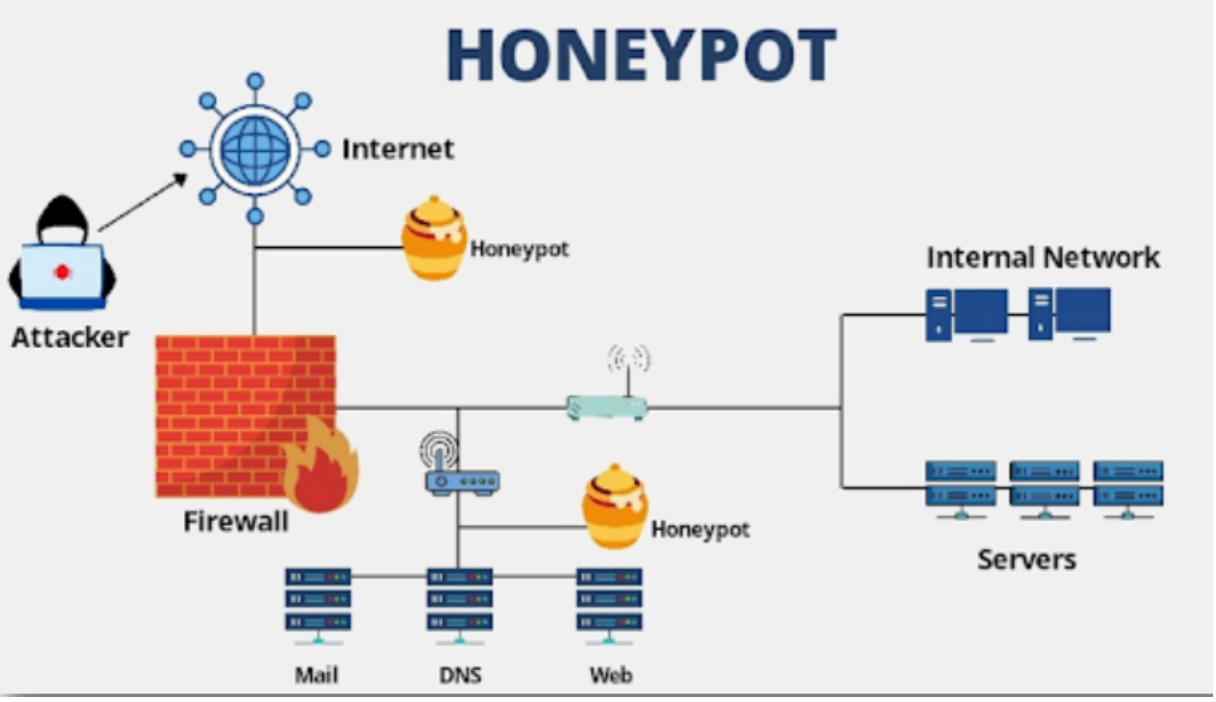
- репликация кода,
- попытки связаться с командным сервером,
- попытки загрузить дополнительное ПО,
- и ещё множество других.

На рынке представлено множество различных решений. Есть как самостоятельные и бесплатные, например, известный в узких кругах *Sandboxie*, так и встроенные в браузеры. Например, в *Google Chrome* применяются технологии песочницы.
Также стоит отметить наличие *API*-песочниц. В данном случае в закрытом пространстве проверяется реакция приложений на различные ответы *API*.

Средства обнаружения ВПО

Honeypot

Начнём с *Honeypot*. Допустим, что злоумышленник скомпрометировал один из хостов нашей сети, и мы не узнали о данном факте. Ситуация распространённая, так как не везде есть нормально настроенные *SIEM*-системы. И при грамотном проведении атаки (в пиковые часы работы компании) следы могут просто затеряться на общем фоне.
Так вот, у нас есть некий скомпрометированный хост, скорее всего, злоумышленники будут дальше пытаться сканировать сеть на предмет уязвимых хостов и расширять свою зону покрытия. Специально для таких случаев и существуют *Honeypot*.



Honeypot представляет собой ловушку, в большинстве случаев специально уязвимую систему с приложениями и данными, которую злоумышленники могут атаковать и тем самым специалисты по информационной безопасности могут как собрать информацию об атакующем, так и узнать о факте наличия потенциально вредоносной активности в сети организации.

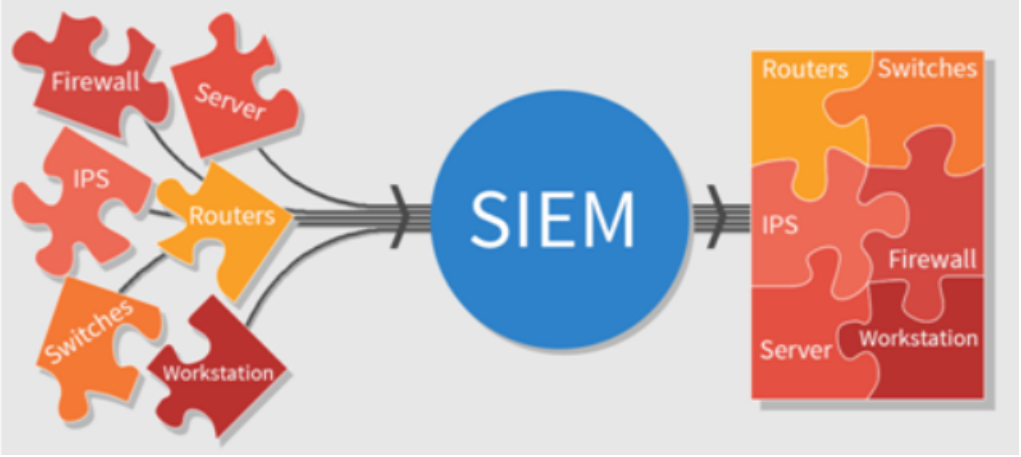
Виды Honepot:

1. Почтовые ловушки
   Специальный электронный адрес где-нибудь в спрятанном месте, где найти его могут только автоматические сборщики электронных адресов. Отправленные на такие адреса письма с наибольшей вероятностью являются спамом, а *IP* можно заносить в чёрный список.
2. Поддельная база данных
   Служит для обнаружения атак с использованием *SQLI*, архитектуры и уязвимостей ПО.
3. Ловушка для вредоносного ПО
   Имитируют приложения, сервисы, *API* с целью поощрения проведения атак с помощью вредоносных программ. Делается это для сбора информации о таких программах, их анализа и устранения уязвимостей.
4. Ловушка для веб-краулеров
   Создаются специальные веб-страницы и ссылки, доступ к которым теоретически может быть только у веб-краулеров (поисковые роботов). С помощью таких ловушек блокируют рекламных поисковых ботов, а также вредоносных ботов.

Есть несколько причин использовать ловушки вместо того, чтобы пытаться обнаружить атаки на настоящую систему. Так, в ловушке по определению не может быть легитимной активности: любые зафиксированные действия, скорее всего, являются попыткой прозондировать систему или взломать её.
Можно легко обнаружить закономерности (например, похожие или происходящие из одной страны *IP*-адреса), свидетельствующие о прочёсывании Сети. Такие признаки атаки легко потерять на фоне обычного интенсивного трафика в вашей опорной сети. Большой плюс ловушек в том, что вы вряд ли зафиксируете что-то, кроме вредоносных адресов, что сильно упрощает обнаружение атаки.

SIEM

SIEM (*Security Information and Event Management*) представляет собой класс программных продуктов, в котором хранится и обрабатывается сетевая активность на наличие событий безопасности.



Основные задачи, выполняемые *SIEM*-системами:

1. Отслеживание сигналов тревоги от приложений и сетевых устройств,
2. Анализ и обработка полученных данных,
3. Оповещение об обнаружении инцидентов,
4. Выявление отклонений поведения.

То есть **задача *SIEM*** — собрать, обработать, найти отклонение и сообщить об этом. В рамках данного класса программных продуктов не стоит задача предпринимать действия по ликвидации обнаруженных инцидентов.

Собирать информацию *SIEM*-решения могут с нескольких источников, таких как:

- контроллеры домена,
- *IDS/IPS*,
- *DLP*,
- системы авторизации и аутентификации.

И так далее. Благодаря широкому спектру источников система с большей вероятностью сможет обнаружить нарушение.
SIEM-системы помогают выявить нарушения политики ИБ в компании, кибератаки и минимизировать ущерб от них. Также полученные *SIEM*-системами данные используются при расследовании инцидентов.

Обход обнаружения ВПО превентивными средствами

Обфускация кода ВПО

Обфускация — приведение кода программы к такому виду, при котором функционал кода остается прежним, однако сильно затрудняется анализ кода и, следовательно, понимание алгоритмов его работы.

Обфускация сама по себе не является чем-то противозаконным и нежелательным. В некоторых случаях благодаря ей удаётся уменьшить размер программного кода.
Методы обфускации в основном работают со следующими моментами в коде программы:

- Данные: преобразуют элементы кода во что-то, чем обозначенные элементы не являются.
- Поток кода: работают с потоком кода таким образом, чтобы он выглядел странным и бессмысленным.
- Структура формата: форматируют данные, переименовывают все, что можно, и так далее.

Пример обфускации кода

Вот список популярных обфускаторов, используемых как в легитимных целях, так и во вредоносном ПО (ВПО), на основе предоставленных результатов поиска:

1. PySimpleGUI

- Используется разработчиками для защиты лицензионных проверок, но может применяться и злоумышленниками для маскировки кода.

2. SmartChart

- Обфускация применяется для усложнения логики кода, что затрудняет анализ. Вредоносные версии могут скрывать вредоносную активность.

3. Obfuscapk

- Популярный обфускатор для Android-приложений, используемый, например, в троянце Zanubis для запутывания кода.

4. Обфускаторы на базе eval/exec

- Часто встречаются в открытых репозиториях (например, GitHub). Злоумышленники используют их для запуска расшифрованного кода через `eval` или `exec` в Python.

5. ASMCrypt

- Криптор/загрузчик, продающийся на подпольных форумах. Маскирует вредоносный код внутри изображений (например, .png) для скрытия загрузки ВПО.

6. DoubleFinger

- Предшественник ASMCrypt, также используется для скрытия вредоносных загрузчиков.

7. Упаковщики и протекторы (например, .NET Reactor, Themida)

- Хотя в результатах поиска явно не упоминаются, подобные инструменты часто используются для обфускации PE-файлов. В курсах по анализу вредоносного ПО (например, от INSECA) рассматриваются техники распаковки таких файлов.

8. Самописные обфускаторы

- Злоумышленники часто создают кастомные методы обфускации, например, внедряя вредоносный код в легитимные библиотеки (как в случае с `aliyun-python-sdk-core`).

Примечание

Многие обфускаторы изначально создаются для защиты интеллектуальной собственности, но злоумышленники адаптируют их для ВПО. Для анализа таких методов полезны инструменты вроде Ghidra, x64dbg, и FLARE-VM.

Если вам нужно больше технических деталей или примеров кода, уточните, какой аспект обфускации вас интересует (Python, PE-файлы, мобильные приложения и т. д.).

Поподробнее о методах обфускации можно прочитать тут
https://cyberleninka.ru/article/n/algoritmy-i-metody-zaschity-programmnogo-koda-na-baze-obfuskatsii/viewer

Но как мы понимаем, обфускация — далеко не универсальное решение. Например, посмотрим на обфускацию *JS* кода в *Metasploit*. В большинстве случаев в обфусцированном коде будут очень длинные и странные названия переменных, а также определённые паттерны в самом коде. *ActiveX*-объекты будут вызываться всегда через `newActiveXObject()`, будут использоваться странные куски кода для сокрытия циклов.
На данные паттерны большое количество *SIEM*-систем, антивирусов спокойно реагируют и оповещают пользователя.

Упаковка кода

Что такое упаковка кода? Упаковка кода подразумевает сжатие файла с дальнейшим добавлением кода, необходимого для распаковки и выполнения.
Существует множество причин для использования упаковки кода: уменьшение места, затруднение обратной разработки (если используется параллельное шифрование).
Создатели вирусного ПО используют упаковку кода с шифрованием для сокрытия вирусного ПО. Таким образом затрудняется обнаружение вируса сигнатурным способом.

1. **Упаковщик с шифрованием** — классическая схема. ВПО упаковывается, после чего вместе с упаковщиком шифруется. Помогает обойти некоторые антивирусные решения, однако далеко не самая эффективная схема ввиду её распространённости.
2. **Множественная упаковка** — ВПО несколько раз упаковывается, используя различные упаковщики кода. Помогает обойти некоторые антивирусные решения.
3. **Использование редких упаковщиков** — ВПО упаковывается новым/редко используемым упаковщиком. Актуальная и серьёзная проблема для создателей антивирусных решений, особенно в аспекте возникновения категории киберпреступников, предоставляющих «упаковщик как сервис». То есть данные группировки занимаются сугубо работой с различными упаковщиками с целью минимизации шансов обнаружения предоставленного им ВПО.

Иные средства предотвращения обнаружения

Из банального — редактирование кода приложения. Актуально для «массовых» вирусов. Немного подправив исходный код, например, убрав неиспользованные функции, можно снизить шанс обнаружения ВПО.

Разница в нескольких строках кода, но этого уже хватило. Основным средством/способом является комбинирование всех указанных выше. Обфусцировать, упаковать несколькими упаковщиками, использовав при этом несколько редких. Таким образом вы сильно увеличите шанс того, что ваш ВПО пройдёт незамеченным.

Способы обхода sandbox

В основном все методы сводятся к одному: обнаружить факт работы в песочнице и перестать работать/не запускаться. Техник обнаружения огромное множество, я приведу типы методов.

1. Проверка запущенных процессоров
   Если мы примерно предполагаем, какие песочницы могут использоваться, то можем попытаться поискать *sandbox* в списке выполняющихся процессов. Пишется специальная функция, которая перечислит все процессы и определит нужный нам по *PID* (идентификатор процесса).
   Например, для поиска *sandboxie* может использоваться такая строка:

```
if(getPIDproc("SbieSvc.exe")) std::cout << "Sandboxie detected!\n";
```

Если найден — то всё, если нет — то вернёт 0 и продолжает выполнение.

2. Проверка подключённых модулей в адресном пространстве
   При помощи *WimAPI*-функции *GetModuleHandler* можно посмотреть модули в адресном пространстве. Если мы предполагаем, какими песочницами нас могут проверять, можно вписать проверку на конкретный модуль.
   Пример для *Sandboxie*:

```
if (checkLoadedDll(L"sbiedll.dll")) std::cout << "Sandboxie detected!\n";
```

А если мы не знаем, какой *sandbox* нас будет проверять или он самописный? Ситуация сложная. Приведём несколько косвенных признаков, по которым можно предположить запуск песочницы. После чего рассмотрим

реальные примеры ВПО и использованные в них средства обнаружения *sandboxie*.

Несколько косвенных признаков, которые могут говорить о запуске в песочнице:

Указатель мышь. В случае запуска программы человеком он так или иначе будет двигать мышкой, она не будет статична. В случае запуска через *sandbox* мышка может оставаться статичной на протяжении всей работы ВПО. Мы можем зафиксировать координаты мышки и проверить их через некоторое время. Если они не изменились, то +1 к тому, что ВПО запущено в *sandbox*.

Количество ядер процессора. Переходим к более неочевидным признакам. Песочница обычно ограничивает себе доступ к ядрам. В большинстве случаев песочница оставляет доступ только к одному ядру. В случае, если при запросе количества “видимых” нашей программой ядер мы получаем ответ, что доступно только одно ядро, то +1 к тому, что ВПО запущено в *sandbox*.

Оперативная память. Когда в последний раз вы держали устройство, у которого 1-2 гигабайта оперативной памяти? Вот именно, большинство современных устройств имеют внутри намного больше оперативной памяти. Если при запросе о доступном объёме оперативной памяти мы получаем результат 1-2 гигабайта, то +1 к тому, что ВПО запущено в *sandbox*.

Свободное место на жёстком диске. С данным признаком всё не настолько очевидно. Если при запросе мы получаем результат, что доступно на жёстком диске меньше 20 ГБ, то это повод задуматься. Однако никто не исключает варианта, что на настоящем устройстве может быть меньше 20 ГБ свободного места, так что данный признак является достаточно спорным. Это +0.5 к тому, что ВПО запущено в *sandbox*.

Перечислим **методы обхода и обнаружения средств виртуализации и анализа**:

1. Отправка запросов для проверки количества ядер процессора. Для успешного продолжения работы должно быть больше 1.
2. Получение информации о версии и производителе *BIOS*.
3. Получение информации об объёме физической памяти. Для успешного продолжения работы должно быть >= 2 900 000 000 байт.
4. Получение информации о количестве запущенных процессов.
5. Проверка наличия отладчика.

Спать в Песочницах

«Песочницы» для вредоносного ПО часто ограничены по времени, чтобы предотвратить перерасход ресурсов, который может значительно увеличить очередь «песочниц». Это важный аспект, которым мы можем воспользоваться: если мы знаем, что «песочница» будет работать только в течение пяти минут, мы можем реализовать таймер сна, который будет работать в течение пяти минут до выполнения нашего шелл-кода. Это можно сделать разными способами; один из распространённых способов — запросить текущее системное время и в параллельном потоке проверить, сколько времени прошло. По истечении пяти минут наша программа может приступить к обычному выполнению.

- https://evasions.checkpoint.com/src/Evasions/techniques/timing.html
- https://www.joesecurity.org/blog/660946897093663167

Обход обнаружения ВПО средствами обнаружения

Обход SIEM

Для себя во время работы с *SIEM*-системами я выделил несколько проблем, которые грамотным злоумышленником могут быть использованы для усложнения нахождения их активности.

Человеческий фактор

За каждым *SIEM* сидит оператор. Оператор — это обычный человек, и ничто человеческое ему не чуждо. Он может пропустить события, связанные с вредоносной активностью (особенно если вы реализуете атаку не через легко обнаруживаемые средства, такие как *Metasploit*).

Однако для того, чтобы повысить шансы, что ВПО не обнаружат, необходимо грамотно выбирать время атаки. Лучше всего подходят часы наибольшей активности в организации (11-16). В таком случае за большим массивом событий безопасности (многие из которых окажутся вполне себе стандартной активностью) ваша атака может затеряться.

Не использовать распространённые инструменты, IOC которых точно известны защищающимся

В организации, где на отделе безопасности не экономят, где собрана хорошая команда специалистов, где операторы активно дают аналитикам фидбек по работе *SIEM*, а аналитики, в свою очередь, на основании фидбека переписывают правила, провести атаку особенно сложно.

К счастью для нас (и к сожалению для безопасников), очень редко можно найти организации, где бы все эти условия были соблюдены. Но что я вам точно скажу: если вы захотите запустить атаку с использованием какого-либо модуля *Metasploit*, вас обнаружат. И либо обрубят доступ, либо будут собирать о вас информацию, и уже потом накроют.

Обход Honeypot

Теперь поговорим о *Honeypot* и о том, как избежать обнаружения. В данном аспекте, к сожалению, у вас как у атакующего меньше всего возможностей. Всё, что я могу вам посоветовать, это быть очень аккуратными при проведении сканирований, при обнаружении слишком уж «дырявых» хостов и/или сервисов. То, что кажется лёгкой целью, в реальности может быть наживкой для злоумышленника.