Эссе на тему

Honeypots

Подготовил

студент 917 группы

Афанасьев Илья Валерьевич

1. Предисловие.

В связи с лавинообразно растущим количеством пользователей Internet в наши дни самым острым образом встает вопрос безопасного использования этой глобальной сети. Огромное количество важной и зачастую секретной информации находится без должной защиты от различных теневых организаций. Вся защита до текущего момента была основана на предупреждении или принятии контрмер против наиболее распространенных типов сетевых атак. Естественно что при появлении очередного нового метода атаки, любая защищающаяся сторона оказывается безоружной. Новая технология “Honeypots” помогает в какой-то степени решить прблему защиты информации.

2. Технология «Honeypots».

2.1. Общие понятия.

Honeypot - это некоторый ресурс, представляющий собой «ложную цель» для атакующей стороны. Изначально предполагается, что honeypot должен быть взломан. Главная задача honeypot'а – отвлечение внимания атакующей стороны от реальной цели и получение как можно более полной информации об атакующих, а так же методах и ресурсах, используемых при атаке. При всем этом получение описанной информации должно происходить незаметно для атакующей стороны.

Honeypot может быть как системой, которая эмулирует другие системы или приложения, так и какой-либо стандартной системой. Обычно honeypot устанавливается на специально предназначенном для этого сервере, не содержащем никакой полезной информации. Естественно при этом, что любой поток информации проходящий через этот сервер является несанкцианированным действием атакующей стороны. Благодаря относительной «пустоте» сервера все действия атакующей стороны легко контролируемы и, как правило,
Постоянно регистрируются с целью дальнейшего изучения.

2.2. Классификация honeypots.

Имеются две больших категории honeypot'ов.

Первая категория – так называемые пассивные, или анализирующие honeypot'ы. Они не защищают пользователей от нежелательных атак, однако накапливают исчерпывающую информацию обо всех действиях атакующей стороны. Эта информация используется для предупреждения и избежания последующих атак а так же для разработки новых методов защиты и устранения уязвимостей систем защищающейся стороны.

Вторая категория - активные honeypot'ы, непосредственно помогающие избежать атак. Honeypot'ы не являются прямым средством защиты сетей. Наоборот, по задумке они должны привлекать к себе большое количество взломщиков.

По уровню сложности honeypot'ы делятся соответсвенно на два класса: *низкого* и *высокого*.

Honeypot'ы первой категории обычно представляют собой набор нескольких эмулированных приложений или функций. При несанкцианированном проникновении такой honeypot просто «наблюдает» за действиями атакующего. Операционной системы, с которой атакующий мог бы работать, такой honeypot не имеет и риск взлома этого honeypot'а

минимален, так как у атакующего нет способов получить реальный доступ к серверу. Впрочем, минимально и получаемое от него (honeypot'а) количество полезной информации. Создание таких honeypot'ов не занимает много времени, их цель - обнаружение простейших атак и идентификация атакующей стороны.

Honeypot'ы высокого уровня, напротив, способны накопить большое количество полезной информации об атакующей стороне, так как предоставляют реальную функциональность и обычно имеют операционную систему. Однако реальная функциональность дается лишь ценой реальной уязвимости. Опасность взлома таких honeypot'ов обусловлена тем, что взломщик тем или иным образом может получить доступ к операционной системе honeypot'а. В случае если атака подобного рода пройдет незамеченной, атакующий сможет использовать взломанный honeypot в своих интересах, что является крайне нежелательным для защищающейся стороны. Создание honeypot'ов этой категории - занятие крайне ресурсоемкое, так как требует усилий не только на создание операционной системы, но и на создание средств защиты такой системы.

Имеется еще один класс honeypot'ов - очень высокой сложности. Этот класс представляет собой объединение нескольких honeypot'ов с различными средствами сетевой защиты и называется по-другому honeynet. До недавнего времени использовались только простейшие honeypot'ы, способные работать на одном персональном компютере. Honeynet'ы призваны увеличить производительность honeypot'ов. С помощью таких систем появилась возможность эмулировать реалистичное сетевое окружение, правда ценой довольно больших затрат на администрирование и техническую поддержку. К примеру, регистрационные файлы honeynet'ов обрабатываются гораздо сложнее, чем аналогичные файлы одного honeypot'а. Естественно, что опасность взлома honeynet значительно увеличивается даже по сравнению с honeypot'ами высокого уровня сложности.

2.3. Наиболее распространенные типы honeypots.

В настоящее время имеется несколько свободно и коммерчески распространяемых honeypot'ов, различающихся своей сложностью и функциональностью. Упомянем наиболее яркие:

- ManTrap (производитель - Resourse Technologies) - Honeypot средней сложности, работающий на ОС Solaris.
- Deception Toolkit (автор - Fred Cohen) - Несколько скриптов, написанных на языке perl, эмулирующих

некоторые сервисные функции, работающие как honeypot низкого уровня сложности

- Specter (производитель - NeoWorx) – honeypot низкого уровня сложности, работающий на различных ОС

3. Сбор информации с honeypot.

Получаемую honeypot'ом информацию можно условно разделить на две части:

- собираемая информация об атакующем (его IP адрес, используемые при атаке утилиты, частота атак, и т.п.);

- информация, позволяющая администратору «заглянуть» в удаленную систему (например, получить информацию о текущем использовании процессора, и т.п.). Сервер с установленным на нем honeypot'ом здесь и далее будет также называться honeypot'ом.

Методов получения такой информации от honeypot'а два: прямо на хосте с honeypot'ом – host-based, и удаленное получение – network-based.

3.1 Host-based.

Возникает вопрос о том где хранить полученные об атакующем данные. Конечно, можно хранить эти данные прямо на honeypot'е, например на какой-нибудь скрытой партиции, откуда позже эти данные забирать. Этот подход имеет множество недостатков. Во-первых, эти данные не могут быть проанализированны в момент атаки, так как для этого пришлось бы создавать дополнительные специфические утилиты, по которым атакующий смог бы понять, что работает с honeypot'ом. Во-вторых, на honeypot'е может просто закончиться свободное для хранения данных место, в результате чего атакующий вообще перестает быть наблюдаемым. И в-третьих, атакующий может каким-либо образом узнать о расположении собираемых данных на honeypot'е и переделать их по своему усмотрению. В этом случае атакующий так же перестает быть наблюдаемым. В силу перечисленных причин получаемые honeypot'ом данные должны храниться в недоступном для атакующего месте. Для этого используются любые не характерные для передачи данных через интернет устройства, такие как, например, обычный принтер (хотя, конечно, напечатанную информацию анализировать намного сложнее чем электронную).

Получение информации о текущем состоянии honeypot'а задача более сложная, так как при этом нет никакой специальной аппаратуры для

мониторинга honeypot’ов, а даже если бы она была, это могло бы поставить под угрозу секретность honeypot’а.

3.1.1. Сбор информации с помощью распространенных ОС.

Microsoft Windows сейчас это наиболее распространенная операционная система, используемая многими организациями. Поэтому идеальным вариантом было бы внедрять так называемые логгеры (средства записи происходящего с системой на файл или поток) в ядро Windows. Эта задача осложняется тем, что исходные тексты Microsoft Windows не являются свободно распространяемыми и какие-либо изменения в ядро Windows внести крайне сложно (если вообще возможно).

Проще обстоит дело с UNIX. Здесь логгеры действительно можно присоединять к системе, благодаря ее открытости. Правда, опытный атакующий может отличить обычный UNIX от UNIX со встроенным honeypot’ом. Например, сравнить подозрительные системные файлы с оригинальными, проверить библиотечные зависимости (скажем, команда grep вдруг начинает зависеть от демона syslog,и т.п.), или просто пронаблюдать работу функций (и заметить, что grep обращается к демону syslog). Еще хуже будет, если этот атакующий перепишет свои системные файлы UNIX поверх существующих, из-за чего встроенный ранее в один из файлов «надсмотрщик» вообще пропадет. Бороться с этим достаточно просто – встраивать honeypot’ы прямо в ядро UNIX, где обнаружить их уже практически невозможно. Опять же, опытный атакующий может свести все наши старания на нет путем установки нового ядра.

3.2. Network-based.

Получение информации host-based всегда происходит на конкретном компьютере, ставя его под угрозу атаки. Network-based тип получения информации не обязан происходить на самом honeypot’е. Кроме того, процесс может происходить незаметно, так как поток информации проходящий через сеть только анализируется и при этом остается неизменным. Такое получение информации проходит безопаснее, его сложнее обнаружить и уж тем более остановить. Получение может происходить, например, с помощью firewall, если её настроить таким образом, чтобы она записывала весь проходящий через нее трафик.

4. Расположение honeypot в сети.

Рассмотрим модель «internet-firewall-intranet». Атакующая сторона может находиться как за firewall, в глобальной сети, так и перед ней (в локальной сети), например, в сети нашей организации. Как

расположить honeypot'ы так, чтобы они работали одинаково эффективно для обоих случаев?

Honeypot'у не требуется какая-либо особая поддержка и он может быть установлен, по большому счету, где угодно. Располагая honeypot' перед firewall, мы не влияем на защищенность нашей внутренней сети и можем наблюдать или ловить атакующих из глобальной сети. Однако сделать что-либо с атакующими из локальной сети гораздо сложнее, т.к. firewall будет ограничивать трафик идущий от атакующего к honeypot'у. Аналогично, расположив honeypot в локальной сети, мы сможем контролировать атакующих внутри нее и не сможем контрролировать атакующих за пределами firewall. Таким образом, расположение honeypot'а зависит от того, кого мы хотим скомпроментировать.

5. Недостатки honeypot.

Самый большой недостаток – ограниченное поле зрения honeypot'ов. Атакующий может взломать на нашем сервере все что ему заблагорассудится, и при этом если он не будет атаковать наш honeypot, последний не подаст сигнала тревоги.

Еще один недостаток – стандартное поведение honeypot'ов, по причине того что они в основном эмулируют что-либо. Если атакующий несколько раз наткнется на один и тот же honeypot, есть большая вероятность того, что он его рассекретит.

И наконец, в случае, когда honeypot будет взломан атакующей стороной, в дальнейшем он будет использован против его изначальных создателей.

Из-за перечисленных недостатков honeypot не может заменить других систем безопасности, таких как firewall или IDS(Intrusion Detection System).

6. Пример honeypot: ManTrap.

ManTrap – общедоступный коммерческий продукт, созданный компанией Resource Technologies. Он работает на ОС Solaris 2.6, 2.7 и 2.8. Поддерживает архитектуры платформ Sun-Sparc и Intel-x86.

Так как разработчики ManTrap утверждают, что в среднем атаки из внутренней внутри сети организации приносят гораздо больший урон самой организации, нежели отаки извне, ManTrap главным образом направлен на безопасность внутренних сетей организаций, хотя и может быть использован для контроля внешней сети.

Идея ManTrap основана на такназываемых «клетках» (cages). Клетка - это копия операционной системы компьютера (сервера), находящегося в сети. На одной машине ManTrap может поддерживать до четырех таких операционных систем. Во время установки рабочая среда внутри клетки создается такой же, как на хосте. Все клетки взаимодействуют с ядром хоста через такназываемый kernel wrapper (средство создания системной оболочки для стандартизации внешних обращений и изменения функциональной ориентации действующей системы). Таким образом в сети создается эмуляция четырех операционных систем, каждая из которых имеет свой собственный сетевой интерфейс.

Так как ManTrap использует реальную ОС, он может создавать эмуляции любого software и заполнять клетки любыми приложениями, которые поддерживаются ОС хоста. Все приложения, запускаемые в пользовательском пространстве могут запускаться и внутри клетки, с одним ограничением – ManTrap суживает возможности приложений, работающих напрямую с ядром ОС. Клетки заполняются с помощью «генератора стандартного содержимого» (CGM, Content Generation Module). Такое содержимое генерируется случайным образом, вследствие чего все созданные клетки являются уникальными (попарно не совпадают). Помимо этого CGM предусматривает несколько уровней «изменяемости» с тем чтобы не дать атакующему определить факт попадения в honeypot, даже в том случае, если у самого атакующего имеется копия системы ManTrap. Модуль CGM генерирует следующие данные:

- e-mail сообщения создаваемые по шаблонам (шаблоны могут содержать имена общеизвестных членов организации, название организации, интернет-домен организации, и т.п.);
- домашние директории для каждого пользователя, использующего ManTrap.

Уникальным ключом к каждой копии ManTrap является начальное число, которым инициализировался генератор случайных чисел (random seed), и таким образом содержимое, созданное каждым ManTrap, отличается от данных, созданных любым другим ManTrap. ManTrap может администрироваться с центрального клиента, имеющего GUI Java.

Существенная активность в клетке (запуск всеразличных процессов, обращение к файлам, и т.п.) регистрируется. Регистрационные файлы (log-файлы) могут скачиваться с помощью syslog на удаленный сервер. ManTrap предоставляет графический анализ событий (которые берутся из log-файлов) происходящих с сервером. ManTrap использует систему цифровой подписи для подтверждения подлинности создаваемых log-файлов.

Для сбора Сетевых пакетов, идущих к хосту, на котором установлен ManTrap, или от него, последний использует модуль проверки текущего состояния сети (network sniffer). Network sniffer просматривает трафик и получает информацию, относящуюся к клеткам ManTrap или хосту. Потоковый модуль регистрирует всю считанную или записанную в течение сеанса работы с терминалом информацию. С помощью отслеживания информации идущей от сеансов работы с псевдотерминалами логгер может следить за несколькими разными пользователями, работающими с системой одновременно. Помимо регистрации сетевой информации и информации полученной от работы с псевдотерминалами, ManTrap использует дополнительное ядро с тем чтобы из таблицы процессов получить исчерпывающую информацию обо всех процессах, запущенных ядром.

Система сигнализации у ManTrap может быть настроена так, чтобы посылать предупреждающие сообщения, основанные на специфических классах событий. Соответсвенно, сигналы тревоги, поступающие от ManTrap, могут быть различной степени важности.

***Использованная литература*:**

Reto Baumann, Christian Plattner - *White Paper: Honeypots.*

Project Honeynet Members. *Project Honeynet*.
http: //project.honeynet.org

Open Source Honeypots: Learning with HoneyD.
http: //www.securityfocus.com/infocus/1659.html

Lance Spitzner. *Honeypots – Definitions and Value of Honeypots*.
http: //www.enteract.com/˜lspitz/honeypot.html