Оглавление

Часть I. Пример применения уровней безопасности к простому DevOps-конвейеру

Часть II. Выявление аномалий и защита сервисов от атак

Часть III. Усиление безопасности в DevOps

9 Обнаружение вторжений

В этой главе

- Изучение фаз продвижения вторжения по инфраструктуре.
- Обнаружение вторжений с помощью указателей.
- Применение журналов аудита Linux для обнаружения вторжений.
- Удаленное изучение файловых систем, памяти и сети конечных точек.
- Фильтрация исходящего сетевого трафика с помощью систем обнаружения вторжений.
- Роль разработчиков и администраторов в обнаружении вторжений.

На дворе июль 2015 года. Хакер, известный под псевдонимом Phineas Fisher, публикует в Twitter короткое, но ужасающее сообщение: «gamma и HT пали, другие на очереди :)».

Сообщение быстро распространилось по сообществу информационной безопасности. Gamma International и Hacking Team (HT) — это две известные фирмы, работающие в сфере безопасности, которые торгуют пренеприятнейшими технологиями вторжения. Обе они прославились продажей эксплойтов в популярных приложениях по самой высокой цене, что подорвало их репутацию в среде специалистов по безопасности. Phineas взломал Gamma International в 2014-м, поэтому новость о вторжении в еще одну фирму, оказывающую услуги в сфере безопасности, заставила всех нервничать. Действительно ли Phineas вторгся в сеть одной из наиболее параноидальных на планете компаний среди занятых в области безопасности? Сначала люди не верят,

но вскоре Phineas публикует данные с целого почтового сервера компании, развеяв все сомнения о том, что их защита была взломана. Но как?

Спустя несколько месяцев после вторжения Phineas опубликовал подробный отчет, в котором детально объяснил каждый шаг, предпринятый для получения наиболее защищенных данных компании. Сканирование открытых сетевых точек входа не сообщило о явных уязвимостях, но Phineas продолжал изучать сетевое оборудование, используемое в HT, и создал для него эксплойт нулевого дня. В тексте говорится, что на разработку этой атаки потребовалась лишь «пара недель работы». Как только Phineas оказался внутри сети, он распространил бэкдоры и инструменты для исследования, чтобы получать все более широкие права доступа, совершая на своем пути кражи паролей и извлекая данные, пока все секреты HT не оказались у него в руках. Прочтите отчет в http://mng.bz/Ca4t — он действительно на многое открывает глаза.

Мы можем только надеяться, что на нашу голову не обрушится гнев таких решительных хакеров, как Phineas Fisher. Но все равно однажды настанет момент, когда кто-то совершит ошибку и оставит входные данные на отрытом сайте, воспользуется неактуальными программами из Интернета, забудет незаблокированный телефон в баре или поделится паролем со взломанным сайтом.

Рассмотрим меры безопасности, которые вы должны предпринять в определенных местах для того, чтобы выявлять вторжения.

9.1. Семь фаз вторжения: цепочка вторжения

Цепочка вторжения, или килл чейн (kill chain), — это термин, который был предложен компанией Lockheed Martin в статье, опубликованной в 2011 году, для описания серии из семи шагов, предпринимаемых атакующими для вторжения (http://mng.bz/wtdH). Термин произошел из военного жаргона, описывающего захват противника на поле сражения, и был адаптирован для мира технологий. В цепочке вторжения представлено целостное описание фаз вторжения, и это стандартный термин, используемый в индустрии безопасности, поэтому полезно его понимать.

Семь фаз цепочки вторжения

1. *Разведка*. Цель исследуется поверхностно, возможно, с использованием сканеров уязвимостей, таких как ZAP или NMAP, или с помощью изучения сведений из соцсетей, почтовой рассылки и т. д. Это может быть также реальная разведка с посещением здания офиса цели. Фаза сбора информации.
2. *Вооружение*. Разработка атаки на цель, такой как троянский конь, в PDF-документе с логотипом компании или эксплойт в каком-то оборудовании.
3. *Доставка*. Развертывание атаки на жертву. Механизм зависит от цели, самый популярный прием — удаленные сетевые атаки.
4. *Заражение*. Активация атаки на жертву с целью ее взлома. Заражение часто происходит автоматически по вызову пользователя, как с развертыванием троянского

коня при успешном ходе событий, также его может в любое время удаленно запустить атакующий.

5. *Установка.* Как только цель оказывается взломанной, атакующие обычно «заселяются» и начинают развертывание своих инструментов. На этой фазе развертываются бэкдоры, разведчики и другие трояны.
6. *Получение управления.* Большинство атакующих действуют вслепую, пока в жертве не будут установлены их инструменты, которые начнут высылать отчеты. Соединение называется C2-каналом и позволяет атакующему получить контроль над жертвой.
7. *Выполнение действий у жертвы.* Атакующий находится по ту сторону барьера и может продолжить преследовать свои цели, будь то кража данных или получение доступа к другой системе (что называется *боковым перемещением*).

Вы можете просмотреть отчет Phineas о взломе Hacking Team и легко найти каждому действию соответствие в списке цепочки вторжения.

1. *Разведка* — Phineas сканирует сеть HT извне и не находит очевидных проблем, а затем решает заострить свое внимание на сетевом оборудовании.
2. *Вооружение* — Phineas провел инженерный анализ программного обеспечения сетевого оборудования и написал эксплойт для него.
3. *Доставка* — эксплойт был отправлен по сети в общедоступное сетевое устройство.
4. *Заражение* — когда эксплойт достиг цели, он был автоматически запущен.
5. *Установка* — Phineas установил различные оригинальные инструменты на скомпрометированной машине, чтобы продолжить прощупывание сети, оставаясь незамеченным.
6. *Получение управления* — C2-канал — это обратная оболочка, установленная с помощью DNS. DNS используется также для извлечения данных, так как UDP-порт 53 зачастую открыт для корпоративных сетей.
7. *Выполнение действий у жертвы* — вторжение в первоначальную цель — это первая фаза атаки, и Phineas повторяет процесс по цепочке, пока не получит доступ к наиболее защищенным данным компании, для того чтобы извлечь их из ее сети и выложить в Интернете.

Понимание цепочки вторжения позволит вам расположить механизмы обнаружения в правильных местах. Вернемся к четырем уровням обнаружения, размещенным в инфраструктуре, которую вы создали для invoicer в части I (рис. 9.1). Атакующий слева вторгся в инфраструктуру и получил доступ к системе. Первыми бить тревогу начнут журналы аудита системных вызовов в скомпрометированной системе, так как они разворачиваются вместе с эксплойтом. Когда атакующий продолжит

действовать, переходя в фазу установки и загружая инструменты для дальнейшего вторжения, система обнаружения вторжений (IDS) зафиксирует исходящее соединение и поднимет тревогу. При рутинной проверке систем с использованием инструментов для исследования безопасности конечных точек можно выявить также подозрительные файлы и бэкдоры, оставленные в скомпрометированных системах. И наконец, администраторы могут заметить необычное поведение компонента в инфраструктуре, исследовать его и обнаружить вторжение.

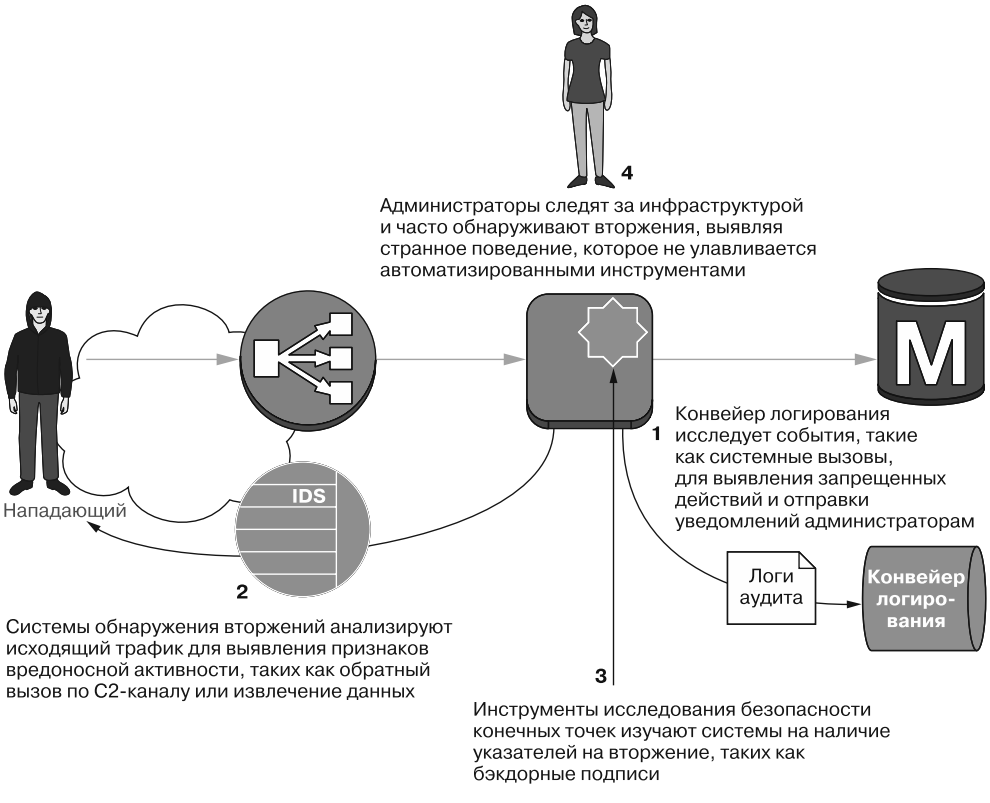

Рис. 9.1. Четыре уровня обнаружения — журналы аудита, IDS, безопасность конечных точек и бдительность администраторов

Мы подробно обсудим все уровни в этой главе, но перед тем, как погрузиться в инструменты, я хочу ознакомить вас с концепцией *указателей на вторжение* (indicators of compromise, IOC) — термином, используемым для обозначения сведений, которые представляют собой признаки атак. В IOC содержится информация, позволяющая вам выявлять вторжения в свою организацию и делиться этой информацией с другими организациями, которые могли оказаться в подобном положении.

9.2. Что такое указатели на вторжение

Даже самые лучшие инструменты не смогут принести пользу без базы данных вредоносной активности, с которой можно сравнивать фиксируемые события. Опытные команды безопасности посвящают много времени созданию таких баз данных (что называется *разведкой угроз*) и ее внедрению в свою инфраструктуру обнаружения вторжений — этот шаг сложно осуществить силами малых изолированных команд. Эксперты в сфере информационной безопасности давно поняли, что распространение информации очень важно для защиты их собственного окружения, и пытались стандартизировать процесс ее распространения.

Указатели на вторжение — это способ, которым эксперты в области безопасности делятся информацией о вредоносной активности. Существует множество типов IOC, в частности:

- MD5 или SHA256-хеши вредоносных программ или бэкдоров;
- IP-адреса C2-каналов или узлов атаки;
- домены, связанные с атаками;
- ключи реестра в системах Windows, созданные или модифицированные вредоносными программами;
- байтовые строки, расположенные во вредоносных программах, которые можно найти на диске или в памяти.

В чем-то IOC подобны сигнатурам антивирусов. Основное отличие — то, что они предназначены для широкого распространения, в то время как производители антивирусов жадно прячут содержание своих баз данных. К тому же IOC не ограничиваются описанием вредоносных программ или вирусов и могут содержать паттерны фишинг-атак или IP-адреса, связанные с атаками типа «отказ в обслуживании» (denial-of-service, DoS). Термин *«указатель на вторжение»* связан скорее с принципом совместного использования исследований об угрозах, чем с конкретным объектом.

Команды по безопасности из разных организаций могут обмениваться IOC для увеличения зоны покрытия своих систем обнаружения. Правительственные агентства и фирмы, предоставляющие услуги в области безопасности, тоже часто публикуют IOC для того, чтобы помочь организациям защититься от реальных угроз. Группа быстрого реагирования на компьютерные инциденты Соединенных Штатов (US-CERT), например, регулярно издает аналитические отчеты о вредоносной активности, которые содержат IOC (вы можете прочесть один из них на https://securing-devops.com/us-cert-grizzly.pdf). Команды по безопасности читают эти отчеты и используют предоставленные IOC, чтобы проверить наличие потенциального вторжения в собственных средах.

В следующих разделах мы рассмотрим некоторые распространенные форматы IOC: Snort Talos, Yara, OpenIOC и CybOX.

9.2.1. Правила Snort

Snort — это самая старая сеть IDS, которая используется по сей день. В течение почти двух десятилетий со времени создания в 1998 году Мартином Рэшем (Martin Roesch) Snort использовали для защиты границ сетей. Позднее администраторы по безопасности осознали важность распространения информации в своем кругу для увеличения производительности Snort-систем, и, чтобы эффективно это реализовать, был создан формат правил. Формат Snort-правил все еще популярен. Другие IDS-продукты, такие как Suricata, поддерживают их, так что публикация правил Snort вместе с аналитическими отчетами — обычное явление.

Правило Snort описывает вредоносную активность на сетевом уровне. В листинге 9.1 приведен пример правила, предназначенного для выявления активности бэкдора Dagger. Оно состоит из четырех частей.

- Первая строка в правиле описывает его действие (`alert`), согласно которому сгенерируется уведомление в случае нахождения соответствий правилу. Другие действия могут журналировать активность или полностью аннулировать соединение.
- Вторая строка описывает сетевой протокол (`tcp`) и параметр соединения. Для того чтобы соответствовать этому правилу, соединение должно исходить из домашней сети во внешнюю (в большинстве случаев в Интернет), а также иметь порт-источник 2589 и порт назначения.
- Третья часть — это дополнительные возможности для правила. Здесь вы найдете сообщение `msg`, которое можно добавить в уведомление `alert`, журнал, запускаемый правилом, а также информацию, которая поможет организовать и классифицировать правила (`metadata`, `classtype`, `sid` и `rev`).
- И наконец, четвертая часть правила содержит параметры, используемые для нахождения соединений, активность которых соответствует поведению бэкдора Dagger. Параметр `flow` описывает то, к какой части цепи соединения применяется

Листинг 9.1. Правило Snort для обнаружения сетевой активности бэкдора Dagger

```
alert                                   ← Действием правила отправляется уведомление
tcp $HOME_NET 2589 -> $EXTERNAL_NET any ( ← Соответствие протоколу
    msg:       "MALWARE-BACKDOOR — Dagger_1.4.0";
    metadata:  ruleset community;
    classtype: misc-activity;           ← Дополнительные варианты классификации правила
    sid:       105;
    rev:       14;

    flow:      to_client,established;   ← Параметры нахождения соответствий в полезной нагрузке
    content:   "2|00 00 00 06 00 00 00|Drives|24 00|";
    depth:     16;
)
```

правило, — здесь оно используется, начиная с ответов сервера до клиента. Параметр `content` содержит бинарные и ASCII-строки, которые будут применяться для выявления вредоносных пакетов с помощью нахождения соответствий в полезной нагрузке пакета. А параметр `depth` устанавливает ограничение на то, насколько глубоко правило должно погружаться ради поиска соответствий. Здесь поиск ограничен первыми 16 байтами в каждой полезной нагрузке.

В начале 2000-х годов правила Snort были стандартным методом защиты сетей от распространения вирусов. Они используются даже сегодня, но, как мы выясним позднее, могут быть сложны в развертывании в среде IaaS, где администраторы не управляют сетями.

Эти правила ограничены также выявлением подозрительной активности в сетевом трафике, и их не получится использовать для описания вредоносных файлов в системах. Следующий формат, который мы обсудим, сосредоточен как раз на этой задаче.

9.2.2. Yara

Yara — это и инструмент, и формат IOC, предназначенный для идентификации и классификации вредоносных программ. Он был создан Виктором Альварезом из VirusTotal для организации и распространения информации среди аналитиков. В листинге 9.2 показан пример файла Yara для Linux-руткита. Документ разделен на три части.

- Раздел *«мета»* содержит информацию об IOC, такую как имя автора, время создания или ссылка на подробную документацию.
- Раздел *«строк»* содержит три строки — одну в шестнадцатеричном формате и две в формате ASCII, которые идентифицируют руткит.
- В разделе *«условий»* к исследуемым файлам применяется фильтр, чтобы найти соответствующие особым критериям. В этом примере сначала производится поиск по заголовку файла, который соответствует формату ELF (uint32(0) == 0x464c457f), а затем поиск общедоступного выходного файла (uint8(16) == 0x0003) типа ELF. ELF расшифровывается как *исполняемый и связываемый формат* (Executable and Linkable Format) и является форматом для исполняемых файлов Unix-систем. Если для обоих условий найдены соответствия, Yara станет искать строки, указанные ранее. Если все они будут присутствовать в файле, то это будет расцениваться как обнаружение руткита.

Консольный инструмент Yara способен сканировать целые системы на наличие файлов, которые соответствуют сигнатурам вредоносных файлов, применив команду `Yara -r rulefile.yar /path/to/scan`. Проект правил Yara собирает IOC, выявленные аналитиками безопасности во время расследований, и делает их широкодоступными (http://mng.bz/ySua). Это прекрасное место, с которого можно начать работать с Yara и сканировать системы на наличие IOC.

Yara сосредоточен на файловых IOC. Он обеспечивает мощным и продуманным интерфейсом для сканирования файловых систем, только вот не все IOC содержатся

в файлах. Другие форматы IOC, такие как OpenIOC, способны искать указатели, которые не основываются на файлах.

Листинг 9.2. Правило Yara для выявления руткита Umbreon

```
rule crime_linux_umbreon : rootkit
{
    meta:                                   <-- Раздел «мета» с описанием правила
        description = "Catches Umbreon rootkit"
        reference = "http://blog.trendmicro.com/trendlabs-security-
    intelligence/pokemon-themed-umbreon-linux-rootkit-hits-x86-arm-systems"
        author = "Fernando Merces, FTR, Trend Micro"
        date = "2016-08"

    strings:                                <-- Строки для идентификации руткита
        $ = { 75 6e 66 75 63 6b 5f 6c 69 6e 6b 6d 61 70 }
        $ = "unhide.rb" ascii fullword
        $ = "rkit" ascii fullword

    condition:                              <-- Условия, которым должен соответствовать файл, чтобы определяться как руткит
        uint32(0) == 0x464c457f // Generic ELF header
        and uint8(16) == 0x0003 // Shared object file
        and all of them
}
```

9.2.3. OpenIOC

OpenIOC — это формат, созданный Mandiant (теперь FireEye) для управления их инструментами безопасности для конечных точек. Mandiant обрели известность после публикации в 2013 году нашумевшего отчета APT1 (http://mng.bz/0RKL), в котором раскрывалась активность китайского военного подразделения, спонсируемого государством, задачей которого было совершать хакерские атаки на международные корпорации, в большинстве своем американские и европейские. Некоторые IOC, опубликованные в формате OpenIOC, были представлены вместе с отчетом, чтобы позволить командам по безопасности по всему миру проверить собственные среды на наличие вторжения.

В отличие от IOC Yara, OpenIOC использует XML, что делает эти документы почти нечитабельными для неподготовленного взгляда. В листинге 9.3 приведен пример документа IOC, который ищет бэкдор Sourface, нацеленный на Windows-системы. Это лишь отрывок файла, полную версию вы найдете на https://securing-devops.com/ch09/openioc.

Если вы какое-то время будете разглядывать данный документ, то можете начать понимать структуру этого формата. В первой части указаны метаданные с уникальными идентификаторами, автором и датой. Любопытная информация находится в разделе `<definition>`. Раздел начинается с элемента `Indicator`, который объявляет оператор `OR`, что означает: любой элемент `IndicatorItem`, который последует за ним, будет указывать на наличие соответствия (для оператора `AND` потребуется, чтобы все элементы `IndicatorItem` имели соответствия).

Листинг 9.3. Отрывок из документа OpenIOC для бэкдора Sourface

```
<?xml version='1.0' encoding='UTF-8'?>
<ioc
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://schemas.mandiant.com/2010/ioc"
  id="e1cbf7ca-4938-4d3c-a7e6-3ff966516191"
  last-modified="2014-10-21T13:08:41Z">

  <short_description>SOURFACE (REPORT)</short_description>
  <description>SOURFACE is a downloader that obtains a second-stage
  backdoor from a C2 server.  Over time the downloader has evolved
  and the newer versions, usually compiled with the DLL name
  'coreshell.dll'.  These variants are distinct from the older versions
  so we refer to it as SOURFACE/CORESHELL or simply CORESHELL.
  </description>
  <authored_by>FireEye</authored_by>
  <authored_date>2014-10-16T20:58:21Z</authored_date>

  <definition>
    <Indicator id="e16e6299-f75b..." operator="OR">
         <IndicatorItem id="590-7df8..." condition="is">
           <Context document="PortItem"
                    search="PortItem/remoteIP" type="mir"/>
           <Content type="IP">70.85.221.10</Content>
         </IndicatorItem>

       <IndicatorItem id="5ea9f200-01f1..." condition="is">
         <Context document="FileItem"
                  search="FileItem/Md5sum" type="mir"/>
         <Content type="md5">8c4fa713c5e2b009114adda758adc445</Content>
       </IndicatorItem>

       <IndicatorItem id="3f83ca5b-9a2c..." condition="is">
         <Context document="ProcessItem"
                  search="ProcessItem/SectionList/MemorySection/Name"
                  type="mir"/>
         <Content type="string">Local Settings\Application Data\conhost.dll
         </Content>
       </IndicatorItem>

    </Indicator>
  </definition>
</ioc>
```

XML-схема для IOC

Метаданные, описывающие IOC

Проверяет наличие соединения системы с вредоносным IP-адресом

Проверяет наличие вредоносного файла с помощью контрольной суммы MD5

Проверяет наличие вредоносного процесса, запущенного локально

Затем в разделе `Indicator` определяются три элемента `IndicatorItem`.

- Первый элемент, `PortItem`, проверяет, соединен ли удаленный IP-адрес 70.85.221.10 с системой.
- Второй элемент, `FileItem`, проверяет присутствие файла с контрольной суммой MD5 8c4fa713... на диске, что потребует вычисления контрольных сумм MD5 всех файлов на диске, чтобы сравнить их с контрольной суммой вредоносного файла.

- Третий элемент, `ProcessItem`, исследуя память, ищет библиотеку `conhost.dll`, загруженную в запущенном процессе.

OpenIOC — формат не очень красивый, но мощный. Mandiant определили сотни условий для поиска указателей в различных частях операционной системы. Несмотря на то что они нацелены на Windows-системы (инструменты, предоставляемые Mandiant, такие как Redline и MIR, работают только на Windows), OpenIOC можно использовать для того, чтобы делиться указателями на вторжение с другими типами систем.

Исследователи часто распространяют IOC в этом формате, но Yara постепенно становится стандартом в индустрии, вероятно, из-за простоты написания правил Yara по сравнению со сложностью формата OpenIOC с XML. Но OpenIOC все еще играет важную роль в распространении указателей среди участников сообществ из-за своей способности делиться не только сигнатурами файлов.

Последний формат, который мы обсудим, — это STIX. Он подобен OpenIOC своей выразительностью, но стремится быть более читабельным форматом и преследует цель стать стандартом для распространения IOC.

9.2.4. STIX и TAXII

Структурированные информационные сообщения об угрозах (Structured Threat Information eXpression, STIX) — это инициатива, поддерживаемая техническим комитетом разведки киберугроз OASIS, предназначенная для стандартизации анализа угроз, спецификации IOC, процессов реагирования на вторжения и распространения информации среди организаций. В отличие от ранее обсуждаемых форматов, которые были сосредоточены на спецификации IOC, STIX стремится упорядочить весь процесс защиты организаций от атак.

В STIX имеются два протокола: CybOX (Cyber Observable eXpression — выражения для киберпризнаков) — формат IOC-документа, подобный OpenIOC, и TAXII (Trusted Automated eXchange of Indicator Information — доверенный автоматизированный обмен информацией об указателях) — протокол, основанный на HTTP и предназначенный для распространения информации среди участников сети STIX. Протокол TAXII выделяется особенно, так как решает проблему распространения и открытия IOC. Многие годы специалисты по безопасности создавали собственные инструменты и составляли списки ресурсов для сбора новых IOC и передачи их в инфраструктуру обнаружения. Но с TAXII процесс автоматизировался в соответствии со стандартами, которые поддерживают многие организации и поставщики продуктов для безопасности.

Кто угодно может присоединиться к обмену по протоколу TAXII и получить IOC в формате STIX. Листинги 9.4 и 9.5 демонстрируют запрос обмена на hailataxii.com по протоколу TAXII с клиентом под названием cabby (http://mng.bz/xuEA), содержащимся в Docker-контейнере. В листинге 9.4 запрашивается служба обнаружения элементов обмена, в результате чего возвращается список коллекций, содержащих IOC, из другого источника. Вывод в примере показывает лишь одну коллекцию, принадлежащую `EmergingThreats`, а полный список будет насчитывать десятки результатов.

Листинг 9.4. Запрос доступных коллекций из TAXII-обмена на hailataxii.com

```
$ docker run --rm=true eclecticiq/cabby:latest
taxii-collections
--path http://hailataxii.com/taxii-discovery-service
--username guest --password guest

=== Data Collection Information ===
  Collection Name: guest.EmergingThreats_rules
  Collection Type: DATA_FEED
  Available: True
  Collection Description: guest.EmergingThreats_rules
  Supported Content:   urn:stix.mitre.org:xml:1.0
  === Polling Service Instance ===
    Poll Protocol: urn:taxii.mitre.org:protocol:https:1.0
    Poll Address: http://hailataxii.com/taxii-data
    Message Binding: urn:taxii.mitre.org:message:xml:1.1
================================
```

Команда Docker для получения данных от сервиса обнаружения с помощью клиента cabby

Метаданные из коллекции, обнаруженной с помощью сервиса taxii

Сервис обнаружения возвращает каждой из коллекций имя, которое можно передать опрашивающим командам для скачивания полного списка IOC в формате STIX, содержащихся в этой коллекции. В листинге 9.5 показано, как клиент cabby используется для скачивания этих IOC. Из-за невероятной насыщенности XML-документов STIX здесь показан лишь один сокращенный IOC, а некоторые дополнительные поля удалены.

Листинг 9.5. Получение IP-адреса IOC в формате STIX из TAXII-обмена

```
$ docker run --rm=true eclecticiq/cabby:latest taxii-poll \
--path http://hailataxii.com/taxii-data \
--collection guest.EmergingThreats_rules \
--username guest --password guest

<stix:STIX_Package id="edge:Package-96b-38-4d-8f-8f" version="1.1.1"
     timestamp="2017-03-06T17:21:19.863954+00:00">
  <stix:Observables cybox_major_version="2" cybox_minor_version="1"
   cybox_update_version="0">
   <cybox:Observable id="opensource:Observable-6-8-4-7-16b"
    sighting_count="1">
    <cybox:Title>IP: 64.15.77.71</cybox:Title>
    <cybox:Object id="opensource:Address-a5-0-4-b-372">
      <cybox:Properties xsi:type="AddressObj:AddressObjectType"
       category="ipv4-addr" is_destination="true">
        <AddressObj:Address_Value condition="Equal">
         64.15.77.71
        </AddressObj:Address_Value>
       </cybox:Properties>
     </cybox:Object>
   </cybox:Observable>
 </stix:Observables>
</stix:STIX_Package>
```

IP-адрес, который расценивается IOC как вредоносный

Рациональное использование дискового пространства, очевидно, не является первостепенной целью формата STIX (да и всего, что основано на XML): распространение одного четырехбайтового адреса IPv4 потребует его оформления в 4000 байт XML-файла. Кроме этого, STIX и TAXII являются открытыми стандартами, реализованными в небольшом количестве открытых (http://mng.bz/U0ZK) и коммерческих проектов (http://mng.bz/2E8R), и сейчас они представляют собой наилучший способ обмена указателями IOC.

Сейчас, когда я пишу эту книгу, еще рано говорить о том, получат ли STIX и TAXII широкое применение. Вторая версия спецификаций значительно их упростила и ввела использование формата JSON вместо XML (листинг 9.6), и, возможно, ее будет легче поддерживать в различных инструментах для безопасности. Следите за этими проектами. Они окажутся полезными, когда ваша организация достигнет того уровня зрелости, который позволит вам обмениваться данными разведки угроз с другими.

Листинг 9.6. IOC-документ в STIX v2 в формате JSON для бэкдора Poison Ivy

```
{
  "type": "indicator",
  "id": "indicator--a932fcc6-e032-176c-126f-cb970a5a1ade",
  "labels": [
    "file-hash-watchlist"
  ],
  "name": "File hash for Poison Ivy variant",
  "pattern": "[file:hashes:sha256 = 'ef537f25c895bfa...']",
},
```

Хеш SHA256 для файла бэкдора

А пока этот момент еще не настал, вам стоит сосредоточиться на своих исследовательских возможностях. Теперь, когда мы обсудили назначение и форматы IOC, настало время для того, чтобы узнать, как сканировать инфраструктуру на их наличие. В следующем разделе начнем анализировать системы с помощью инструментов для исследования безопасности конечных точек.

9.3. Сканирование конечных точек на наличие IOC

Выявление скомпрометированной системы в вашей инфраструктуре повлечет за собой параноидальную спираль, которая остановится лишь после проверки всех систем на отсутствие признаков вторжения. Я видел, как команды по безопасности изобретали всякого рода приемы для реализации этого задания, начиная со сложного bash-скрипта, переданного в команду `parallel -ssh`, которая соединяется с сотнями систем, и заканчивая оригинальными исполняемыми файлами, завернутыми в манифест puppet, которые возвращают результаты через системные журналы. Фантазия инженеров, которые ищут способы запуска произвольного кода на сотнях систем, не знает границ, но давайте признаем, что эти решения — так себе.