# RootKit RFC, part 1
# Рекомендации по созданию програмных закладок.
# Аналитическая статья.
# Version 1.1 .

NetHack club teamwork

8 января 2019 г.

# Содержание

...
Уходят волки в оптике прицела..
И все про все - твой выстрел на удачу..
...
группа Би 2.

...
Игра ума..
Кончается расстрелом.
И здесь и там..
Все та же волчья стая.
...
группа Би 2.

# 1 Обзор статьи.

## 1.1 Насколько это законно?

В разных странах существуют различные законодательные ограничения вплоть до уголовного преследования разработки програмного обеспечения вредоносного характера. Данная статья не содержит програмного кода и является аналитическим обзором, таким образом ее развитие не подлежит преследованию, по крайней мере, согласно законодательству РФ. Если Вы считаете иначе - присылайте maintainer'у Ваши комментарии с ссылками на соответствующие статьи законодательства.

## 1.2 Использование.

Изначально статья распространялась исключительно в узком кругу заинтересованных лиц - специалистов IT, в особенности среди тех из них, чья работа связана с обеспечением безопасности IT инфраструктуры, либо с ее тестированием.

Теперь же статья выложена в public domain в надежде на поддержку сообщества.

Email текущего релизера документа в рамках NetHack club: grey-olli@ya.ru, PGP ключ доступен к поиску в интернет: gpg –search-keys grey_olli , сервер ключей: hkp://keys.gnupg.net

Подразумевается использование в качестве ознакомительного материала для:

1. Управляющего персонала среднего и высшего звена (главы: 1, 12 )
2. Персонала ответственного за безопасность сетевой ( internet(IPv4/IPv6), ЛВС (ethernet / wifi LANs / Corporate WANs) ) инфраструктуры. (особое внимание главам 3 и 4).

3. Заказчика практической реализации описываемых в данной статье принципов - стоит просмотреть все уделив особое внимание главам 3 (со стр. 13), 6 (со стр. 46), 12 (со стр. 56) .

4. программиста или группы программистов, реализующих закладку класса «rootkit» для сетей общего пользования, в частности, интернет (особенное внимание главе 11 ).

5. Любых заинтересованных лиц - как обзорный материал по возможным способам скрытия и обнаружения закладок, использующих сети общего доступа, в частности, интернет (если нет конкретной цели и достаточно времени, то лучше просмотреть все ).

Разумеется, если Вы заинтересовались, есть смысл прочесть все.

**Глоссарий**

Статья содержит глоссарий терминов, составленный в рассчете на человека мало знакомого с предметом, в том числе с интернет. Однако, в нем присутствуют и термины, которые могут быть новыми и для некоторых IT-специалистов, особенно «прикладников». Так что пролистать глоссарий (14 стр. 64 - 78) рекомендуем всем. Для удобства сделан предметный указатель страниц по встречающимся терминам (в основном в глоссарии).

### 1.2.1 Порядок чтения

В случае если Вы не знакомы с предметом, или не уверены, что знакомы достаточно хорошо, имеет смысл начать с просмотра глоссария.

**Ответственным за сетевую безопасность** желательно просмотреть все, уделив особенное внимание обзору в главах 3 и 4

**Техническим специалистам** рекомендуется уделить внимание главе 11 (стр. 55 - ??), все остальное, если не возникает вопросов по аргументации, можно лишь бегло просмотреть.

### 1.2.2 Замечания по главам

**Глава 11** (стр. 55 - ??) выделена в отдельный документ, предназначенный только для технических специалистов. Скорее всего у Вас имеются обе части данного RFC.

**Глава 13** (стр. 63) дает краткий обзор ограничений присущих rootkits.

**Глава ??** (стр. ??) может быть интерсна только тем, кто участвует в развитии документа, она полностью посвещена редакторской правке и ведению версий документа, по смысловой нагрузке статьи в ней мало интересного, однако, возможно стоит взглянуть на ?? на стр. ?? - там есть ответы на кое-какие комментарии людей невнимательно читавших часть документа. Также ?? на стр. ?? описывает то, что предлагалось поменять, но релизер счел изменения не стоящими усилий.

## 1.3 Цель

**Данная статья ставит своей целью описание рекомендаций по написанию rootkits. В идеале - описание минимально необходимых требований к качественному rootkit, то есть документ класса RFC.**

*Для разработки качественного ПО, требуются соответствующие подходы - выработка требований, построение архитектуры, планирование, а не кодинг на коленке сломя голову. Это все применимо и к руткитам, и даже в большей степени - из-за специфичности разработки - код работающий с ядром ОС (существенная часть кода rootkit) требует внимательности и продуманности.*

Данная статья развивается чтобы заполнить нишу в части описания правильной архитектуры и выработки требований к реализации.

**Сделано:**

Рассматриваются методы обнаружения (вкратце) и методы скрытия (в общих чертах), модели угроз для закладки, управляющих серверов и ботнета в целом. Примеры и описания утилит пока под только под Windows. Декларируется минимально необходимый набор требований к реализации качественного rootkit: наборы требований к протоколам, алгоритмам, языкам реализации, наборы требований при реализации клиент-серверной (не peer2peer) модели взаимодействия.

**В планах:**

см. ??, ?? .

**Что такое rootkit**

rootkit - «самый продвинутый» вариант реализации программных закладок. Для того чтобы удовлетворять классу rootkit, закладка должна реализовывать невидимость своего присутствия - как для средств обнаружения программ включённых в комплект ОС, так и для утилит сторонних производителей. Такой уровень невидимости достигается путём перехвата внутренних функций ОС на уровне ядра[1], т.е. перехватом функций к которым обращаются как процессы прикладного режима, так и функции ОС. Существуют различные подходы к классификации руткитов, они будут вкратце рассмотрены далее.[2]

[1] самый низкий, или «самый внутренний» уровень работы ОС

[2] в частности, встречается понятие 'user-mode' rootkit. С учетом развития современных средств противодействия нежелательному ПО такие rootkit'ы, по большому счету, за rootkit'ы

**Copyright**



**Спасибо**

Хочется сказать спасибо:

- всем представителям компьютерномого undeground'а вообще.
- virii/coding сцене.
- Техническим специалистам, предоставившим свои комментарии к букве и сути данной работы.

считать уже нельзя.

# 2 Краткий обзор .

## 2.1 Текущее положение дел

**Современный RootKit** это закладка работающая с сетью и позволяющая осуществлять удалённое управление компьютером на котором он установлен. Конечно возможны варианты с исключительно локальным использованием, однако удалённое управление более чем удобно - зачастую это просто необходимость.

Это часто не индивидуальная закладка, а массово применяемая, но достаточно гибкая для индивидуальной настройки. В первую очередь это связано с тем, что, так или иначе, но для rootkit желательна отдельная серверная машина в сети.[3] Если такая машина есть, то разница между управлением с нее одним компьютером или тысячей не слишком велика.

В этой главе рассматривается пример существующих сетей зараженных машин.

## 2.2 Массовость

Разработка алгоритма массовых заражений окупается гораздо лучше индивидуальных закладок. Более того, разумно построенный алгоритм массовых заражений не исключает возможности индивидуальной работы с клиентом. Плюс контроль над значительным числом компьютеров позволяет реализовывать схемы распределённых вычислений и DDoS атаки, что весьма популярно в современности.

### 2.2.1 Примеры

Для того чтобы проще воспринимать суть вещей начнём с примера. Подходящий пример - распространение спама.

**Пример: спамерские bot-net'ы** Современные сети изобилуют спамом. Спам рассылается при помощи сетей зараженных компьютеров. Спамеры или создают собственные сети зараженных компьютеров, или покупают доступ к сети уже зараженных закладками компьютеров. Вторых - большинство. Спамеры (и не только они) используют термин «bot» для обозначения заражённого компьютера[4], а «botnet» для обозначения сети таких компьютеров.

[3] ведь не у каждой машины носителя, подключенной к сети, есть адрес доступный всему интернет.

[4] точнее этот термин часто употребляется как по отношению к заражённому компьютеру, так и по отношению к заразившей программе

**Боты(bots)** , фактически представляют собой программные закладки различной сложности. Работа спамерского бота заключается либо в предоставлении анонимного транзита для рассылки спама(см. 10), либо в собственно рассылке спама.

**Проникновение в систему** спамерских ботов происходит за счёт действий самих пользователей. Фактически, схема проста - спамеры покупают хостинг (обычно - на порно серверах - самый популярный ресурс в интернет де факто - порно контент) для exploit'ов . При посещении порно сервера у пользователя пользующегося Internet Explorer[5] срабатывает уязвимость браузера: скачивается и запускается программа, которая собственно и инсталирует закладку (т.е. бота) в ОС пользователя.

Такова картина для Win32 систем. Возможно разумеется и проникновение ботов в linux и другие unix-like системы, но опять-же из-за популярности лидируют боты на Windows системах.

**прочие популярные методы проникновения**

Получила также распространение методика установки ботов методами аналогичными распространению сетевых червей - через email рассылки с вредоносным контентом, через использование уязвимостей в ОС по сети.
Альтернативой работы с собственным хостингом может быть взлом web серверов с подменой линков или внедрением дополнительных фреймов с эксплойтами, при этом, если web страницы генерируются динамически из БД в которой присутствует sql injection моодифицируют саму БД так чтобы генерируемый код вызывал эксплуатацию уязвимостей обратившихся к сайту.

**Загрузки**

Хостинг уязвимостей среди спамеров называется хостингом «загрузок», говоря «загрузки» подразумевают покупку хостинга для эксплойтов к браузерам, поскольку, фактически, это обеспечивает загрузку закладок в компьютеры незадачливых пользователей. Загрузки не подразумевают покупку хостинга (для сайта например), они могут использоваться как услуга.

**Предоставление ботом транзита** для рассылки осуществляется следующим образом: бот[6] устанавливает на зараженном компьютере открытый прокси сервер, позволяя подключаться к нему с любого адреса в сети, при этом протокол работы не ведётся. После установки свободного прокси сервера закладка

[5] Браузер по умолчанию в windows системах, соответственно самый часто используемый браузер. Разумеется IE постоянно обновляется, но так или иначе находятся новые серьезные ошибки. Также возможно проникновение через ошибки в других браузерах, однако всвязи с популярностью (IE лидирует, за ним FireFox, затем Safari и остальные) большинство эксплойтов пишется для IE и его расширений.

[6] т.е. закладка

каким либо образом рапортует о себе, после чего спамер, зная адрес закладки, может подключаться к ней и, используя её, отправлять сообщения «от имени» заражённого компьютера.
Де факто, компьютер, предоставляющий сервис свободно доступного прокси, интересует многих пользователей сети по различным причинам[7] Поскольку бот позволяет использовать себя как прокси всем желающим, открытый им порт достаточно быстро находит множество заинтересованных людей. Таким образом, из за большого количества соединений, становится труднее вычленить из общего потока данных трафик хозяина закладки. При такой схеме спамеры обычно используют один или несколько компьютеров объединённых в сеть для рассылки почты через тысячи зараженных «транзитных» машин .
Вариантом этого случая является предоставление транзита после аутентификации клиента - чаще всего в этом случае доступ к транзиту продается владельцем сети зараженных машин (аренда ботов).

**Рассылка с бота** осуществляется в общих чертах так: после установки и осуществления ряда проверок бот обращается к серверу спамера, забирает у него список заданий и осуществляет отправку почты самостоятельно.

Владельцев рассылающих ботов, на 2006й год было не много, однако рано или поздно их станет больше, поскольку выгода такого решения в производительности очевидна - тысячи компьютеров отсылая спам дают скорость бОльшую, чем один или несколько компьютеров, рассылающих через тысячи. Это решение эффективнее, хотя и сложнее - сложнее сам алгоритм работы. Кроме того оно расширяет «целевую аудиторию». Дело в том, что компьютеры организаций находятся обычно за шлюзовым компьютером и если открыть на них прокси свободного доступа, то этот прокси из интернет будет напрямую недоступен[8], т.е. им смогут пользоваться только члены этой локальной сети, поскольку нет корректного способа установить соединение с компьютером за шлюзом. Как следствие - такие компьютеры неинтересны тем, кто рассылает спам используя транзитные соединения. Однако для алгоритма с рассылкой с бота таких ограничений нет - бот приходит к спамеру на сервер за заданием сам, не требуя никаких предварительных соединений извне и не открывая сервисов на зомбированном компьютере. У такого бота есть различные плюсы и минусы. Например: плюс в том, что отсутствие открытых портов затрудняет активный поиск через сеть, минус в том, что нет маскирующего трафика от случайно нашедших открытый сервис членов глобальной сети.

[7] Кому то, так же как и владельцу закладки, интересно использовать её для отсылки спама, кому то, в силу особенностей подключения, выгоднее работать через прокси, чем напрямую, а кому то нужна анонимность при работе с сетью.

[8] например, компьютеры за NAT'ом

## 2.3 Клиент-серверная модель

Поскольку с одного или нескольких серверов возможно управлять многими тысячами компьютеров логично называть управляющие компьютеры серверами, в основном исходя из того, что они раздают задания для клиентов (ботов/закладок). Вообще отношения закладки и рассылающего сервера соответствуют тому, что принято называть «клиент-серверная технология»:

`см. http://ru.wikipedia.org/wiki/Технология_«клиент-сервер»`

Здесь и далее взаимодействие между rootkit и управляющим сервером называется для удобства клиент-серверным[9]. В различной литературе управляющие сервера часто называют «C&C» от словосочетания «Command & Control».

### 2.3.1 Преимущества схемы клиент-сервер

- масштабируемость
- экономичность кодирования

**Масштабируемость**

- Возможность распределения серверной части алгоритма на несколько серверов.
- Возможность распределения нагрузки при росте числа контролируемых машин.

**Экономичность кодирования** Однажды написав базовую функциональность сервера и клиента далее ее можно улучшать и доводить до ума не тратя время на разработку программ целиком для каждого конкретного случая.

[9] при этом, в зависимости от контекста, rootkit может выступать и клиентом (получение заданий) и сервером(выдача файлов и прочей информации по запросу с управляющего компьютера)

# 3 Обнаружение rootkit

## 3.1 Классификация обнаружения

Поскольку современные руткиты это модульное ПО - обнаружение можно классифицировать по месту:

- на зараженном компьютере
- в сети

и по модулю обнаружение которого требуется осуществить:

- закладка
- дроппер
- управляющий сервер

**Обнаружение управляющих серверов** происходит исключительно по выявлению адресов и url к которым подключаются закоадки. В основном происходит на хонипотах.

**Обнаружение дропперов** сводится к выявлению хостящих серверов (купленных либо зараженных), скачиванию тела и созданию сигнатур для определения противодействующим локальным и сетевым ПО - антивирусами, сетевыми сенгсорами. Происходит, в основном, либо на активных хонипотах, либо на поисковых серверах - например google предлагает фильтрацию найденного ссылдочного контента по степени опасности.

**Обнаружение закладок** может быть либо сетевым либо локальным. Под сетевым обнаружением понимается и активное и пассивное сетевое обнаружение.

**Активное сетевое обнаружение закладок** это либо сканирование компьютеров подключённых к защищаемой сети на предмет несоответсвия открытых портов используемым на них сервисам, либо определение нестандартной реакции на проходящий мимо тестируемых компьютеров трафик[10].

**Пассивное сетевое обнаружение закладок** подразумевает под собой анализ трафика в сети. Либо на прокси, маршрутизаторах и коммутаторах по статистическим аномалиям и сигнатурам(шаблонам), либо на IDS (статистические аномалии плюс сигнатуры на трафик определённого типа). HoneyPots также используют пассивное определение по трафику, по крайней мере, на первоначальном этапе.

[10] классическое определение глупого сниффера - по взаимодействию с адресами из трафика не относящегося к данному компьютеру, хотя и доступному на нем, например по resolving'у таких адресов

**Обзор**

Вариантов локального обнаружения по большому счету два. Первый - обнаружение инструкций для загрузки кода[11] и прочих изменений после загрузки с внешнего к системе носителя, например CDROM, так можно провести, например, сравнение чтения реестров - «offline» и «online». Второй - обнаружения несоответствий результатов вызова тех или иных функций API на уровне пользователя (ring3) и на уровне ядра (ring0). Возможен также вариант с опросом через различные службы в пределах ring3 на предмет неких данных о системе и сравнение полученного, однако это ненадёжно.

## 3.2 Локальное обнаружение rootkit

### 3.2.1 Используемое готовое ПО

На данный момент рассматриваются только Windows ситемы. Желающие дополнить unix-like аналогами - присылайте краткие описания релизеру статьи.

**RKDetect**

RKDetect - утилита обнаружения Windows rootkit по поведению.

**Общие принципы работы**

RKdetect - небольшая утилита основанная на обнаружении отклонений, которая позволяет обнаруживать службы скрытые распространенными руткитами типа Hacker Defender. Утилита чрезвычайно проста - она перечисляет службы удалённого компьютера через WMI (на пользовательском уровне) [12] и через Service Control Manager (приложение работает на уровне пользователя - services.exe), сравнивает результаты и отображает различия. Таким образом находятся скрытые службы, используемые обычно для запуска rootkit. Такой же подход может быть использован для обнаружения процессов, разделов реестра и всего, что может скрыть rootkit.

**RootkitRevealer**

Утилита для обнаружения руткитов от 2006го года, «лечить» не умеет. Сравнивает контент файловой системы и реестра полученный чтением на низком уровне и на уровне windows API. На 2007 год в среде продвинутых писателей rootkit считался неактуальным старьем. Легко определяет только руткиты доступные полностью или частично в исходных кодах в интернет на 2006й год - AFX, Vanquish и HackerDefender .

[11] например ключей реестра и т.п.

[12] Windows Management Instrumentation (WMI) - интерфейс, обеспечивающий взаимодействие с компонентами системы, в общем случае доступными лишь через особые механизмы. WMI можно использовать в различных целях, в частности для управления компьютерами с помощью сценариев.

Скачать можно тут:
http://technet.microsoft.com/en-us/sysinternals/bb897445.aspx

**Антивирусный монитор Adinf**
Утилита устанавливается в проверяемую систему сразу же после её инсталляции и создает контрольные суммы файлов имеющихся на диске (конфигурабельно). В дальнейшем при запусках контрольные суммы сравниваются с оригинальными. Умеет обнаруживать некоторый набор вирусов, умеет передавать измененные файлы в качестве параметров для проверки антивирусным ПО.

**Tripwire**
Утилита устанавливается на проверяемую систему сразу же после её инсталляции и создает контрольные суммы файлов имеющихся на диске (конфигурабельно). В дальнейшем при запусках контрольные суммы сравниваются с оригинальными.

**AVZ,GMER** Наиболее серьезные из известных нам утилит специально нацеленных на обнаружение rootkit, активно развиваются[13] авторами.

**Антивирусные мониторы DrWEB, Kaspersky, NortonAntivirus, Panda antivirus и другие.**
На 2006-2007й годы всерьёз почти не рассматривались как антируткит-средства, поскольку 99.(9) их функциональности состояло в обнаружении известных вирусов, троянцев, червей и прочего деструктивного ПО. Обнаружение новых закладок у всех антивирусов отнюдь не на лучшем уровне - максимум, что может сказать антивирусный продукт - «подозрительно».

Тем не менее, в 2009м антируткит-технологии различного качества предоставляет большинство антивирусов. Например, на конец 2007го - KAV детектит скрытые процессы как руткит, DrWeb стал детектить и лечить новый MBR Rootkit. Качество их работы требует отдельного рассмотрения. Поскольку цель данной главы лишь обзор средств борьбы с руткитами - антивирусные продукты упомянуты весьма поверхностно.

Также отдельного рассмотрения заслуживает проактивная защита рекламируемая многими производителями антивирусов.

О методах противодействия классическим сигнатурным анализаторам можно сказать «в двух словах» следующее:

- использование алгоритмов полиморфизма

[13] на лето 2009го года

- шифрование тела закладки

Кстати, наиболее правильным является полиморфизм реализуемый при компиляции, когда антивирусные компании не получают для последующего реверсинга алгоритм полиморфизма и, таким образом, не способны выпустить версию антивируса, который будет ловить следующий «морф» rootkit'а и его дроппера.

## 3.3 Сетевое обнаружение rootkit

Обнаружение любых закладок и вредоносного кода использующего возможности сети достаточно просто, если не используются скрытые каналы передачи данных с маскировкой паразитного трафика под легитимный. При этом сетевое обнаружение можно разделить на две категории:

- обнаружение локальное - средствами Personal Firewall, как то: Kerio, Outpost и др., а также интегрированными в antivirus-software продукты программными модулями, предоставляющими функциональность personal firewall в качестве одной из опций (например в комплекте у Norton, Panda, Kaspersky), функционал аналогичный Personal Firewall имеется у HIPS (host intrusion prevention systems), например Cisco Security Agent
- обнаружение на транзитном сетевом оборудовании.

### 3.3.1 Обнаружение на локальных firewall'ах и HIPS

Доступно любому пользователю ПК, однако ненадёжно, поскольку использует перехват обращений к функциям реализующим передачу данных по сети, то есть, если rootkit осуществил перехват раньше personal firewall, то весь трафик rootkit'а идёт мимо firewall (не блокируется, не замечается) . В идеале очередность загрузки значения не имеет - руткит за счет нестандартных приемов скрытия активности (в т.ч. сетевой) может оставить персональный firewall «в дураках».

Существуют закладки, в которых реализована "необнаружимость"персональными firewall'ами любых производителей.[14]

Обнаружение средствами HIPS более вероятно, так как HIPS чаще всего включает в себя функционал локального firewall'а, но не ограничивается им. Чаще всего HIPS поставляется с некоторым набором правил и средствами их

[14]Это было абсолютно справедливо в 2006м году - на подобных закладках работал достаточно приличный по объёмам ботнет, rootkit-часть которого тестировалась на большинстве personal firewalls того времени - достоверные данные «из первых рук». В 2009м это утверждение можно считать спорным (особенно с учетом появления комплексных систем с использованием технологии гипервизоров), однако де факто борьба firewall vs malware в этом контексте - классическая вечная борьба щита и меча - после появления новых способов взлома со временем появляются способы защиты от них.

управления и дополнения. В числе прочего HIPS мониторит обращения к файловым системам, загрузку приложений и драйверов, возможные переполнения буфферов приложений.

Примеры HIPS: System Safety Monitor, AntiHook, CyberHawk, DefenceWall,Sandboxie, Cisco Security Agent.

### 3.3.2 Обнаружение на транзитных сетевых устройствах

В общем случае обнаружение активности rootkits на транзитных устройствах сводится к трем вариантам:

- обнаружение асимметрии трафика
- обнаружение аномальной активности (нетипичные объёмы трафика, порты, протоколы)
- обнаружение на основе сигнатур IDS

**Асимметрия трафика**

Протоколы используемые для работы с сетью имеют статистические характеристики доступные на точках через которые проходит трафик. Например, http протоколу характерна нормальная асимметрия, когда пользователь получает данных больше, чем отправляет[15]. Существуют анализаторы асимметрии трафика, способные сигнализировать нетипичную статистику обмена данными по различным протоколам. Помимо этого возможно определение типа туннелируемого трафика и для зашифрованных туннелей по статистическим характеристикам, так, некоторые современные IDS способны по статистическим характеристикам передаваемых данных определить, например, что внутри ssh туннеля идет http трафик. Фильтрация peer2peer сетей и приоритезация peer2peer трафика на транзитном оборудовании в ISP построена в т.ч. на этом принципе.

**маршрутизаторы**

Часто совмещают в себе задачи firewalling'a . Для обнаружения может быть использована возможность снятия статистики с устройства. Причем, поскольку трафик в интернет вещь платная, к анализу трафика во многих организациях подходят отнюдь не спустя рукава. Любой мало-мальски серьёзный маршрутизатор способен выдать статистику программе-агрегатору, которая, в свою очередь, способна выводить различные репрезентационные графики с выборкой за различные периоды времени (час, день, месяц). Такое представление данных весьма наглядно и позволяет узнать о установленных закладках по нетипичному трафику (нестандартные порты, протоколы не используемые в данной организации), либо по пикам трафика определённого типа, а также по графикам расходования трафика . Для примера можно указать на маршрутизаторы компании cisco

[15]фактически трафик пользователя это запросы, а ответный трафик (контент запрошенный сервера) обычно существенно больше по объёму данных

systems и пакет netflow tools. Аналоги программных комплектов визуализаторов трафика есть для любых производителей маршрутизаторов. Даже в случае, если маршрутизатором является обычный ПК - у не слишком ленивого администратора есть все необходимое чтобы добиться такой же наглядности представления трафика, как и при использовании маршрутизаторов той же cisco.

Обнаружение нетипичного поведения на основе трафика весьма опасно для владельцев rootkit и может послужить поводом для дальнейших разбирательств, что рано или поздно приведёт к обнаружению и, в, худшем для владельцев, rootkit'а случае, реверсингу алгоритмов работы, инсталляции и протоколов передачи информации . Это может привести к выходу на владельцев, вплоть до выявления личности с вытекающими из этого судебными преследованиями.

**прокси**

Часть сетей (значительная часть сетей организаций) предоставляют доступ к web ресурсам (и, иногда, к другим ресурсам) через proxy. Прокси могут, как и маршрутизаторы, быть совмещены с программным комплексом по визуализации трафика, то есть возможности обнаружения в общих чертах те же, что и у маршрутизаторов. Кроме этого прокси обладают возможностью хранения трафика проксируемого протокола, избирательной блокировки ресурсов доступных в сети по проксируемому протоколу, а также позволяют реализовать запуск программы по факту доступа к тому или иному ресурсу доступному через проксируемый протокол и, разумеется фильтрации и вызова скрипта по факту совпадения трафика с шаблоном какой либо атаки. То есть потенциально более опасны для владельцев rootkit.
Существуют решения «Web Application Firewall», основная их задача - фильтрация атак на web-приложения по прокси-принципу (защищается web-сервер перед которым ставится такая proxy система).

**IDS и IPS**

Основных отличиий IDS от IPS два:

Во первых, место установка в сети: IDS прослушивает трафик, а IPS пропускает трафик через себя, что позволяет на IPS использовать правила фильтрации для блокирования атак и изоляции атакующего. Преимуществом IDS является то, что он не вносит задержки в сеть. Недостатком IDS является меньшее количество возможных реакций - их всего два - посылка пакетов TCP RST и запуск некоего скрипта, который уже может обрабатывать событие по различным сценариям (это заведомо медленнее реакции IPS).

Во вторых - скорость реакции - IPS реагируют «real time», IDS с некоторой задержкой.

IDS и(или) IPS устанавливаются в любой организации всерьёз относящейся к

собственной безопасности. Представляют из себя мониторы трафика, «заточенные» под выявление неправомерных действий в сети, как то:

- атак извне сети
- подозрительной сетевой активности
- трафика генерируемого известными закладками
- запрещённых типов трафика(по портам, протоколам, содержимому)

Классические примеры:
snort (доступен всем, бесплатен[16]), cisco IDS (дорого, но очень производительно, доступно только организациям способным потратить порядка 100 тыс долларов и затем тратить деньги на обновления сигнатур),Ourmon(доступен всем, бесплатен, специально нацелен на противодействие ботнетам).
Так же широко известны:
IDS от Internet Security Systems «RealSecure»:
система RealSecure - это интеллектуальный анализатор пакетов с расширенной базой сигнатур атак действующий в реальном масштабе времени (real-time packet analysis), она относится к системам обнаружения атак, ориентированных на защиту целого сегмента сети.

IBM RealSecure Network IDS
Используемый сенсором модуль обнаружения атак Protocol Analysis Module (PAM) аналогичен модулю используемому в аппаратных устройства Proventia Network IPS. Кроме того, RealSecure Network Sensor может переконфигурировать правила межсетевого экрана Checkpoint Firewall-1 используя протокол OPSEC.

Также существуют различные коммерческие IDS доступные для фирм «средней руки».

IDS и IPS весьма опасны для владельцев rootkit. Для них существуют утилиты агрегации и обработки статистики, которые могут выдать отчёт о событиях связанных с безопасностью в аналогичном наглядном виде, как для маршрутизаторов и прокси. Это позволяет например обращать внимание на регулярные нарушения, которые в единственном экземпляре не привлекли бы внимания. Также возможна запись трафика по выявлении атак для последующего анализа.

**DarkNets** представляют из себя области адресации внутри ЛВС, в которых ни один адрес не занят компьютером или другим оборудованием способным отвечать по сети. Для DarkNets действуют нормальные правила маршрутизации, но трафик идет через сенсор IDS. Наличие unicast трафика в darknet с большой вероятностью говорит о том, что хост инициировавший трафик заражен.

[16] плюс доп. поддержка за деньги

Помимо ЛВС такие же сети существуют и в области адресов реального интерннета, что позволяет выявлять аномалии в трафике, в частности, наличие нового типа трафика в darknet может сигнализировать о появлении нового сетевого червя.

**коммутаторы, концентраторы** Существенным интеллектом для обнаружения не обладают, но могут быть использованы для подключения монитора трафика (в простейшем случае - tcpdump) и записи его на хранение для последующего анализа. Так, все коммутаторы компании cisco поддерживают port mirror - зеркалирование траффика одного порта в другой. Некоторые IPS и IDS могут предоставлять возможность динамического управления конфигурацией по событиям (такие функции также есть у специализированного ПО cisco). Последнее время концентраторы практически не выпускаются (сняты с производства).

**honeypots**

Самый эффективный и опасный для владельцев root-kit вариант. HoneyPot переводится как «ловушка». Это программа или компьютер, работающие в качестве приманки.

Наиболее продвинутые варианты honeypot'ов делаются профессионалами, причем есть проекты, где машина ловушка не просто пассивно ждет, когда на неё попадет незадачливый взломщик, а используется для простейших действий (например серфинга сети), что позволяет таким ловушками при определённом стечении обстоятельств получить закладки инсталлируемые автоматически только на компьютеры используемые для net-serfing'а.

**интерактивные honeypots**

Под интерактивным honeypot здесь понимается компьютер, которые используется для обычного серфинга сети[17]. Таким образом, на этот компьютер попадают не только активные члены сети в виде червей, вирусов и незадачливых хакеров, но так же закладки устанавливаемые исключительно при входе на какой нибудь сайт (например многие сайты с порно продают хостинг для спамеров, которые

[17]Тут стоит отметить, что классификация honeypot'ов в этой статье отличается от ставшей классической Spitzner'овской классификации, где honeypot'ы деляться на «High-interaction», «medium-interaction» и «Low-interaction» - по глубине эмуляции сервиса. Спицнер подразумевает под интерактивностью взаимодействие программ - низкоинтерактивный хонипот отличается от высокоинтерактивного глубиной эмуляции сервиса атакуемого злоумышленниками, я же использую термин интерактивность для того чтобы отделить honeypot'ы без активного участия человека (неинтерактивные) от honeypot'ов предполагающих серфинг сети пользователем (или эмуляцию поведения пользвателя скриптами). Как показано в данной статье на не серфящий сеть компьютер руткит может просто не попасть (практический пример - распространение так называемого bootkit'а (классификация команды Kaspersky lab) - заражения происходили только для тех компьютеров, которые кликали по линкам на сайтах с вредоносным контентом - открыть сайт было недостаточно), поскольку алгоритм заражения предполагает наличие пользователя за компьютером

распространяя через этот хостинг закладки организуют botnet'ы, которые затем используются для распространения спама, фишинга и кражи персональных данных).

Примером такой интерактивной системы honeypots может служить открытый недавно микрософтом проект: http://research.microsoft.com/HoneyMonkey/ .

После заражения компьютер поступает на анализ к команде специалистов, которая занимается, как минимум,выявлением методов заражения. В худшем случае для владельцев root-kits пойманный экземпляр подвергается полному реверсингу - дизассемблирование, выявление алгоритма работы, используемых для обмена информацией серверов и ресурсов сети.

Позволить себе интерактивные хонипоты с участием людей могут только большие компании либо большие госучереждения. Однако системы с эмуляцией поведения человека не так дороги в обслуживании. Различные организации поддерживают также автоматизированные системы анализа malware (например CWSandbox, ) Типичные «заинтересованные лица» в создании хонипотов:

- Компании производители ОС (Microsoft уже «проставилась», кто следующий?)
- Компании производители антивирусного ПО
- Компании продающие различные security-related сервисы
- Военные/разведывательные ведомства

**Реакция на обнаружение закладки** В случае попадания в ловушку с мониторингом на внешних к хонипоту устройствах, закладка будет обнаружена по трафику гарантированно. Если за время существования в режиме «подопытного кролика» закладка не проявила видимых деструктивных действий, то дальнейшее его исследование зависит от типа honey-pot'а. Во многих honeypot'ах проигнорируют безвредного серфера, или же просто занесут его в базу adware ПО. В редких антивирусных компаниях, возможно, займутся боле-менее плотным reversing'ом и в конечном итоге закладка появится в наборе сигнатур какой либо антивирусной компании(а значит, рано или поздно - у всех разработчиков антивирусов). Крайне редким, на мой взгляд будет случай детального разбора закладки с выяснением подробностей алгоритма её работы. Большинство антивирусных компаний удовлетворится списком перехватываемых функций и созданием сигнатуры в базу - для лечения на других компьютерах.

# 4 Скрытие rootkit

**обзор главы**

В этой главе будут рассмотрены различные варианты скрытия рукткитов от локального и сетевого обнаружения, то есть обнаружения "локальными"средствами (антивирусным ПО, anti-spyware, antiadware and so on) и сетевыми средствами - на маршрутизаторах, прокси, IDS.

Задача противодействия honeypots является задачей комплексной, в ней присутствуют элементы локального и сетевого характера, поэтому она выделена в отдельную главу номер 6 (см. на странице 46 ).

Задача по сокрытию rootkit делится на самом деле, как минимум, на три задачи:

- Локальное скрытие присутствия rootkit.
- Скрытие трафика генерируемого rootkit.
- Скрытие полной функциональности закладки при попадании в honeypot

## 4.1 Локальное скрытие rootkit

### 4.1.1 Общие рекомендации

**алгоритмы, проверки, противодействие сетевой трассировке**

**Со стороны управляющего сервера**

Для наиболее успешного сокрытия результатов проверки на honeypots и прочее контролирующий сервер должен внешне одинаково реагировать как на «плохого», так и на «хорошего» клиента.

Алгоритм противодействия антивирусным продуктам не должен быть реализован их отключением. Отключение работы HIPS/Antivirus может быть замечено самим пользователем.

Более того, в случае, если на передачу закладкой данных, согласно используемому транспортному протоколу возможно выдавать ошибку - это следует делать.[18]

При работе с управляющим сервером закладка должна проверяться на точное соблюдение инструкций сервера. Например, закладке можно по приходу за заданием отдавать команду прийти еще раз через определенный промежуток времени. Это позволит выявить закладки в процессе реверсинга и отрабатывая их по отдельному алгоритму скрыть действительный алгоритм работы ботнета и полный протокол обмена с сервером.

[18]Это поможет откреститься от обвинений обмена данными с закладкой

**Со стороны бота** По возможности часть алгоритма работы закладки должна быть реализована на сервере - это усложнит и в ряде случаев сделает практически невозможным реверсинг, в частности при возможности установления соединения на timeoute'ах можно реализовать протокол проверки на отладку.

В случае если закладка модульная необходимо шифрование модулей, при этом необходимо использовать отдельный ключ для каждого модуля. При этом сервер должен выдавать только ту часть ключей, которой достаточно для выполнения текущего набора задач. Так делается невозможным реверсинг неиспользуемых модулей. Возможна реализация последовательного получения ключей с сервера по мере прохождения проверок.

Для усложнения реверсинга возможна реализация интерпретатора своего байт-кода.[19]

**работа с файловой системой**
Желательно создать следующую схему:

1. Мониторинг присутствия пользователя (движение мыши/нажатие на кнопки) или присутствия screen saver'a.

2. В случае присутствия пользователя: встраивание своего кода в исполняющийся процесс, на каждое обращение процесса к диску «навешивается» дополнительное обращение к диску одного из thread'ов rootkit'a;

3. В случае работы screen-saver'a: равномерное обращение к диску, прекращаемое сразу по прекращению screen-saver'a.

Таким образом можно замаскировать работу с диском под работу самого пользователя, либо под работу screen-saver'a - такая маскировка полезна, поскольку активная работа с диском может быть замечена пользователем компьютера (светодиодная индикация работы с диском на многих ноутбуках сделана на виду у пользователя).

### 4.1.2 Сокрытие при загрузке с внешнего носителя

Бороться с обнаружением при загрузке с внешнего носителя можно следующим образом:

- Сохранение в BIOS компьютера или комплектующих - пока проверкой этих областей проверяющее ПО не озаботилось.

- Сохранение легитимных ссылок на места в сети, открытие которых вызовет загрузку дроппера и повторное заражение. То есть сам рукит перезагрузку не переживает.

[19] при этом однако всегда есть смысл делать оценку затраты/эффективность. То есть затраты оправданы когда труд противодействующей стороны умножается от вашего в разы, а лучше десятки и сотни раз.

**хранение руткита в BIOS**

Не имеет пока очень широкого распространения, однако известны случаи «in the wild», так, в Тайване были обнаружены компьютеры в firmware которых находилась программная закладка пытавшаяся отсылать некие данные по сети. К сожалению подробностей инцидента пока нет[20]. При хранении в BIOS необходимо использовать шифрование как можно большей части закладки, поскольку рано или поздно антивирусные компании и другие заинтересованные организации и персоны начнут проверять BIOS, как минимум, сигнатурным или другим методом.

**Методы которых следует избегать**

Если будет найдена уязвимость в проверяющем файловую систему ПО возможно хранение загрузчика спец. областях файловой системы с загрузкой по факту проверки FS при старте. Однако это легко отслеживается, если проверяется отличие всего диска, а не только файлов в рамках FS, к тому же данное поведение зависимо от наличия ошибки в проверяющем ПО, то есть ненадежно - исправленная версия будет рано или поздно выпущена производителем и закладка «попадется».

### 4.1.3 Вариант 1: Отсутствие внешней маскировки

**Описание** Под отсутствием маскировки можно понимать такое поведение, при котором работа с файловой системой и реестром не прячется а исполнение реализуется так, что нет отдельного процесса и т.о. закладка не отображается в стандартном мониторе процессов. В т.ч. позволяется произвести деинсталляцию закладки (с возможными вариантами в духе перехода на другой метод маскировки работы в системе). Маскируется только трафик работы с сетью, причем создается также легитимный трафик, который пользователю позволяется разрешить или запретить. В идеале закладка должна нести также некую минимальную полезную нагрузку.

**Реестр**

Этот метод состоит в том, что загружающая rootkit часть не прячется от просмотра во время работы rootkit, также не прячутся временные файлы. Таким образом при проверке чтения реестра разницы просто нет. Такой метод работы подразумевает, что после инсталляции rootkit реестр не меняется вообще. Это не может не создавать определённых неудобств программирующему rootkit.

[20] Хороший повод для внесения комментариев аудиторией - у текущего релизера документа, к сожалению, мало времени чтобы указывать конкретные линки в сети на те или иные инциденты, информация о которых получена общением с специалистами занятыми в области computer/network security. Так что, если Вы можете подтвердить информацию здесь и в других местах линками на ресурсы в сети - присылайте ваши дополнения.

**файловая система**

Временные данные rootkit возможно размещать в swap файле, временных системных файлах которые изменяются при каждой загрузке ОС. Вариантов достаточно много.

Сам rootkit размещается среди прочих драйверов, которых в современных ОС более чем достаточно. Предусматривается деинсталляция штатными для ОС средствами деинсталляции драйверов с последующей отработкой какой нибудь подпрограммы на это «недружественное» действие.

**Плюсы**

- При правильной реализации должно быть очень похоже на какой нибудь обычный системный драйвер.
- Возможно совместить с методом сокрытия от програмных мониторов контрольных сумм

**Минусы**

- Сложность в программировании.

### 4.1.4 Скрытие от мониторов контрольных сумм

Основой возможности скрытия от мониторов контрольных сумм служит тот факт, что современные компьютерные системы весьма динамичны. А именно подавляющее большинство пользователей инсталлирует программное обеспечение, которое совершенно легально изменяет реестр и инсталлирует свои файлы, иногда в системные каталоги. При последующем запуске монитора контрольных сумм объёмы информации об изменениях произведённых в системе может оказаться настолько большим, что пользователь, или системный администратор, контролирующий данный ПК вынужден будет просматривать имеющиеся изменения весьма поверхностно. Также весьма характерно для подобных мониторов то, что администратор монитора вынужден делать исключения для определённых несущественных файлов и каталогов, в противном случае он просто утонет в объёмах совершенно неинтересных изменений в, например, кэше internet explorer или swap file'e.

Ещё один характерный нюанс современной сетевой инфраструктуры - привязка ОС к обновлениям. Практически все современные системы подключённые к интернет по каналам ADSL пользуются обновлениями. Например MS Windows based системы пользуются сервисом windows update . Не будет преувеличением сказать, что данным сервисом пользуются и многие пользователи подключённые по менее высокоскоростным каналам.

Основная идея - сделать инсталляцию root-kit максимально похожей на часть инсталляции стороннего ПО или процедуры обновления системы.

Основной нюанс этого метода в том, чтобы не модифицировать исполняемые и прочие системные файлы вообще, либо стараться делать это как можно реже, в идеале - только один раз, во время инсталляции, причем инсталляции совмещённой с инсталляцией другого ПО.

Результатом такой методики при правильной реализации должны быть весьма успешны.

**Плюсы**

- При правильной реализации данные хранимые rootkit должны попадать в список исключений из мониторинга.
- Возможно совместить с методом сокрытия от проверки с внешнего носителя

**Минусы**

- Сложность в программировании.
- Требует довольно кропотливой и длительной пользовательской работы с установкой различных мониторов контрольных сумм и прогонкой в различных вариантах возможных конфигураций доступных для установки мониторов

### 4.1.5 Маскировка под известные закладки.

**Описание**

Такой метод базируется на отказе от скрытия факта различий между «offline» и «online» реестром, т.е. покуда ОС загружена - ключ реестра запускающий rootkit скрывается (либо вместо имеющегося отдается пустой ключ, либо выдается инф-я о его отсутствии вообще). В то же время загружающий rootkit бинарь на который указывает запись реестра делается похожим на какой нибудь известный относительно безобидный вирус. На мой взгляд этот метод не заслуживает существенного внимания, хотя может применяться, когда необходимо создать у пользователя представление о том, что утечка данных произошла через вирусное ПО или же, что он словил относительно безобидное adware.

**Плюсы**

Если пользователь таки обнаружит что-то, то, много шансов, что он успокоится на обнаружении и удалении безопасного вируса.

**Минусы**

Очевидно, что если rootkit не имеет альтернативных способов «реинкарнации», то его присутствие на обнаружении загружающего модуля будет завершено с перезагрузкой машины.

### 4.1.6 Сокрытие при попытке обнаружения несоответствий

Основоной метод борьбы в данном случае - устранение несоответствий, что может быть достигнуто только при перехвате функций на уровне ядра (а не только на уровне hook'ов «обёрток» к вызову ядерных функций) и модификации внутренних структур данных с которыми оперируют функции ядра.

## 4.2 Сетевое скрытие rootkit

По итогам нескольких выставок современных достижений в области защиты инфии я делаю следующие выводы: преимущество (по эффективности борьбы с вредоносным ПО) у технологий распознающих последствия, например увеличение трафика, нехарактерный пользователю трафик и т.п. (т.е. в аналогичные несоответствия типичному поведению можно искать и в поведении исполняемого кода на самом компьютере - жертве). Характерным для всех продуктов реализующих защиту от заранее неизвестных проблем является довольно большой порог срабатывания, так как поведение пользователя в различные дни может колебаться. По моей оценке двух/пяти процентные несоответствия должны быть незаметны на общем фоне в большинстве систем. Таким образом медленные атаки (т.е. сцеживание информации в час по чайной ложке) имеют гораздо больший шанс остаться незамеченными, чем любые другие. Вообще говоря - чем ближе активность закладки к активности самого пользователя, тем больше шансов не вылезти за рамки типичного поведения. Огрублённые оценки «темплейта работы пользователя» на мой взгляд реализуемы достаточно просто. Фактически речь идёт о наборе правил, на основании которых должна регулироваться работа с сетью (трафик), причем набор правил желательно формировать динамически на основе наблюдения за пользователем.

### 4.2.1 Доступ к сетевым дискам

При работе с сетью в крупных учреждениях вероятна ситуация, когда доступ к определённым ресурсам возможен только из «представившейся» программы (процесса прошедшего аутентификацию).

Желательно создать следующую схему:

1. Мониторинг обращения к сетевым дискам.
2. Встраивание своего кода в исполняющийся процесс, который обращается к сетевым ресурсам

3. Обращение к сетевым дискам «от имени» процесса, который к ним уже обращался

Таким образом можно обойти возможно-существующие ACL разрешающие обращение к сетевому диску только из одной программы. В принципе ту же практику можно применить по отношению к локальным дискам, однако необходимостью для локальной работы она станет только в случае установленных программ реализующих криптодиски и установленных программ мониторов работы приложений - второй случай относительно редко, но может встретиться.

### 4.2.2 Сокрытие трафика

Имеют место следующие рассуждения:

- на основании того, что трафик может быть получен для анализа сетевыми средствами следует вывод что трафик rootkkit'а надо зашифровывать
- на основании того, что нестандартные порты и протоколы могут вызвать подозрения у администратора просматривающего статистику, так и потому, что это однозначно вызовет срабатывание тревожного триггера на любой IDS такими приёмами пользоваться не следует - при возможности обмена с внешним миром по стандартным протоколам следует использовать именно эти протоколы.
- на основании того, что IDS наверняка сгенерирует тревожное событие («алерт») на несоответствие портов протоколу не следует генерировать подобный трафик
- На основании того, что существуют стандартные средства анализа трафика на правильную асимметрию следует генерировать трафик с правильной асимметрией (пояснения ниже).
- На основании того, что сама по себе передача зашифрованных данных уже повод задуматься - канал передачи данных необходимо прятать (см. выше о тунелировании в картинках на стр. 29).
- на основании того, что за закладку могут взяться всерьёз следует встраивать в неё средства обнаружения(но не защиты от!) трассировки

**Удовлетворение требований к асимметрии трафика**

Вкратце: для того чтобы удовлетворить нормальной статистике по количеству принятых и переданных байт для протокола http закладка должна эмулировать поведение пользователя. То есть она должна скачивать что то, поскольку чаще всего по http происходит именно download, а не upload файлов. Наиболее подходящим для скачивания являются картинки. Помимо прочего можно, усложнив алгоритм, приблизить похожесть контента к виду нормальной http страницы, то

есть некий случайный безобидный текст плюс картинки. То есть иными словами закладка должна уметь серфить сеть. При этом характер допустимой работы с сетью можно получить мониторингом сетевой активности самого пользователя.

Вообще говоря асимметрия трафика актуальна только для тунелирования в протоколах не использующих шифрование. Внутри ssl заботиться об асимметрии нет смысла.

**Шифрование трафика**

Требования к алгоритму шифрования просты: скорость, минимальный размер. Выбор остается за разработчиком соответствующего rootkit, могу лишь сказать, что заслуживает внимания idea всвязи с высокой скоростью его реализации на mmx-совместимых процессорах - таких сейчас большинство. Кроме того возможно применение разных алгоритмов на различных данных, т.е. если актуальность данных для анализа закладки/ботнета в случае их расшифровки ничтожно мала - можно применять на таких данных банальный xor. Разумеется payload должен быть зашифрован с использовавнием серьезных алгоритмов шифрования.

**Получение payload через скрытый канал**

payload передается закладке в зашифрованном виде. Ключ к расшифровке может быть передан по условию (например после проверки на sanbox/honeypot и прочих проверок). Желательна передача по каналам организованным согласно рекомендациям в 4.2.2 на стр. 29, при этом необходимо чтобы получить их с сервера иначе невозможно было невозможно.
Необходимо исключить возможность записи на диск расшифрованного payload и ключа расшифрования.

**Скрытие канала передачи данных**

Одним из возможных способов для организации скрытого канала передачи данных закладке является прием модифицированных картинок. Так, например, в стандарте на формат jpeg определяются поля, которые могут использоваться только специальным ориентированным на графику софтом (как то adobe photoshop, gimp и подобные специальные редакторы графики), причем некоторые - только после того как будут установлены соответствующие галочки в настройках этих программных продуктов(по крайней мере photoshop ведет себя именно так). Другой же софт, в том числе любой браузер, проигнорирует эти поля.

Еще один момент, который стоит учитывать - передача информации наружу может, но не должна быть организована через картинки, поскольку upload файлов на сервер по http происходит весьма редко. Для отправки данных на сервер можно использовать запросы с зашифрованными данными внутри запроса (например POST). Чтобы это было незаметно на фоне остального - подобные запросы, но со случайными данными должны идти на случайные сервера.

Для того чтобы наличие шифротекста было тяжело отличить от просто измененной картинки соответствующее поле в картинке должно инициализироваться случайной строкой, тогда не зная алгоритма (то есть не дизассемблировав закладку) невозможно будет отделить мусор от значащих данных.

Однако, следует понимать, что очистка картинок от таких дополнительных полей легко автоматизируется (например есть утилита jpegclean). В то же время существуют утилиты которые прячут данные в поля цветности картинки без заметного глазу ухудшения качества. Выбор конкретной реализации скрытого канала остается за реализатором закладки. По крайней мере очистка jpeg от информации в специальных полях не является сейчас стандартным пунктом в настройке прокси систем.

Также при организации тунелирования через картинки необходимо использовать картинки с статистически часто попадающимися параметрами (размер, разрешение, цветность и т.д.).

# 5 Противодействие honeypots.

Помимо классификации honeypot по типу работы (см. 3.3.2 на стр. 20 ) следует различать хонипоты по тому кто и для чего их использует. Хонипоты устанавливаются следующими типами организаций и людей:

1. Организации цель которых - противодействие вредоносному ПО
2. Любители/отдельные исследователи
3. Организации использующие honeypots для защиты корпоративной сети

Первые два типа хонипотов наиболее опасны для владельца ботнета, поскольку их цели чаще всего включают выявление алгоритмов работы закладок и ботнетов вцелом.
Последний вариант зачастую служит лишь для индикации проникновения, когда хонипот сигнализирует владельцам сети о том, что в ней завелся нежелательный элемент (программа-бот или человек). В том числе такие хонипоты могут использовать псевдо-аккаунты в различных instant messaging сетях (включая внутре-корпоративную) для регистрации ссылок на вредоносный контент (так, например, ловятся авторассылки заражёнными хостами линков на дропперы).

## 5.1 Общие замечания.

Противодействие honeypots с существенным успехом возможно только при взаимодействии с внешним по отношению к honeypot ресурсом. Дело здесь, в первую очередь, в том, что honeypot, направленный на выявление закладок, может быть абсолютно прозрачен, т.е. может использовать сетевой метод выявления закладок и offline'овый метод их анализа. Разумеется в случае offline'овых средств исследования защититься практически невозможно. Конечно можно использовать шифрование, однако ключ расшифрования придётся хранить на машине, на которой исполняется программа. Даже в случае, если ключ расшифрования берется из сети и производится зашифрование/расшифрование при исполнении программы («на лету»), это всего лишь усложнит задачу, поскольку существуют методы получения слепков памяти ОС - от стандартного для windows режима hibernate[21] до получения слепков состояний работы ОС целиком если она загружена внутри виртуальной машины.

### 5.1.1 Возможности

Резюмируя вышесказанное, можно сказать, что противодействие honeypots можно разбить на следующие направления:

[21] имеется ввиду режим при котором ОС сохраняет содержимое памяти на диск, затем выключается. При следующей загрузке ОС восстанавливает свое состояние, включая внешний вид и набор запущенных программ

- регистрация известных honeypots на управляющем сервере
- максимальное усложнение процесса реверсинга
- протокол работы с одновременными двусторонними проверками
- противодействие дизассемблированию
- аккуратная работа с LAN
- аккуратная рассылка линков на дропперы

**регистрация известных honeypots на управляющем сервере** часть IP ханипотов можно выявить через публичные services - часть проектов предоставляют free services for malware checks. Также после определенного количества ошибок коммуникации отдельно взятый бот должен признаваться хонипотом и отрабатываться отдельно, в т.ч. с записью в базу как минимум IP.

**максимальное усложнение процесса реверсинга** для того чтобы сделать его слишком дорогим для рядовых исследователей держателей хонипотов.

**протокол работы с одновременным двусторонними проверками**
Подразумевает разделение протокола работы закладка-сервер на взаимодополняющие части, одновременную эмуляцию которых на одной стороне реализовать невозможно либо крайне трудно.

**аккуратная работа с LAN**
Один из вариантов распространения внутри LAN - атаковать только хосты которые уже инициализировали трафик к зараженному, либо хосты к которым заражённый хост соединялся, т.е. отказ от активности типа сканирования сети для распространения. Это позволит избежать детектирования с использованием darknet (см. 3.3.2, стр. 19).

**аккуратная рассылка линков на дропперы**
Использование рассылки через сети instant messaging ссылок на сайты с дропперами возможно, но при этом стоит продумать схему так, чтобы не «засветить» заражение. Очевидно, что массовый icq/jabber/whatever spam будет замечен довольно быстро самим пользователем по жалобам из списка его контактов, кроме того, в случае корпоративной сети возможно попадание хоста в списки машин выявленных IDS/honeypot как зараженные.

### 5.1.2 Выявление

**Обзор методики**

Выявление хонипотов можно построить, во первых, на проверке ошибок в их конфигурации, когда хонипот имеет существенные отличия от нормальной машины по установленному на нем софту и железу. Эти несоответствия возможны у неряшливо построенных ловушек, т.е. в случае если хонипот примитивный. Во вторых, со стороны сервера можно регистрировать настораживающие события, например, уход в offline'овое состояние на несколько дней и более сразу после начала тестового периода[22]

## 5.2 Особенности honeypots

Характеристики хонипотов можно разделить на безусловные и возможные.

**Безусловные:**

- Ограничение доступа в сеть для закладки.
- Ограниченность комплектующих и соответственно парка компьютеров используемых как ловушки.

**Возможные:**

- Атипичная конфигурация софта.
- Атипичная конфигурация железа.
- Работа ОС honeypota в виртуальной машине
- работа honeypot как usermode или kernelmode rootkit

**работа honeypot как usermode или kernelmode rootkit** - наличие перехватов системных таблиц и usermode rootkit - тоже возможно ханипот.

### 5.2.1 Блокировка части алгоритмов работы закладки

Невозможность коммуникации с драйвером/модулем процедура инсталяции которого вернула success - один из признаков сандбокса.

[22] о тестовом периоде ниже, см. 5.3.6 на странице 40.

### 5.2.2 Блокировка доступа к сети

Блокируют на хонипотах заражённую машину обычно либо сразу по проявлению от неё подозрительного трафика, либо, что реже, это происходит через одну-две недели после того как наличие закладки было выявлено по создаваемому ей подозрительному трафику. Эта особенность объясняется тем, что по законодательству многих стран на деструктивные действия производимые с его сетевых адресов на владельца хонипота могут подать иск в суд, причем заведомо выигрывая его, поскольку владельцу хонипота сложно будет доказать, что он не знал о деструктивном характере трафика закладки, равно как и то что он не мог его заблокировать. Срок в, минимум, две недели объясняется тем, что немногие владельцы хонипотов могут позволить себе ждать проявления активности более длительное время.

**Факт обращения в сеть** , это очевидно, регистрируется даже если трафик шифруется и, пусть даже расшифровать его не представляется возможным, но сам факт обращений в сеть и их характер могут позволить отследить управляющий сервер (или несколько серверов, что не принципиально, так как число серверов конечно). Однако существуют весьма эффективные способы маскировки управляющих серверов - использование паразитного трафика того же типа (но никому не предназначенного) и использование средств анонимизации (цепочки анонимных прокси, специализированные сети анонимизации работы с интернет, например TOR network ). Также могут быть использованы способы частичной маскировки, когда при выяснении A-записи управляющего сервера выдаются каждый раз разные данные, например за счет ротации в DNS, либо за счет того, что используется доменное имя генерируемое по псевдослучайному алгоритму. Выявление управляющих серверов приведет, рано или поздно, к их закрытию в силу жалоб владельцев honeypot и пострадавших от деятельности ботнета.

**Однако,** это ещё не повод отказываться от зашифрования трафика, поскольку оно существенно усложняет проблему реверсирования алгоритма работы закладки и делает необходимым её дизассемблирование, что ресурсозатратно - реверсинг вообще времяемкая и отнюдь не простая процедура. Решение проблемы маскировки сервера будет рассмотрено ниже.

### 5.2.3 Набор железа

В случае организации хонипота его владелец явно ограничен доступными ресурсами - у него не может быть тысячи материнских плат, процессоров, видеокарт и прочих железок необходимых для создания компьютера. Если закладка за счет неумелого поведения владельцев хонипота сумела определить что машина, на которой она установлена является хонипотом, то в дальнейшем сервер может отрабатывать её по отдельному алгоритму. Как будет показано далее, это позволит

и в дальнейшем выявлять установку на хонипоты собранные с участием того же железа. Поскольку количество железа доступного любой конкретной компании ограничено, то хонипоты будучи пересобраны для другой конфигурации будут, весьма вероятно, содержать комплектующие от предыдущих сборок[23]. Это, в свою очередь, позволяет после первого обнаружения узнавать о любых других конфигурациях, в которых используется то же железо.

### 5.2.4 Атипичная конфигурация железа

Атипичность, в данном случае, это, например, мощный современный процессор, но совсем маленький единственный диск на пару гигабайт и всего 64 Мб памяти. Возможны другие не менее подозрительные варианты. Основной признак нетипичности конфигурации - несоответствие конфигурации оборудования некоторому эмпирически известному минимуму для боле-менее комфортной работы на компьютере с windows данной версии «на борту», а также несоответствие здравому смыслу. Возвращаясь к вышеописанному примеру, можно заострить внимание на том, что если у пользователя нашлись деньги на несколько гигагерцовый Pentium4, то очень странно, что при этом у него не было денег хотя бы на 256 Мб памяти и современный жесткий диск в десятки гигабайт .

### 5.2.5 Атипичная конфигурация ПО

Атипичность может, например, состоять в том, что на компьютере, выглядящем по набору программ так, будто на него только что поставили ОС, происходит серфиг сети, но длительное время не устанавливается никакого программного обеспечения. Тут дело в том, что «голый виндовс» практически бесполезен в работе. Можно поверить в то, что пользователь едва установив ОС сразу полез в сеть и подцепил там закладку. Но очень маловероятно, что этот пользователь за месяц не поставит ни одной программы.

### 5.2.6 Работа внутри виртуальной машины

Некоторые хонипоты могут работать внутри виртуальных машин. Это может показаться очень удобным владельцу хонипота, однако обнаружение таких хонипотов не составляет труда - в сети присутствуют примеры исходного кода.

## 5.3 Решение.

### 5.3.1 Выявление виртуальных машин

В числе прочих проверок необходимо проверять тип машины и, если это виртуальная машина (например ОС выполняется внутри **эмулятора** физической

[23]Не стоит забывать, что у каждой используемой в компьютере железки есть свой серийный номер доступный для считывания программным способом.

машины, а не на **физическом** компьютере), то предусматривать альтернативное исполнение. Причем чтобы не возникало подозрений и попыток помешать выявлению виртуальной машины ветвление алгоритма должно происходить на сервере. Реализация выявления виртуальной машины проста, в сети есть примеры.

Однако часть пользователей использует виртуальные машины с windows в повседневной работе. Т.е. сам факт работы в виртуальной машине еще не 100% доказательство, что это honeypot. Закладка должна хранить данные о своем состоянии, в частности если закладка в прошлом выполнялась в нормальной среде, а затем оказалась в виртуальной машине - это скорее всего хонипот или среда для проверки вредоносного ПО в антивирусной компании или у энтузиаста-исследователя, поскольку современные производители виртуальных машин не предоставляют возможности загрузки виртуальной машины с физического диска[24].

### 5.3.2 противодействие дизассемблированию

Шифрование блоков закладки. Нет смысла противодействовать на уровне ином как отсутствие ключей расшифрования у авера, поскольку любой другой способ лишь оттянет получение кода - в рамках общей модели угроз10.3.

### 5.3.3 обнаружение работы под отладчиком

Нет смысла противодействовать отладке, однако ее надо пытаться обнаружить и отрабатывать работу в отладчике отдельно, а именно отрапортовать о хонипоте и не расшифровывать payload.

Из полезных для обнаружения отладки способов можно перечислить следующие:

1. Проверка CRC исполняемого кода
2. проверка и использование указателя стека
3. перехват int1, int3, int0
4. использование SEH
5. Win32 API IsDebuggerPresent()
6. ключи реестра,процессы,семафоры
7. использование указателя на стек при расшифровании блоков кода
8. расшифрование от конца к началу
9. entry point tricks (TLS)

[24]так, насколько нам известно в vmware и sun virtual box, ранее в vmware была возможность загрузки с физического диска

**Проверка CRC исполняемого кода**

В x86 архитектуре отладчик в пошаговом режиме использует модификацию исполняемого кода трассируемого приложения вставкой инструкций INT3. Код закладки должен проверять CRC исполняющихся участков кода и в случае изменения отрабатывать функционал «работа под отладчиком».

**проверка указателя стека**

Отладчик может хранить данные об отладке в стеке отлаживаемого приложения. Для проверки запоминаем текущий указатель на стек, кладем в стек любое значение, вынимаем его из стека, затем сравниваем указатель на стек с сохраненным - если не равно - работа под отладчиком:

```
проверка указателя стека:
MOV     BP,SP  ; Let's pick the Stack Pointer
PUSH    AX     ; Let's store any AX mark on the stack
POP     AX     ; Pick the value from the stack
CMP     WORD PTR [BP-2], AX ; Compare against the stack
JNE     DEBUG  ; Debugger detected!
```

**использование указателя на стек при расшифровании блоков кода** существенно затруднит отладку поскольку int1 использует стек. Т.е. под отладчиком блок не будет расшифрован.

**перехват int1, int3, int0**

Прерывания int1 и int3 используются отладчиками. Перехваченные прерывания можно использовать для расшифровки блоков кода, а также для рапорта о хонипоте. Перехват int1 и int3 легко обходится в отладчиках использующих виртуальные машины, виртуальные машины следует распозавать отдельно (см. стр. 35). Некоторые отладчики некорректно отрабатывают перехват int0.

**использование SEH**

SEH - structured exception handling предоставляет из себя использование блоков типа:

```
try {
// код который может вызывать exception
}
catch {
// код который обрабатывает возможные exceptions
}
```

идея в том, чтобы намеренно создавать исключения в блоке try гарантированно получая упраление в блоке catch. Есть шанс, что в процессе отладки реверсер может пропустить часть кода, которая выполняется в обработчике эксцепшена. Т.е. проверки CRC например можно разместить в блоке catch.

**ключи реестра,процессы,семафоры**

Отладчиков не так уж и много, соответственно можно обнаруживать соответствующие им ключи реестра, имена процеесов, глобальные системные семафоры и прочие объекты (например имена драйверов). Далее закладка может иметь некий индивидуальный подход к установленному отладчику.Впрочем наличие установленного в системе отладчика еще не говорит о том, что он применяется именно к боту. Наиболее безопасным был бы отказ работы закладки в случае если в системе установлен отладчик, однако таким образом пропускаются компьютеры многих программистов.В зависимости от целей работы данного ботнета можно отрабатыавть ситуацию по разному.

**расшифрование от конца к началу**

использование процедур расшифрования таким образом чтобы начало расшифровывваемого блока записывалось последним немного улучшает шансы на выполнение части кода без контроля отладчиком - реверсер не может поставить breakpoint на начало блока до его расшифрования - int3 перезапишется данными в процессе расшифрования.

**entry point tricks (TLS)**

Windows specific: PE формат позволяет использовать более 1й точки входа при запуске - Thread Local Storage (TLS) позволит выполнить код до основной точки входа.

### 5.3.4 Скрытие управляющего сервера

Как уже указывалось ранее, совершенно очевидно, что выявить управляющие сервера можно по частоте обращений и их характеру. Выход прост: закладка должна сама серфить сеть в случайном порядке. Таким образом исключается возможность выяснить управляющий сервер просто анализируя трафик, без анализа закладки с отладчиком в руках.

**Реализация серфинга** должна быть сделана внимательно, корректным образом, поскольку в случае ошибок могут возникнуть проблемы со стороны поисковых машин и рядовых пользователей.

**Во первых,** не следует искать случайные наборы символов. Резкое увеличение поисковых запросов, с мизерным hitrate в поисковом кэше гарантированно привлечет внимание.

**Во вторых,** не стоит только искать - надо «ходить» по найденным ссылкам, иначе возрастет нагрузка на поисковик и сильно изменится статистика наиболее часто искомых слов, что тоже плохо. Кроме того, если не позаботиться о распределении запросов случайным образом, то нагрузка на поисковые системы также

сильно возрастет, что может привести к незапланированным отказам в обслуживании, что тем более привлечет внимание.

**В третьих,** следует составлять поисковую строку так, чтобы находить достаточное количество результатов, то есть не более трех-пяти слов в запросе, количество должно меняться от запроса к запросу.

**Инициализация серфинга** возможна списком поисковых машин (google, yandex, rambler, yahoo и другие ), а набор слов для поиска закладка может брать из файлов носителя либо из собственного словаря в случайном порядке.

### 5.3.5 Инкубационный период

Пользуясь природными аналогиями можно вспомнить, что у многих заболеваний есть инкубационный период, когда анализы не дают возможности установить факт заболевания. С целью скрытия rootkits от владельцев хонипотов следует прибегнуть к тому же средству: в момент инсталляции закладка может не проявлять вообще никакой сетевой активности.

Так же актуально наличие паузы в несколько минут перед тем, как закладка прописывает себя на диск - часть хонипотов используют раннее обнаружение - через несколько секунд (или минут) делается холодная перезагрузка хонипота, после чего используются методы offline анализа.

Большой инкубационный период нужен для того чтобы, во первых, обойти краткосрочные проверки на очень многих honeypots.

Инкубационный период, равный месяцу, а тем более двум, позволит избежать выявления закладки во многих интерактивных хонипотах.

Также возможно деление инкубационного периода на стадии - например:

1. пауза от начала выполнения - никакой активности
2. пауза перед инсталляцией в автозапуск на диск - сбор информации о системе, тесты на хонипот без участия сервера
3. регистрация на сервере с отправкой информации о системе
4. двусторонние тесты (участвуют и сервер и бот) на хонипот и отладку

**Учет возможности переустановки времени** на компьютере является необходимым условием соблюдения сроков инкубационного периода. При установке закладки она должна запомнить текущее время на компьютере, плюс сохранять локальное время при каждой загрузке компьютера. Подозрительными являются скачки времени между перезагрузками компьютера более чем в трое суток. В таких случаях закладка должна зафиксировать это и, возможно, отреагировать какими либо действиями.

Сервер должен иметь возможность выдать клиенту текущее время, равно как и клиент серверу. Закладка таким образом может детектить сетевой honeypot за счет выявления разницы между данными полученными по NNTP и по скрытому каналу от сервера. Сервер может сопоставлять время на боте с прочими параметрами отрапортованными им выявляя реверсера.

Например, если скачки времени вперед на существенную величину производятся постоянно (например, более нескольких раз подряд за короткий срок[25]), также нормальными являются скачки если используется компьютер с несколькими ОС - есть шанс, что пользователь все это время работал в другой ОС. Однако скачки времени вперед будут однозначно характеризовать хост как нетипичный если реальное время (полученное по NTP и от управляющего сервера) существенно отличается.

При наличии доступа в сеть закладка может отрапортовать конфигурацию железа, которое с большой вероятностью является honeypot'ом. Проверка времени по NTP может не дать достоверных результатов, так как обращения по NNTP и другим стандартным протоколам могут перенаправляться и модифицироваться. Так, например, CWSandbox умеет сэмулировать отправку по SMTP не отправляя данных вовне, а лишь представляясь удаленной системой.

### 5.3.6 Тестовый период

После окончания инкубационного периода закладка должна пройти тестовый период.

Во время работы в тестовом режиме закладка должна серфить сеть также как и в рабочем, причем не исключая из серфинга списка контролирующих серверов[26].

### 5.3.7 Ступенчатая загрузка

Чтобы на каждом этапе работы закладки исключить потенциальную возможность попадания в руки вероятного противника информации о структуре закладки необходимо сделать загрузку закладки ступенчатой.

1. 1-я ступень инсталлируется «как обычно» - при входе на заражённый web server[27]. Начинается инкубационный период.

[25] «несколько» и «подряд» - существенные факторы в том смысле, что бывает, что люди переустанавливают время на компьютере чтобы обмануть регистрационную программу устанавливаемого платного ПО. Однако не стоит забывать, что человеку свойственно бывать в отпусках, так что выключение компьютера может оказаться вполне нормальным

[26] это позволит исключить обособление списка управляющих серверов от прочих, что было бы видно человеку, реверсирующему закладку

[27] либо в индивидуальном порядке человеком временно получившим доступ к атакуемому компьютеру - скачиванием и запуском инсталлятора первой ступени, что в других случаях выполняет эксплойт к браузеру

2. -я ступень скачивается первой при удачном окончании инкубационного периода. Начинается тестовый период.

3. -я ступень скачивается второй при удачном окончании тестового периода. С этого момента можно использовать payload, имея существенную вероятность того, что носитель не является хонипотом.

### 5.3.8 Общие требования к payload

С целью значительного усложнения выявления алгоритма payload необходимо реализовать:

1. Получение payload через скрытый канал в зашифрованном виде[28]
2. Получение данных для работы payload с сервера через скрытый канал в зашифрованном виде
3. Получение ключа для расшифровки payload через скрытый канал в зашифрованном виде
4. Получение отдельного ключа для расшифровки данных для работы payload через скрытый канал в зашифрованном виде
5. хранение payload и ключа его расшифровки только в памяти без записи на диск
6. минимизация количества инструкций payload хранимых в расшифрованном виде
7. ротация ключей шифрования на сервере раз в сутки или чаще [29]
8. хранение внутри закладки информации о конфигурации компьютера на момент исполнения с отрабаткой ситуаций по перемещению на другой ПК как нештатной.

### 5.3.9 Общие требования к данным для payload

Если payload подразумевает получение данных для произведения операций над данными носителя, например, для поиска по ключевым словам, то эти данные надо хранить, во первых, исключительно в памяти ОС, а во вторых их крайне желательно зашифровывать на отдельном ключе, который получается с сервера и также хранится только в памяти. На том же ключе должны быть зашифрованы промежуточные данные получаемые при обработке данных на основе информации получаемой с сервера. Если payload потребуется хранить, например, копии

[28] поток данных в скрытом канале зашифровывается в штатном порядке, то есть ключ и payload будут зашифрованы дважды, но это не существенно - так проще для единообразия

[29] поскольку payload для большинства закладок одинаков - ротация на сервере даже раз в час не должна быть существенной проблемой с точки зрения ресурсоемкости

файлов пользователя, то они должны быть зашифрованы на отдельном ключе, отличном от текущего для зашифрования/расшифрования самого payload и его временных данных. Назовем такой «ключ хранения». Использование ключа хранения с ротацией «on-demand»[30].
Способ хранения payload только в памяти также известен как «anti-forensics».

**Преимущества данного подхода**

- Использование in memory only хранения payload и ключа его расшифрования позволит значительно затруднить реверсинг, а также позволит выдавать ложную информацию о задачах которые были поставлены закладке, если реверсирующая сторона была обнаружена. Более того, при использовании получения ключа для расшифровки исполняемого кода и данных с сервера возникает гораздо больше узловых точек, в которых реверсер может обнаружить свое присутствие в силу несоблюдения протокола обмена данными, в первую очередь - по временным параметрам

- Использование ротируемых ключей хранения позволит избежать раскрытия объемов утечки информации с носителя rootkit, поскольку, после того как файл признан необходимым к хранению очень мало возможных ситуаций, когда для владельца rootkit будет осмысленным его расшифрование на клиенте вместо отправки файла на сервер и дальнейшей обработки локально, в безопасных условиях, когда нет необходимости прятать свои действия от пользователя и проверяющего ПО.

### 5.3.10 Реализация шифрования

**Шифрование сетевого трафика**

Некоторые компании имеют маленький интернет трафик и делают полное журналирование работы с интернет за день и, возможно, более. Для того чтобы исключить расшифровку трафика обмена с сервером для любой закладки за счет утечки ключа полученного реверсингом одной закладки необходимо реализовать:

- индивидуальность сеансового ключа для каждой закладки

- алгоритм смены сеансового ключа

Это позволит исключить возможность составления сигнатур для IDS с целью их дальнейшего использования для обнаружения установленных закладок в других местах.

При этом однако необходимо учитывать проблему старых ключей, например, клиент был выключен на время отпуска или выходные, если ротация ключей уже

[30] то есть должна производиться ротация ключей: например раз в день ключ должен меняться

произошла - такого клиента необходимо либо игнорировать, либо, например, переводить обратно в режим инкубационного периода. В некоторых случаях будет иметь смысл прогон такого клиента на дополнительных проверках и понижение индекса доверия со стороны управляющего сервера.

**Сеансовый ключ.**

Под сеансовым ключом понимается ключ используемый для зашифрования/расшифрования в транспортном протоколе поверх скрытого канала[31]

**использование нескольких алгоритмов шифрования**

Первичный запуск должен использовать bruteforce схему, payload должен быть зашифрован с использованием стойких алгоритмов с длиннами ключей не поддающимися bruteforce в разумные сроки (сто и более лет). См. также 6.2 на стр. 47 и 6.2 на стр. 48.

## 5.4 Реакция

### 5.4.1 Удачность периода

Окончательные выводы о любом периоде в процессе которого идет обмен данными с сервером, должны осуществляться сервером. Разумеется, если в инкубационном периоде закладка не проявляет сетевой активности - у сервера просто нет данных для выявления его удачности или неудачности.

Поскольку сервер отдает обновления для каждой ступени, то по факту обнаружения несоблюдения алгоритма он может:

- зарегистрировать во внутренней базе данных информацию о характеристиках противодействующей стороны - IP,OS,идентификаторы железа, уникальный идентификатор пойманной закладки и прочее
- выдавать различные файлы с мусором вместо зашифрованных данных в области картинки реализующей скрытый транспортный протокол передачи данных (только если нет средств проверки валидности данного типа данных на закладке, иначе передача мусора будет говорить о том, что сервер обнаружил противодействие).
- выдать payload не соответствующий реальному использованию остальных закладок, т.е. фактически скормить любой алгоритм, в следовании которому данного типа закладок надо убедить реверсера/автоматическую систему анализа malware.

[31] в случае с рассматривавшимся выше скрытым каналом в jpeg картинках это ключ которым расшифровывается область данных содержащая блок данных транспортного протокола(что мы прячем в транспортном протоколе - набор команд боту или набор данных для последующей дообработки - в данном случае не важно.)

- блокировать обмен данными с закладкой на уровне сети или на уровне протокола в рамках ботнета

### 5.4.2 Варианты реакции на обнаружение хонипота

В случае запуска в сандбоксе основная задача - сдетектить и отстучать на C&C для внесения в базу очередного хонипота.

**Возможности:**
Возможны различные варианты ответов на выявление неудачности каждого из периодов. Их гораздо больше чем описанных здесь, но по мнению авторов наиболее разумным является следующее поведение:

- Выявление хонипота на момент инкубационного периода должно повлечь, в первую очередь, попытку отправки информации о нем на сервер. Отправку информации желательно организовать одновременно по нескольким каналам, например smptp(email)/http/https/dns, как минимум двум
- Выявление хонипота в тестовом режиме на мой взгляд, должно повлечь выдачу на хост заведомо неправильного бинарника, а если точнее - бинарника с совсем другими, почти безобидными функциями.
- Выявление хонипота в рабочем режиме должно повлечь за собой блокирование работы с данной закладкой - в нормальной ситуации хонипот должен быть обнаружен на более ранних этапах.

Следует также по разному отрабатывать хонипот и реверсинг. В частности, при обнаружении реверсинга закладки желательно инициировать смену части ключей используемых в ботнете.

**Реакция на обнаружене в инкубационном периоде:** Неплохо было бы использование dns туннеля в числе прочих методов. Есть немалая вероятность того, что DNS трафик, который по сути своей деструктивным быть почти не может (кроме попыток DoS), не будет запрещаться, или перенаправляться.

**Реакция на обнаружение в тестовом периоде:** В зависимости от этапа выявления алгоритма по бинарю, на котором находится реверсер,он может знать определенную часть общей схемы. Так если выявлено, что закладка находится в тестовом режиме, то можно предположить, что реверсер уже выявил в ней возможности класса «download and execute» используемые для обновления. Разумно в таком случае отдать реверсеру какую нибудь безобидную закладку, которая не будет делать ничего особенно неприятного, а например, всего лишь, рапортовать список установленного в системе ПО на какой нибудь сервер. Однако это подразумевает поддержку дополнительного сервера и другого протокола обмена между закладками и сервером (чтобы не вызывать подозрений), и, таким образом,

может быть слишком накладным. Поэтому более подходящим может оказаться просто блокирование работы с пойманной закладкой, хотя вариант с выдачей некоего «левого» payload гораздо вероятнее приведет на противодействующей стороне к решению, что закладка уже достаточно изучена и дальнейшее исследование можно прекратить.

**Реакция на обнаружение в рабочем режиме:**
Обнаружение хонипота в рабочем режиме, а не ранее может говорить о том, что часть алгоритма работы ботнета раскрыта, так что этот вариант надо отрабатывать особым образом.

### База данных клиентов

Основываясь на вышеуказанном сервер контролирующий закладку должен вести базу данных по железу на котором исполняется закладка. На основании имеющейся базы данных и ряда проверок сервер[32] сможет определить[33] с достаточно высокой вероятностью имеет ли он дело с honeypot, реверсером энтузиастом, профессиональным реверсер или же с нормальным пользователем.

[32] не клиент!
[33] например используя повторяемость ситуации

# 6 Противодействие антивирусному ПО.

## 6.1 Общие замечания.

### 6.1.1 Особенности антивирусного ПО

**Безусловные:**

- при применении эмуляции в момент анализа бинарника не может ждать существенное время

**Возможные:**

- Реализация personal firewall

## 6.2 Возможности противодействия

Резюмируя вышесказанное, можно сказать, что противодействие антивирусному ПО можно разбить на следующие направления:

- антиэвристические приемы - antiheuristics
- антиэмуляционные приемы -antiemulation
- обнаружение антивирусного ПО + индивидуальный подход к антивирусу
- антиотладка - antidebugging
- противодействие дизассемблированию - antidisassembly

**противодействие дизассемблированию** Помимо шифрование блоков закладки5.3.2 для скрытия алгоритмов используемого payload необходимо использовать шифрование для исключения определения антивирусом последовательностей кода используемых для эксплуатации уязвимостей ОС и перехвата управления. Это аналогично шифрованию используемому вирусами.

**обнаружение антивирусного ПО + индивидуальный подход к антивирусу** - антивирусного ПО выпускается не так уж и много - не более десятка популярных производителей. Соответственно можно обнаруживать соответствующие им ключи реестра, имена процеесов, глобальные системные семафоры и прочие объекты.

**антиэвристические приемы**

**Обфускация кода.**

Как минимум - подготовка регистров арифметическими операциями вместо прямой загрузки перед вызовом функций их использующих. Пример:

Вместо:

```
MOV   CX, 100h ; this many bytes
MOV   AH, 40h  ; to write
INT   21h      ; use main DOS handler
```

Используется:

```
MOV          CX,003Fh  ; CX=003Fh
INC          CX        ; CX=CX+1 (CX=0040h)
XCHG         CH,CL     ; swap CH and CL (CX=4000h)
XCHG         AX,CX     ; swap AX and CX (AX=4000h)
MOV          CX,0100h  ; CX=100h
INT          21h
```

**CRC**

Использование CRC вместо имен функций для поиска.

**non-writabble entry point & other code**

Writeable сегмент кода - добавка подозрений эвристику. Альтернатива - расшифрование в стеке.

**антиэмуляционные приемы**

**медленный вход**

Подразумевает использоваааание длительных вычислений перед началом работы кода который может быть признан эмулятором как подозрительный. Эмулятор кода в момент проверки файлов не может ждать несколько минут на каждый файл.

**использоваание random encryption key**

В момент запуска закладки она может расшифровывать часть себя используя подбор ключа шифрования. Это одновременно и медленный вход и позволит избежать хранения ключа на закладке. Однако такой способ не должен использоваться для зашифрования всего кода, поскольку для того чтобы закладка смогла расшифровывать себя в разумное время (10 - 30 минут) схема шифрования должна быть относительно слабой.

**использоваание расшифровывающего кода как части ключа**
Некоторые эмуляторы оптимизируют код подбирающий ключи, т.е. этот прием нарушит расшифрование.

### 6.2.1 Инкубационный период

1. пауза от начала выполнения - никакой активности
2. пауза перед инсталляцией в автозапуск на диск - сбор информации о системе, тесты на известные антивирусы
3. регистрация на сервере с отправкой информации о системе

# 7 Различные приёмы возвращения rootkit после того как основной модуль был удалён

## 7.1 Установка закладок «реинкарнаторов»

После успешного окончания инкубационного периода rootkit устанавливает несколько автономных программных модулей, единственным payload которых является загрузка текущей версии rootkit-инсталлятора из интернет. Модуль стартует при загрузке системы или, что лучше, при загрузке определённых приложений. Скачивание происходит раз в месяц плюс минус несколько дней по случайному принципу. После того как rootkit скачан он обращается к серверу за инструкциями и, в зависимости от решения принимаемого на сервере получает к исполнению тот или иной бинарь или получает инструкцию выгрузиться из памяти.

### 7.1.1 Описание закладки реинкарнатора.

Закладка-инсталлятор должна представлять собой простой http/irc/(другие протоколы?) downloader/executor. Задача такой закладки - скачать и выполнить новый модуль первой ступени в случае удаления закладки из системы.

Таким образом мы можем добиться того, что данный модуль сможет повторить инфицирование новой версией rootkit данной машины через достаточный промежуток времени. Разумеется только сервер всегда сможет отдать любой бинарь, вместо текущей версии опираясь на собственные данные о запрашивающей машине.

### 7.1.2 Предостережение.

Вообще говоря, делать возрат на машину, на которой модуль был удален опасно. Уж очень велика возможность попасть в грамотно сделанный хонипот. Ведь владелец как то смог отловить и деактивировать rootkit. Так что лучше 10 раз подумать прежде чем вступать на такой опасный очередным разоблачением путь. Единственный вариант, при котором такое может быть нужным, если сеть закладок осталась «на автопилоте» без контроля и за время, которое она так работала все или часть антивирусных пакетов вдруг прозрели и стали определять наличие некоторых модулей закладки, производя лечение не полностью, раз независимая закладка-реинкарнатор сохранилась (если это не ловушка конечно).

### 7.1.3 Минимально необходимый список проверок.

К возвращению rootkit на компьютер, с которого он был ранее удален требует особенно жестких проверок. Ведь, в данном случае **черезвычайно велик риск напороться на активное противодействие и подготовленную ловушку**. Решение о возврате желательно принимать индивидуально, а хосты на которые rootkit инсталируется повторно должны быть, во первых, под особым контролем, а во вторых, обслуживаться на отдельных серверах.

- отдельные сервера

**Отдельные сервера** Это требование связано с тем, что если за сеть закладок возьмутся всерьез, то соответствующие заинтересованные госструктуры могут обязать к принятию мер, как минимум, следующих юридических лиц:

- Провайдера хостинга
- Провайдера трафика

**провайдера хостинга** можно обязать предоставить физический доступ к данным на компьютере , что решаемо со стороны владельца ботнета использованием зашифрованных файловых систем (частично, поскольку если сервер виртуальный - может быть применен дамп памяти, то есть так можно получить ключи к файловой системе, следовательно к данным). Тем более это решаемо в случае, если покупается хостинг физического компьютера - например датчик открытия корпуса завязывается на мобильник, который висит на зарядке питающейся от Б/П. По срабатыванию датчика отправляется СМС. Однако, это отдельный след в real life: предоставить компьютер должен либо живой человек, либо служба доставки, которая, соответственно, должна принять его отнюдь не у виртуального персонажа, а у реального человека с паспортом той или иной страны - это слабое звено, поскольку анонимизация в реальном мире доступна лишь ОПГ и различным боле-менее курпным компаниям и специфическим госструктурам.

**провайдера трафика** могут обязать, с какого либо момента, сохранять лог обращений и даже трафик работы с сервером по некоторым или всем протоколам (СОРМ это предусматривает), таким образом это может помочь в выявлении управляющего сервером персонала и, затем, владельца ботнета. Противодействовать этому можно лишь используя для административного доступа специальные средства анонимизации - цепочки анонимных прокси и сети подобные TOR.

# 8 Различные комментарии

Данная глава на данный момент - разрозненные заметки на полях.

Термин payload я подобрал в журнале 29A, в котором он использовался для обозначения функций выполняемых вирусом не относящихся к заражению и распространению. В нашем случае этот термин употребляется как обозначение задач rootkit выполняемых по указанию владельца в автоматическом режиме (то есть без специальных на то команд от владельца). Пользуясь случаем, хочу высказать огромное спасибо 29A и всем участникам virus scene, кто помог мне своими опубликованными идеями.[34] Обзор истории 29A: http://bugtraq.ru/library/underground/29a.html

## 8.1 Reversing

Reversing - восстановление алгоритма по «бинарю» (см. 14). Очевидно, что владелец rootkit заинтересован в максимально возможной секретности алгоритма его работы, так как знание алгоритма помогает противодействию экземпляров rootkit и их сетей (kitnet/botnet), помогает понять цели установки и исполняемых им в процессе работы действий а также может помочь в преследовании владельца ботнета по законам страны проживания.

## 8.2 Возможные варианты реализации

Чтение различных таблиц из юзер и кернелмоды и сравнение можно попробовать обойти за счет анализа откуда идет операция чтения. То бишь ядро грузится в определенный диапазон адресов. Драйвера тоже рядом, но они после ядра. Можно попробовать вычислить диапазон физических адресов из которого давать на чтение полную таблицу IDT например, а из всего что дальше только часть ее. То есть драйвер загруженный позже будет доступаться из памяти дальше определенной границы, если на чтение области памяти в районе IDT можно поставить хук, то там эту ситуацию можно попробовать отработать.

[34] Журнал 29A это один из современных (на лето 2004) журналов вирусной сцены, на момент начала написания этой статьи с ним можно было ознакомится по url: http://www.29a.host.sk/. На 2009й год команда 29A это уже, к сожалению, история сцены .

# 9 Архитектура ботнета

Это пока пустая секция - в 2do:

классификация - классические + в рамках этой статьи. модульность (dropper/loader/payload/wha
распределение управляющих серверов по задачам. степени свободы в взаимодей-
ствии по сети. типы по протоколам распространения.

```
[08:07:53] XXXX: 7.1.1. drugie protokoli IMHO messaging protocols (msn, xmpp, skype?) s
[08:09:30] [olli]: скайп да.. но скорее маскировка под него. А с реальным участием живы
```

# 10 Модель угроз в рамках ботнета

**Обзор главы**

Рассматриваются модели угроз для ботнета в целом, для управляющих серверов и закладки.

## 10.1 Модель угроз для ботнета вцелом.

1. Перехват управления
2. закрытие всех C&C серверов
3. уничтожение всех ботов
4. выявление алгоритма работы ботнета вцелом
5. выявление алгоритма работы частей ботнета
6. закрытие/очистка серверов с дропперами

**выявление алгоритма работы ботнета вцелом** делиться не несколько возможных точек утечки информации:

1. анализ дропперов на атакуемом хосте
2. анализ дропперов на раздающем их хосте
3. анализ серверной части ботнета при физическом доступе к серверам
4. анализ серверной части ботнета при взломе серверов

**закрытие/очистка серверов с дропперами**

Разрумеется, рано или поздно наличие вредоносного контента будет обнаружено владельцами взломанного ресурса.

Если сервера не покупные на, например, порнохостинге, для которого заработать на дистрибуции дропперов нормальная бизнес модель, а взломанные, то следует учитывать способ доступа к взламываемым серверам. Для безопасных протоколов вполне годятся tor-подобные сети, однако для ftp/http tor должен комбинироваться с тунеллированием - один конец туннеля внутри top сети (исходящий атакующий трафик), второй конец терминируется на платном прокси, туннель разумеется зашифрованный. Это необходимо, поскольку tor exit nodes очень часто снифят трафик и пытаются использовать атаки типа man in the middle на клиентов tor сетей. Работать через tor напрямую используя небезопасные протоколы - дарить доступ к ресурсам владельцам tor exit nodes. Шифрованное соединение через арендованных ботов также не может быть более доверенным, чем платная прокси, поскольку нет гарантии, что арендованные боты не снифят трафик.

## 10.2 Модель угроз на управляющих серверах

1. трассировка управляющего персонала
2. abuse response
3. DoS
4. подмена закладки

## 10.3 Модель угроз на закладке

1. Выявление протокола работы с серверами
2. Выявление алгоритма работы payload и списка payload
3. подмена управляющих серверов

## 10.4 Модель угроз для владельцев и управляющего персонала

1. Выявление личностей владельцев
2. Выявление личностей управляющего персонала
3. Разрушение бизнес-модели

Тут надо будет еще сделать обзор используемых:
систем электронных денег,
тактик анонимизации оплаты,
вывода денег из электронной формы в бумажную.

# 11 Набор свойств необходимых к реализации в качественном rootkit

Тут только анонс - краткое содержание (обзор) - второй части данного rfc, предназначенной только для технических специалистов. Вторая часть данного rfc распространяется на анологичных условиях отдельным документом, скорее всего она у Вас есть, если же ее у Вас нет и Вы чуствуете в себе силы принять участие в обсуждении технических тонкостей - .пишите текущему релизеру документа.

**Внимание: эта глава развиваться не будет. Детальная техническая разработка требований к качественному rootkit вынесена в отдельную статью, которая является логическим продолжением данной. Это сделано, в том числе, для облегчения развития и ведения версий данного RFC.**

**Обзор второй части rfc**

Во второй части техническим языком без аргументации (она как раз тут, в первой части):

1. preface+editorial
2. требования к языкам программирования
3. требования к архитектуре
4. требования к протоколу

Все это поделено сервер/закладка/степень важности/прочие параметры.

# 12 Примеры payload современных rootkits

## 12.1 Paid proxy

Владелец ботнета продает доступ к прокси: доступ к транзиту на боте осуществляется по ключу, по окончании срока аренды ключ меняется - так называемая «аренда ботов».

## 12.2 DDoS

**Массовое единомоментное открытие соединений**

Сотня тысяч компьютеров послав одновременно запрос на использование затруднят работу любого сервиса. Использование сетей зомбированных компьютеров для организации DDoS атак уже давно стало привычным событием.
Раскрытие взаимодействия в рамках сети зомбированных компьютеров - очевидное следствие участия в DDoS атаках. Поэтому, если сохранение rootkit на данном компьютере существенно, то его не следует использовать при организации DDoS атак.

## 12.3 Distriibuted Calculations

### 12.3.1 Distributed Net ( dnet )

- широко известная инициатива интернет сообщества по оценке стойкости алгоритмов шифрования и алгоритмов хеширования. Суть её состоит в том, что каждый желающий может заставить свой компьютер работать над некой вычислительной задачей совместно с компьютерами других энтузиастов. По умолчанию этот проект использует остаточную вычислительную мощность компьютера, т.е. то, что осталось после выполнения задач ОС и пользователя.

Пример организации distributed net энтузиастами на территории exUSSR можно посмотреть на http://bugtraq.ru/dnet/ .

Инфицированные компьютеры, точно так же, как и любые другие, используются большую часть времени не на полную мощность, так что возможно написать модуль к rootkit, который будет использовать время простоя этих машин на пользу владельцу root-kit сети.

### 12.3.2 Intelligent Distributed Calculations

Совершенно необязательно производить перебор «подряд», как это делается в реализации Distributed Net[35] (см. выше, стр. 56 глава 12.3.1). Дело в том, что проект Distributed Net не слишком заинтересован в оптимизации алгоритма вычислений и поддерживается энтузиастами, которые зачастую почти не имеют отношения к криптографии. Однако, согласно Шнайеру, при вычислениях связанных, например, с выявлением закрытых ключей существенную роль может

[35] такой перебор называют перебором грубой силы - «bruteforce»

играть объем доступной памяти. RootKit'у вполне по силам незаметно отъесть десяток и более мегабайт памяти на многомегабайтной системе [36], а на системах со старенькими компьютерами можно использовать меньше памяти. Допустим на каждой зараженной машине отъедается, в среднем, 10Mb ОЗУ. Таким образом, при объемах «ботнета» порядка 300 тысяч компьютеров[37] количество доступного ОЗУ составляет 3 терабайта. Однако, с учетом необходимых накладных расходов (резервирование участвующих вычислительных единиц), существенно меньшая скорость доступа к распределенной памяти выигрышь может быть не столь значителен или вовсе может отсутствовать - это требует математической оценки.

## 12.4 Псевдополезное ПО

Одним из возможных вариантов работы закладки на зараженном компьютере может быть сценарий, когда закладка выдает себя за антивирус - без реальной антивирусной работы выдает предупреждения о якобы присутствующих в системе вирусах с предложениями купить ее для лечения. У доверчивых пользователей таким образом могут быть не только зря потрачены деньги, но и украдены данные необходимые для работы с их кредитными картами. Возможен также выпуск ПО, которое действительно будет делать что-то полезное + участвовать в ботнете..

## 12.5 Spamming

Весьма выгодная деятельность, описанная мной в начале этой статьи в качестве примера зачаточной реализации rootkit (как спамерского узкоспециализированного бота). Спамерство как одна из движущих сил экономической оправданности разработки ПО для организации ботнетов обсуждается во многих источниках. Однако, следует учитывать, что если данный компьютер участвует в рассылке СПАМа это очень быстро будет выявлено - существует множество служб выявления спама, борьбы со спамом, включая регистрацию хостов в публично доступных базах данных. Жалобы на рассылку и прочее. Фактически спамеры постоянно покупают новые загрузки, как минимум два-три раза в меся. Т.е. рассылка спама с большой вероятностью повлечёт переустановку ОС пользователем. Что в свою очередь повлечёт потерю закладки. Также массовая рассылка спама достаточно быстро приведет к пристальному вниманию к самой закладке и значительным усилиям по реверсингу ее алгоритма, выявления управляющих серверов (с попытками блокирования серверов и попытками отследить управляющий серверами персонал, а через них выйти на создателя ПО с дальнейшим уголовным преследованием).

[36] редкий современный компьютер не имеет хотя бы 128 мегабайт памяти

[37] не самая большая сеть

## 12.6 Накручивание баннерной рекламы

Еще один вариант зарабатывания денег - пользователь регистрирует нормальный web сайт и размещает на нем баннерную рекламу. Боты периодически загружают сайт и таким образом накручивают счетчик посещений. Каждая загрузка банера дает небольшую денежную сумму, но поскольку посещений много, они постоянные и с случайных IP - уличить в мошенничестве организатора махинации крайне сложно.

## 12.7 влияние на торговые отношения на биржах

Было актуально на 2006й год для США. Позднее рекламу торгуемых лотов запретили. Возможно данный тип махинаций еще где-то действует. Суть в рекламе через спам определенного набора акций с предсказанием их роста или падения. Поскольку количество получателей такого спама огромно - график продаж меняется за счет поверивших рекламе и тех, кто хоть и представляет суть процесса, но все равно не прочь «сыграть в рулетку». Заказчик рассылки может, например, заранее закупить акции по низким ценам и сбросить их на пике роста. Выигрывает тот, кто успел сыграть на дельте до возврата цен в нормальный для данного типа акций диапазон. Заказчик рассылки выигрывает больше за счет больших объемов вложений в участвующие в афере акции.

## 12.8 Network analysis

Возможность получить из удалённой точки информацию о маршрутизации иногда бывает очень кстати. Вообще для разных IP-сетей некоторый хост может иметь разный список правил в своем firewall software, причем это может быть реализовано в том числе на уровне принадлежности к определенным автономным системам. В частности, многие зарубежные системы реализующие grey-listing («взвешивание» IP адресов) заведомо отдают свое мнение о сетях принадлежащих РФ как о потенциально-опасных.

## 12.9 File Grabbing

Передача файлов пользователя без его ведома - вполне доступная для rootkit задача. Даже если пользователь использует зашифрованные диски. [38] Вообще говоря любые файлы компьютера, на котором установлен rootkit помимо пользователя компьютера становятся доступны владельцу rootkit.[39]

[38] Однако, если используется шифрование и зашифрованные данные не подмонтированы на момент заражения - придется дождаться когда система сама (по требованию пользователя вводящего пароль или предоставившего носитель с ключами) обратится к зашифрованному диску, то есть дождаться входа пользователя в систему с монтированием криптодисков

[39] см. также 12.12.1 на стр. 60

## 12.10 Account Grabs

Современные пользователи чрезвычайно доверчивы и хранят на компьютерах очень много информации связанной с доступом в различные места, включая то, что не имеет отношения непосредственно к интернет. Владельцу rootkit доступна и эта информация. Кроме того rootkit может брать пароли непосредственно из памяти приложения когда оно запущено[40]. Кроме того закладка может считывать пароль в момент ввода (если пользователь не хранит пароли в легко расшифровываемом хранилище).

### Примеры

- пароли к icq и подобным internet pager'ам
- пароли к почтовым аккаунтам, в том числе пароли к webmail сервисам
- пароли к управляющим аккаунтам к различному оборудованию
- параметры банковских карточек
- пароли на доступ к различным системам электронных денег(paypal,webmoney и др.)
- пароли доступа к базам данных, в том числе удаленным базам данных
- pgp ключи

## 12.11 Traffic Sniffing

Некоторые компьютеры подключены к хабам, а не коммутаторам, что позволяет видеть трафик предназначенный другим компьютерам. Реализация такой функции возможна. Также (в индивидуальном порядке) возможно применение атак на коммутирующее оборудование для получения доступа к транзитному трафику[41] Однако в сетях крупных компаний часто используются интеллектуальные устройства коммутации и маршрутизации совместно с программами мониторинга, т.е. выполнение атак подобного рода в автоматическом (да и вручную) режиме может быстро выявить наличие заражения.

### 12.11.1 KeyBoard sniffing

Очень часто применяемый метод слежения за пользователем - запись всего что он набирает на клавиатуре. Помимо прочего так можно красть пароли во время ввода.

[40] Например, в Mozilla есть функция мастер пароль для хранения паролей к сайтам требующим ввода пароля. Пока не введен мастер пароль все пароли хранятся на диске в зашифрованном виде. Когда мастер пароль введен пароли расшифровываются и загружаются в память, которую, в свою очередь, уже может прочитать закладка

[41] Имеются ввиду атаки на ARP и STP.

## 12.12 User Stats

Возможно получать различную статистику по действиям пользователя. Например, список и статистику загружаемых программ (список, время, частота использования), историю его серфинга интернет (список, пароли доступа и прочее).

### 12.12.1 Personal Data

Очень часто пользователи доверяют компьютеру хранение личных данных. Разумеется, если информация есть в компьютере - она доступна для rootkit тем или иным образом.

**Доступность данных на зашифрованных носителях** Даже если пользователь хранит данные на зашифрованных дисках (BestCrypt, PGP-диск и прочее) они тоже могут быть доступны:

- Во первых, не всегда программа реализующая криптодиски позволяет ограничить доступ к ним на уровне приложений, а значит после монтирования зашифрованного диска файлы на нем доступны для просмотра и копирования любым приложением с достаточными локальными правами
- Во вторых, даже если программа организующая криптодиск ограничивает доступ к нему определенным набором программ - можно либо открыть собственный поток в разрешенном приложении или просто перехватив пароль при вводе эмулировать «законный» доступ.

Тем не менее, несмотря на то, что доступность данных с криптодисков для rootkit очевидна, необходимо тестирование доступных программных продуктов во избежание досадных ошибок из за которых пользователь сможет по неадекватному поведению приложений понять, что компьютер находиться под контролем.

## 12.13 Сбор информации с multimedia устройств компьютера

При доступе к системе на уровне OS/драйвера (ring0 access) не составляет проблем доступ к, например, видеокамере или фотоаппарату подключенным к компьютеру. Вопрос лишь в том чтобы написать соответствующий модуль к rootkit, который смог бы воспользоваться удаленным устройством.

**Список устройств которые могут быть использованы rootkit**
Такие устройства на мой взгляд следует поделить на те, к которым следует обращаться косвенно и те, к которым можно обращаться напрямую.

**Устройства, которые можно использовать непосредственно**

- микрофон, если подключен к звуковой карте

- usb video камеры, если подключены и поддерживаются rootkit(хотя возможно обращение через драйвер, если установлен в системе)
- удаленные (ethernet/wifi/IP) video камеры

При этом следует осознавать, что факт доступа к некоторым утройствам может журналироваться, поэтому желательно использовать косвенный доступ к удаленным устройствам (см. ниже).

Также устройства типа web камеры часто имеют светодиодную индикацию, по которой пользователь может понять, что камера используется не им. Для таких устройств нужно совмещать доступ с использованием устройства самим пользователем.

**Устройства, доступ к которым можно осуществлять косвенным образом** Под «доступ косвенным образом» подразумевается перехват результатов работы с устройством.

- Принтеры (любые) - возможно получать задания в момент их отправки на печать с последующей обработкой и возможной отправкой
- Сканеры(любые) - возможно получение результатов сканирования с последующей их обработкой и возможной отправкой
- Все устройства с которыми можно работать непосредственно

Необходимость работы с этими устройствами косвенным образом очевидна - обращение к таким устройствам заметно (индикация на корпусе, шум работы, движение механических элементов). Весьма желательно осуществлять косвенным образом и доступ к удаленным multimedia устройствам, поскольку обращение к ним может журналироваться. При необходимости можно использовать косвенный доступ и к устройствам которые можно использовать непосредственно.

## 12.14 ScreenShots

В числе прочего, возможно получать снимки экрана. Однако, следует учитывать, что при сборе информации средствами Windows API происходит затормаживание графического интерфейса, хоть и кратковременное (доли секунды), но все же заметное внимательному пользователю[42]. Таким образом, если данный тип payload актуален, есть смысл подумать о реализации snapshot'ов «собственными силами» ставя целью, в первую очередь, не качество полученного изображения[43],

[42] Один из моих знакомых озаботился проверкой наличия закладок на своем компьютере именно благодаря таким вот симптомам и, действительно, обнаружил на домашнем шлюзе трафик сгенерированный закладкой

[43] четкость, цветность

а в первую очередь, незаметность процесса «фотографирования» desktop'а и малые размеры полученных «фото» .

### 12.15 Zero Knowledge System (zks) any trafic relay

**Обзор** Существовала когда то контора которая организовывала платный сервис доставки пакетов с гарантией анонимности. Ныне контора перепрофилировалась и больше таких услуг не поставляет - есть мнение - «большой брат настоял». Идея проста - каждый пакет шифруеутся на 3х ключах, отправляется на ближайший сервер сети zks . Сервер расшифровывает пакет, отправляет следующему известному ему серверу. Не зная адреса и порта назначения. Второй сервер делает то же самое. Также не имея понятия куда это все идет и откуда. Далее только на 3м сервере пакет отправляется уже на хост назначения.

Резюме: очень сложно выяснить пути пакета, отправителя и получателя, поскольку сервера раскиданы на разных континентах и под разной юрисдикцией.

**Подробнее** Приведу цитату из доступных опубликованных материалов:

Цитата:

=========

Одним из интересных средств обеспечения конфиденциальности является система Freedom ("Свобода"), разработанная канадской корпорацией Zero-Knowledge Systems. Система Freedom предназначена для анонимного просмотра страниц в Internet, обмена электронной почтой и участия в конференциях Usenet (группах новостей). Система функционирует на базе специальных серверов, разбросанных по всему миру. Когда кто-то хочет послать сообщение в Internet, просмотреть веб-страницу или принять участие в другой электронной транзакции, зашифрованное сообщение посылается с компьютера этого человека на один из серверов Freedom. Первый сервер пересылает сообщение на второй сервер, который, в свою очередь, пересылает его на третий, который, наконец, отправляет его по назначению. Каждое отправляемое сообщение зашифровано три раза, с последовательным использованием ключей серверов. Устройство системы не дает возможности человеку, перехватывающему сообщения (или имеющему контроль над одним из серверов Freedom) одновременно узнать и личность отправителя сообщения, и его содержимое. Фактически, Zero-Knowledge разместила сервера по всему миру, чтобы максимально затруднить для отдельно взятого правительства возможность изъятия содержимого всех трех серверов, задействованных в пересылке конкретного сообщения.

=========

# 13 Ограничения применения rootkits

В этой главе рассматриваются ограничения накладываемые на rootkit, как ограничения естественного характера(сеть, ПО, железо), так и ограничения связанные с средой исполнения (маскировка) .

### 13.0.1 Естественные ограничения

RootKit не панацея, и за кофе бегать не умеет..

Очевидно, что RootKit не сможет:

- дать информацию о действиях пользователя не соотносящихся с компьютером.
- дать информацию которой на компьютере пользователя нет [44].
- Дать больше информации в единицу времени, чем позволяют установленные для данного RootKit естественные или програмные ограничения [45]

### 13.0.2 Ограничения среды исполнения

Есть ряд существенных ограничений, которые должны соблюдаться с целью маскировки деятельности закладки, в особенности сетевой ее деятельности.

Очевидно, что из за того, что это будет очень легко заметить на непривычном поведении компьютера, rootkit не следуюет использовать чтобы:

- получить представление о действиях пользователя в «режиме реального времени»
- получить трансляцию video/audio с мультимедиа-устройств зараженного компьютера в режиме реального времени
- получать в режиме реального времени снимки с экрана
- получить полный дамп трафика проходящего через сетевые интерфейсы

Хочется заметить, что практически все из перечисленного в ограничениях среды исполнения реализовать можно, но делать этого не стоит, поскольку обнаружение последует очень быстро, ставя под угрозу всю систему.[46]

[44] доступ к удаленной информации не всегда сопровождается с передачей на него этой информации, наоборот, чаще всего, доступ осуществляется посредством некоторого интерфейса к удаленной системе(например web-interface'a), когда локально доступны лишь трудносопоставимые с информацией движения и клики мышью.

[45] в простейшем случае - невозможно передать информацию быстро на медленном канале

[46] и дело не в том, что описанная клиент-серверная система как либо особенно уязвима к обнаружениям - наоборот, прилагается масса усилий затруднить получение полной картины по одному пойманному rootkit, - дело в том, что принципы функционирования сети rootkits - маскировка, а значит неправильное использование = создание себе проблем.

# 14 Глоссарий

**СОРМ** - Система Оперативно Розыскных Мероприятий. Для провайдеров соблюдение требований СОРМ обязательно.

**компьютер, машина, комп, ПК, ПЭВМ, ЭВМ** - синонимичные понятия, которые, надеюсь, не требуют пояснений. Тем не менее, в данной статье (в том числе в этом глоссарии) термин компьютер используется иногда для описания любого устройства существенно умнее калькулятора (то есть имеющего возможность обеспечить работу операционной системы, пусть даже специализированной, как, например, CISCO IOS. ) ( alternate4fun: computer - a device designed to speed up and automate errors. )

**ИТ, IT, information technologies** - информационные технологии. Термин очень широко распространившийся в современности и не менее широко обобщаемый. Зачастую под IT понимают вообще все что связано с компьютерами и, в частности, компьютерными сетями.

**тревожное событие, алерт, alert** - в контексте этой статьи - некоторое событие которое воспринимается системой защиты как тревожное, при наступлении такого события могут быть произведены какие либо действия (например изменение правил firewall).

**ОПГ** - Организованная Преступная Группа .

**interface, интерфейс** - набор характеристик для взаимодействия между чем либо. Например, для взаимодействия человека и компьютера чаще всего используется интерфейс в виде совокупности монитора, манипулятора типа «мышь», клавиатуры и набора программ для обслуживания событий (нажатие кнопки на клавиатуре, например) с этих устройств. В свою очередь, человек, для работы с этим интерфейсом,должен иметь интерфейс, как минимум, в виде рук и глаз . В IT контексте под интефрейсом подразумевается, обычно, одно из следующих:

1. внешний вид и характеристики(ток, сопротивление, вольтаж )разъемов на устройстве (hardware context)
2. набор правил работы с программой реализуемых ее внешним видом - меню и т.п. (user context)
3. набор функций приложения или операционной системы доступных для программиста

Примеры интерфейсов:

1. сетевые интерфейсы (плата ethernet, modem(в том числе adsl))
2. физические интерфейсы (ethernet (IEEE802.3), wifi(IEEE 802.11b, IEEE802.11g)), ps2, COM(RS232)

3. пользовательские интерфейсы (внешний вид и средства управления/настройки любых програм, например IE, DrWEB,NortonComander)
4. програмные интерфейсы (набор API для работы с экраном, клавиатурой, мышью)

**виртуальная машина, virtual PC** - программа способная эмулировать компьютер.

**уязвимость, vulnerability** - возможность использования программы не по назначению, чаще всего подразумевается использование либо без ведома пользователя программного продукта, либо, плюс к тому, ещё и использование для получения дополнительных привилегий в системе разграничения полномочий операционной системы в которой работает программа с уязвимостью.

**носитель, зомби** - заражённая машина или операционная система (в зависимости от контекста).

**anti-forensics** - в контексте этой статьи, некие способы затруднения анализа закладки в процессе расследования инцидентов связанных с нарушением политики безопасности, в т.ч. в расследованиях связанных с попытками применить к владельцам закладки меры предусмотренные законодательством.

**offline, оффлайновый** - термин подразумевает, в первую очередь, отсутствие подключения к сети. Часто применяется по отношению к программе до запуска (например, когда диск компьютера подключается к другому компьютеру для исследования его содержимого, все программы на подключаемом диске находятся в offline'овом режиме).

**online, онлайн** - термин подразумевает возможность взаимодействия с сетью, часто используется по отношению к программе, подразумевая, что программа запущенна.

**гипервизор, hypervisor** - виртуализатор, програмный или програмно-аппаратный комплекс позволяющий исполнять несколько операционных систем на одном физическом компьютере, см:

`http://en.wikipedia.org/wiki/Hypervisor`

**protocol, протокол** - набор соглашений об обмене информации. В it-контексте подразумевается обычно протокол обмена данными по компьютерной сети или между программными модулями (в т.ч. определяет объем данных и их логическое представление на каждых приёме/передаче). То есть протокол определяет форму запрос/ответ или, иными словами, как понимать данные, которые приходят по сети и как их в сеть «говорить» (в общем-то, человеческая речь это тоже протокол)

**smtp** - simple mail transfer protocol, протокол для передачи почтового трафика.

**сервис,service** - практически то же самое, что и «в миру», но по отношению к приему информации. Если программа может предоставить пользователю или другой программе информацию по приходу запроса, то эта программа предоставляет сервис. Название сервиса чаще всего созвучно названию протокола который используется для приёма и отправки запросов.

**сервер, server** - с точки зрения пользователя - «какой-то» компьютер, стоящий «где-то» (хоть в соседней комнате, хоть на другом континенте), к которому можно обратится по сети при помощи той или иной программы. В общем случае термин применяется как к компьютеру в целом, так и к программе, которая на нем установлена (работает) и предназначена для обслуживания клиентов, т.е. «предоставления сервисов». Исторически сложилось, что аналогично сервисам и клиентам, сервера именуются созвучно названию протокола - например http-сервер. Бывает также, что слово сервер опускают, как бы, подразумевая его. Например когда говорят «прокси» подразумевают прокси сервер.

**C&C, управляющий сервер** в контексте данной статьи - сервера (возможно не единственный) для управления ботнетом.

**botnet herder** такой «ярлык» часто встречается в сети, когда имеют ввиду либо владельца ботнета, либо административный персонал им управляющий. Перевод herder на русский - пастух.

**клиент, client** - с точки зрения пользователя - он сам. В общем случае, в it-контексте, это либо компьютер, либо программа, которые сами сервисов не предоставляют, но к ним обращаются. См. «сервер», «сервис». Исторически слолжилось, что также как сервисы и сервера, клиенты именуются созвучно названию протокола - например http-клиент.

**атака, attack** - в IT - воздействие на некоторую вычислительную систему или сеть или набор данных или алгоритм их обработки. Атаки используются для:

1. нарушения работы данной системы;
2. модификации алоритма ее работы;
3. модификации или фальсификации данных;
4. получения данных или же модификации/фальсификации за счет недостатков заложенных в алгоритме обработки данных.

**DoS, Denial of Service, отказ в обслуживании** - широко распространенный вариант атак в сети, суть которых - вывод из строя некоторого сервиса для того чтобы он, временно, не мог обслуживать клиентов.

**DDoS, Distributed DoS, Distriuted Denial of Service** - DoS атака реализуемая большим количеством компьютеров через сеть. Чаще всего DoS осуществляется либо за счет превышения пропускной способности канала в интернет атакуемого сервера, либо за счет превышения максимального поддерживаемого сервером количества открытых соединений.

**транспорт, transport** - почти как и «в миру» - средство переноса, в it-контексте речь идёт о переносе данных. Сеть интернет задумана как иерархическая структура, в том числе это касается и передачи данных, а именно: классическая (как у письма) структура каждого протокола может быть представлена как две части - заголовок (с адресом) и данные (все что осталось); в свою очередь интерпретировать эти данные можно как угодно - в том числе можно придумать какой нибудь дополнительный протокол для обработки этих данных. Если данные передаются по протоколу «А» и для обработки полученных данных используется протокол «Б», то получается, что один протокол как бы вложен в другой. В таких случаях говорят что протокол А является транспортным для «Б» или «Б» ходит поверх «А».

**туннелирование, tunneling** - под этим термином понимается вложение данных одного протокола в другой (см. пример для транспорт). Туннели применяются как в «мирных целях» в повседневной практике (например для создания защищённых сетей установлением соединений поверх шифрующего протокола), так и в целях получения/передачи информации из сетей с ограниченным доступом[47] - например, если пользователь может легально ходить в интернет только по протоколу http, написав соответствующую программу он сможет сконвертировать в http трафик любой другой.[48]

**программный модуль, programm module** - исполняемый элемент, чаще всего - исполняемая программа в формате машинных инструкций нечитабельных человеком, (пример для dos и windows - setup.exe), однако в общем случае программный модуль это любой исполняемый набор команд (например autoexec.bat) .

**закладка** - некая сущность, которая превносится в объект «злоумышленником» для получения информации и, возможно, реализации других возможностей не предусмотренных нормальной работой объекта. Под закладкой может, в зависимости от контекста, подразумеваться как электрически пассивное(например действующее как резонатор или усиливающая антена), так и активное (передающее) устройство (например жучок). См. также «программная закладка».

**программная закладка** - исполняемый модуль, созданный для «нелегальной» работы на компьютере, как то: слежение за «законным» пользователем ком-

[47] чаще всего это делают в нарушение правил

[48] детальное описание этого процесса можно найти в интернете

пьютера, воровство идентификационной информации, использование процессорного времени без ведома владельца, воровство файлов владельца компьютера и прочее.

**ОС, Операционная Система** - «системный» набор программных модулей, обеспечивающих минимум возможностей для работы с их помощью других программ и, собственно, пользователя. Типичные примеры: MS DOS, Free DOS, MS Windows 3.11, MS Windows 95, MS Windows 98, MS Windows XP, MS Windows 2000, Debian Linux, Slackware Linux, Red Hat Linux, ALT Linux, ASP Linux, Solaris, IBM DOS, IBM OS/2, IBM AIX, Free BSD, Open BSD, Net BSD .

**ПО, Программное Обеспечение, софт** - синонимичные понятия, которые, надеюсь, не требуют пояснений.

**C/CPP, C, CPP** - класс родственных языков программирования.

**platform, платформа** - термин используемый как для определения сродства компьютерной техники (по типу и организации комплектующих), так и для определения сродства операционных систем (по набору и организации предоставляемых пользователю и программам возможностей, производителю ПО, ОС). Например UNIX и WINDOWS платформы. Платформы различных производителей зачастую имеют несовместимый формат исполняемых файлов, например исполняемый файл Linux не запускается в Windows.

**кроссплатформенный** - используемый на нескольких платформах, ПО работающее более чем на одной аппаратной платформе и/или операционной системе. Кроссплатформенность может быть либо на уровне компиляции (из исходного кода можно собрать бинарь более чем для одной платформы или ОС), либо на уровне выполнения - исполняемый файл может быть запущен на разных ОС (платформах).

**червь, worm** - самораспространяющийся, после запуска на компьютере подключённом к ЛАН или интернет, программный модуль, использующий для распространения сеть (в частности - интернет) и ошибки в ПО обрабатывающем информацию из сети на конечных точках сети (как серверах, так и рабочих станциях). В общем случае черви - кроссплатформенное явление. Разумеется, возникает не сам по себе, а как результат творчества программиста.

**LAN, Local Area Network, ЛВС, Локальная Вычислительная Сеть, локалка** - сеть из компьютеров, территориально расположенная на небольшом (географически) пространстве. Классические примеры - сеть офиса, дома, микрорайона.

**WAN, Wide Area Network, Глобальная Сеть** - сеть из компьютеров, территориально расположенная на большом (географически) пространстве.

Классические примеры - сеть internet, связанные внутренние сети корпораций с филиалами в разных странах.

**ARP, Address Resolution Protocol** протокол используемый в локальных сетях ethernet для преобразования IP адресов в адреса сетевых плат. Атаки с использованием протокола ARP на коммутирующее оборудование используют тот факт, что таблицы коммутации имеют конечный размер и после переполнения коммутации больше не производится - свитч начинает работать как хаб.

**internet, интернет** - глобальная (WAN) сеть публичного доступа, объединяет компьютеры частных лиц и организаций, используется для досуга, работы, рекламы, публикации информации технического и гуманитарного характера, как развлекательного, так и познавательного характера. Вряд ли есть область деятельности человека не упомянутая в интернет. Большинство организаций имеет подключения к сети интернет в том или ином виде с разной степенью ограничений на доступ к сети и извне сети - в зависимости от политики безопасности данной организации. Сеть интернет зачастую используется для объединения частных сетей организаций через так называемые туннели.

**RFC, Request for Comments** - общеупотребительное название стандартов на составляющие при помощи которых организована сеть Internet. RFC свободно доступны на соответствующих серверах интернета и я вляются, по сути, открытыми стандартами, в поиске и исправлении ошибок в которых может принять участие каждый желающий.

**вирус, virus** - самораспространяющийся после запуска на компьютере программный модуль, использующий для распространения модификацию других программных модулей, в том числе поставляемых в комплекте ОС и в комплекте с ПО сторонних производителей. В последнее время в связи с широким распространением ЛВС используют в том числе методы распространения червей (сеть). Разумеется, возникает не сам по себе, а как результат творчества программиста.

**троян, троянский конь** - типичное название программной закладки. В общем случае может распространятся автоматически - и как вирус, и как червь; классическое применение - индивидуальная инсталляция «вручную». Разумеется, возникает не сам по себе, а как результат творчества программиста.

**spam, спам** - незапрошенная получателем email корреспонденция рекламного характера.

**spamer, спамер** - человек рассылающий СПАМ.

**спамерский** - имеющий отношение к рассылкам СПАМа.

**адрес, address** - значение слова практически такое же, как и «в миру», однако в ИТ-контексте необходимо понимать следующее: в любой широко используемой компьютерной сети как глобального, так и локального масштаба используются «адреса», отличие от обычного смысла - адрес указывает не физическое положение компьютера, а его место в логике сети (её логической структуре). Адреса в IT области ориентированы не на людей, а на работу программ. Также, необходимо понимать, что поскольку компьютер может участвовать в различных сетях - адресов у него может быть несколько, кроме того они могут со временем меняться (типичный пример - дозвон поставщику услуг интернет - чаще всего при повторном дозвоне провайдер назначит уже другой адрес). Очевидно, что знание адресов требуется для практически любого обмена информацией.

**Mbit** - Мегабит, сокращение используемое для описания пропускной способности. Для сравнения типичное для Москвы пользовательское соединение с интернет типа «ADSL» предоставляет максимальную скорость приёма до 7Mbit, но в то же время максимальная скорость отправки всего лишь в 0.7 Mbit - именно поэтому ADSL это асимметричное (или же асинхронное) соединение.

**DNS** - domain name system - система трансляции из символьных имен вида www.pornoserver.com в адрес вида 121.123.23.54 и обратно. «Держится» на взаимодействии пользователей(запускаемых ими программ) с dns серверами и взаимодействии dns серверов между собой.

**adress resolving, adress resolution, ресолвинг, преобразование адресов, трансляция адр**
- процедура перевода из машинного представления адреса в легко понимаемый человеком. Не для всех адресов в сети зарегистрировано удобное для человека представление. Процесс resolving'а подразумевает обращение к серверу DNS.

**traffic, трафик** - объем передаваемых данных. Основной доход от деятельности провайдеры доступа в интернет получают за счет взимания платы за объем переданных данных через своё оборудование для подписчиков услуг (то есть оплачивается как трафик клиентов, так и трафик серверов) - расходы обсчитываются на каждый адрес (из списка адресов принадлежащих сети провайдера) используемый при работе в сети интернет.
- поток данных, чаще всего имеется ввиду объем данных в единицу времени (например загрузка канала), причем трафик именуется зачастую по названию протокола согласно которому[49] передаются данные, например http-трафик.

**функция** - некая часть программы, которую можно использовать (вызывать) многократно. Функции зачастую называют «процедурами». Современные

[49] исторически принято говорить «по которому»

ОС реализуют для программ возможность использовать как свои собственные функции, так и функции находящиеся внутри программ ОС и других программ.

**машинное представление** - способ хранения чисел в ячейках памяти компьютера.

**железо, harware** - любая аппаратная часть компьютера либо компьютер «отдельно от программ».

**ядро,kernel** - в контексте этой статьи - часть кода ОС, исполняемая с максимумом возможных прав, основная часть ОС, обеспечивающая её функциональность по работе с железом, выделению временных и прочих ресурсов программам, а также разделение прав пользователей.

**низкий, низкоуровневый** - в контексте этой статьи, чаще всего, - максимально приближенный к внутренним особенностям работы ОС или компьютера, например низкоуровневое программирование - либо программирование очень близкое к работе с ядром ОС, либо программирование в непосредственно инструкциях процессора (то есть фактически «в коде» (циферками то есть), а не командами, которые затем переводятся в код (цифры в том или ином машинном представлении)).

**rootkit, руткит, RootKit** - «самый продвинутый» вариант реализации программных закладок. Для того чтобы удовлетворять классу rootkit закладка должна реализовывать невидимость своего присутствия как для средств обнаружения программ включённых в комплект ОС, так и для утилит сторонних производителей. Такой уровень невидимости достигается путём перехвата внутренних функций ОС на уровне ядра (самый низкий, или «самый внутренний» уровень работы ОС - функций к которым обращаются как процессы прикладного режима, так и функции ОС).

**машинные коды(машинные инструкции)** - исполняемые команды процессора, самый низкий уровень инструкций компьютеру.

**компилятор, compiler** - программа создающая машинные инструкции из инструкций языка программирования с записью их в результирующий файл, который затем может самостоятельно быть использован для запуска.

**интерпретатор (interpretator)** - программа создающая машинные инструкции из инструкций языка программирования с последующим их исполнением в процессе своей работы по интерпретации, то есть инструкции интерпретатора не могут быть использованы для самостоятельного запуска в отсутствии интерпретатора.

**бинарь, binary, application, приложение, программа** - исполняемый модуль программы в виде инструкций процессора на данной вычислительной системе. Не путать со скриптами и исходными текстами, которые тоже есть

набор инструкций, но не для процессора, а для некоей программы (компилятора или интерпретатора). Синонимичные определения: программа, приложение.

**исходник, исходный текст, source code, source, сырец** - синонимичные названия текста программы, после обработки которого компилятором получается бинарник.

**library, библиотека, «либа»** - набор функций в программе или ОС, которые подразумевают возможность независимого использования . Библиотеки бывают статические (тело библиотеки вставляется в тело программы при компиляции, не требуется наличие библиотеки в ОС) и динамические (библиотека загружается из комплекта ОС).

**payload** - нагрузка, ради которой работает rootkit (или закладка вообще). См. 8

**сцена** В контексте этой статьи - понятие объединяющее людей интересующихся определенной информацией и создающих некий контент интересный, в основном, в рамках этого круга. Существует демо-сцена, vx-сцена и другие типа сцены. В рамках таких объединений люди творят ради искусства, самореализации и обмена идеями - «just for fun» .

**vx-сцена,вирусная сцена** Сцена вирмейкеров. Людей, которые пишут вирусы. Не для того (по крайней мере не обязательно) чтобы навредить другим, а «just for fun».

**29A** - журнал команды вирмейкеров. Широко известен в рамках вирусной сцены, см. сноску в 34

**алгоритм** - формализованная последовательность действий или событий поддающаяся описанию словестно и графически.

**реверсинг, reversing, reverse engenearing** - процесс восстановления алгоритма программы по её исполняемому файлу. Занятие трудоемкое и длительное, кроме того требующее высокой квалификации и знания низкоуровнего программирования (ассемблера и машинных кодов).

**отладчик, debugger** - программ для пошагового выполнения некоего кода, не обязательно машинного (в т.ч. бывают отладчики для скриптовых языков программирования) с возможностью просмотра значений переменных используемых в программе - используется для исправления ошибок (отладки) программ.

- программа для трассировки. Позволяет пошагово выполнять машинные инструкции трассируемого приложения с просмотром значений в регистрах процессора, памяти.

**трассировка, отладка, debugging** - процесс пошагового прохождения работы программы с использованием специальной программы отладчика(debugger'a), которая в свою очередь использует для перевода исполняемого файла в пошаговый режим специальные возможности ОС и процессора. Существуют методики определения программой того, что её трассируют, равно как и методы сокрытия этого факта от программы. Вообще говоря трассировка и отладка - не одно и тоже, но часто эти понятия используются именно в указанном контексте. Точные значения этих понятий таковы:
Отладка - этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки.
Трассировка - пошаговое выполнение программы с остановками на каждой команде (assembler) или строке (языки более высокого уровня).

**реверсер (reverser)** - человек занимающийся процессом reverse engenearing.

**jpeg, джипег** - название формата хранения картинок. - формат файла, который содержит сжатые данные обычно также называют именем JPEG, наиболее распространённые расширения для таких файлов .jpeg, .jfif, .jpg . формат jpg использует сжатие с потерями, при сохранении JPEG-файла можно указать степень качества, а значит и степень сжатия, которую обычно задают в некоторых условных единицах, например, от 1 до 100 или от 1 до 10. Большее число соответствует лучшему качеству, но при этом увеличивается размер файла. Обыкновенно, разница в качестве между 90 и 100 на глаз уже практически не воспринимается. Следует помнить, что побитно восстановленое изображение всегда отличается от оригинала.

**гиф, gif** - название формата хранения картинок. - формат GIF способен хранить сжатые данные без потери качества в формате до 256 цветов. Независящий от аппаратного обеспечения формат GIF был разработан в 1987 году (GIF87a) фирмой CompuServe для передачи растровых изображений по сетям. В 1989-м формат был модифицирован (GIF89a), были добавлены поддержка прозрачности и анимации. GIF использует LZW-компрессию, что позволяет неплохо сжимать файлы, в которых много однородных заливок (логотипы, надписи, схемы).

**ресурс** - некий сервис в сети доступный для использования всем или некоторым клиентам.

**web** - альтернативное название http протокола.

**web-ресурс** - http-ресурс, т.е. ресурс доступный по http протоколу, см. url, browser.

**web designer, веб-дизайнер** - человек занимающийся созданием web-сайтов (страниц в интернете).

**браузер, browser** - http клиент. смотрелка страниц в интернете.

**exploit, эксплойт, сплойт** - программный код для использования уязвимости.

**dropper, дроппер** чаще всего загрузка и запуск закладки осуществляется небольшим программным модулем, который включается в себя функционал эксплойта, downloader'a (загрузчика), executor (выполнение скачанного основного модуля).

**IE, Internet Explorer, ИЕ** - браузер используемый в windows системах «из коробки», то есть, если пользователь не инсталлировал другой бродилки по сети IE окажется бродилкой «по умолчанию». Де факто самый используемый в сети браузер. Де факто один из самых «дырявых» браузеров: под IE и расширения к нему пишется большинство exploits. Однако, это объясняется в первую очередь его популярностью - в других браузерах тоже находят уязвимости.

**урл, url, линк, ссылка, сайт, web-ресурс** - Указатель (ссылка) на ресурс в сети, - в частности - строка, которую, например, браузер (например IE, Netscape, Mozilla, Opera) может использовать как адрес.

**баннер** - небольшая картинка рекламного или информационного характера. Баннеры используются, для рекламы, в том числе контекстной, а так же для привлечения внимания и как легко заметные ссылочные элементы (в таком случае к баннеру «привязывается» web-дизайнером некий ссылочный url).

**баннерная сеть** - термин используется для группировки размещаемых в сети баннеров по некоторому общему признаку. В частности баннерные сети организуют компании предоставляющие услуги рекламного характера и услуги по номинированию рейтингов сайтов. За использование баннерной сетью может взиматься плата. Создание и использование бесплатных баннерных сетей может окупаться за счет показа рекламы в теле баннер (изображение обновляется с серверов баннерной сети). Баннерные сети работают поверх протокола http и его расширений.

**нулевое кольцо, ring3, ring0** - термины характерные для описания программ работающих на intel PC - совместимых процессорах (x86, начиная с 286). Указывают на уровень привилегий которые использует в работе программа. ring3 - самый бесправный код, ring0 - самый привилегированный. В ring0 обычно работает ядро ОС, в ring3 - пользовательское ПО и часть ПО ОС не требующая привилегий ring0.

**конфигурабельно, настраиваемо** - подразумевается возможность настройки поведения программы.

**система, системный** - в контексте этой статьи система это чаще всего ОС, системный - относящийся к встроенным в ОС функциям, базам данных, файлам.

**API, Application Programming Interface, АПИ** - набор функций предоставляемых ОС, в частности, для пользовательских программ. Бывают системные API (для использования модулями ОС) и пользовательские API (для использования прикладными программами).

**реестр** - в контексте данной статьи и вообще в контексте Windows совместимых систем: системная база данных, для использования которой каждое приложение может использовать пользовательское API. Для реестра windows характерно использование значений в машинном представлении (то есть значений не понятных не профессионалу). Реестр активно используется большинством windows приложений.

**драйвер, driver** - ПО из комплекта ОС или поставляемое сторонним производителем для поддержки работы ОС с некоторым устройством (железом).

**полиморфизм, полиморфность, мутации кода, пермутация** - подразумевается методика изменения бинарного кода основанная на том, что «на языке процессора можно сделать одно и то же разными словами». Практически представляет собой вариацию бинарного кода с заменой инструкции и блоков инструкций на эквивалентные по смыслу (действию).

**морф** - в контексте этой статьи - очередная версия «программы мутанта» которая делает то же самое, но выглядит по другому.

**фича, feature, возможность, наворот, фичастость** - подразумевается наличие неких «продвинутых», расширенных возможностей, по сравнению с чем то типичным. Может применяться как к софту, так и к железу.

**firewall, файервол, brandmouer, брэндмауэр** - средство контроля трафика в сети. Позволяет, в зависимости от уровня разработки («фичастости») продукта контролировать работу с транспортными протоколами (IP,udp), так и работу на уровне приложений (http, ftp клиенты и прочее).

**proxy,прокси, проксик, прокся** - некоторая программа, которая работает в режиме посредника при работе с сетью. Чаще всего software решение. Прокси классифицируются по типам протоколов. Наиболее распространены http прокси. Классический прокси требует настройки браузера для того чтобы его использовать.

**прозрачный прокси** - прокси, который работает с перенаправлением трафика и таким образом не требует настройки в браузере. Классическое использование - на серверах через которые организация подключена к интернет.

**personal firewall, персональный файрволл** - программное обеспечение контролирующее обращения программ к ресурсам сети, настройка доступна администратору компьютера, в простейшем случае настраивает тот кто пользуется. Настройка производится через графический интерфейс, чаще

всего по факту обращения программы в сеть или обращения к ресурсам компьютера из сети.

**botnet,ботнет** - сеть образуемая инфицированными компьютерами. Управляется владельцем ботнет. Чаще всего владелец ботнет является его создателем. Но при этом отнюдь не всегда владелец ботнета автор закладок при помощи которых botnet функционирует.

**маршрутизатор** - устройства обеспечивающие функционирование сетей, в частности, интернет. Их задача - разделение сетевого трафика в зависимости от типа и направления. В организации есть обычно как минимум один прибор выполняющий функции маршрутизатора - тот, через который организация соединена с internet. Самый примитивный пример маршрутизатора - ADSL modem. Маршрутизаторы работают, как минимум, с транспортными протоколами.

**коммутатор, свитч** - устройства обеспечивающие функционирование, в основном, локальных сетей. Их задача - доставка трафика между конечными рабочими станциями. Обладают минимальным интеллектом для определения, например, того из какого порта в какой порт направить трафик.

**STP, Spanning Tree Protocol** - протокол применяемый в локальных сетях с топологией звезда для выявления непозволительных в таких сетях закольцовываний на уровне кабельных соединений. Стандарт на этот протокол не содержит средств проверки источника данных протокола, что позволяет атаковать его.

**IDS, Intrusion Detection System, ай-ди-эс, ИДС** - программно-аппаратный комплекс для обнаружения в сетевом трафике как следов успешных вторжений / нарушений политики использования локальной сети, так и попыток вторжений/атак на устройства сети (компьютеры, маршрутизаторы, коммутаторы).

**IPS, Intrusion Prevention System, ай-пи-эс, ИПС** - практически то же, что и IDS, но с real-time реакцией на обнаружение атаки (блокировка трафика и прочие возможные способы реакции).

**сигнатура, шаблон** - в контексте этой статьи - некоторое правило, применение которого позволяет выделить тот или иной тип трафика или атаки при анализе потока данных в сети.

**порт** - часть адреса в транспортных протоколах. Наглядная аналогия - номер квартиры в многоэтажном доме.

**концентратор, хаб** - устройства обеспечивающие функционирование, в основном, локальных сетей. Их задача - доставка трафика между конечными

рабочими станциями. Не обладают интеллектом для определения, например, того из какого порта в какой порт направить трафик, в результате чего трафик «тупо» дублируется между портами устройства. Устройства относятся к классу устаревших и постепенно снимаются с производства и в экономически развитых странах почти не используются.

**проксируемый** - сервис, который предоставляется через прокси.

**firewalling** - процесс фильтрации трафика по заданным критериям.

**интерактивный** - в контексте статьи - подразумевается взаимодействие пользователей с сетью.

**serfing, net-serf, net-serfing, сёфринг сети** - просмотр информации из сети, в основном, с использованием браузера.

**honeypot, хонипот** - ловушка. В контексте статьи программа или компьютер, работающие в качестве приманки для «злоумышленников», наиболее продвинутые варианты honeypot'ов делаются профессионалами, причем есть проекты, где машина ловушка не просто пассивно ждет, когда на неё попадет незадачливый взломщик, а используется для простейших действий (например серфинга сети), что позволяет таким ловушками при определенном стечении обстоятельств получить закладки инсталлируемые автоматически только на компьютеры используемые в интерактивном режиме.

**интерактивный хонипот** - ловушка в виде компьютера, который используется человеком (или программой эмулирующей его поведение) для серфинга сети с целью подцепить закладки инсталлируемые пользователям во время серфинга.

**ACL, access control list** - обобщённое название ограничений которые могут в различных устройствах устанавливаться на некоторые действия совершаемые подконтрольным объектом (программой, потоком данных через сеть).

**adware** - термин используемый для названия программ, которые инсталлируются с каким либо прикладным ПО (иногда самовольно, иногда - согласно лицензионному соглашению о использовании программы, которое, впрочем, мало кто читает) и занимаются сбором и отправкой разработчику ПО различной не связанной с «шпионскими функциями» информации, например рапортуют статистику использования программного продукта и прочие не имеющие отношения к личности пользователя детали.

**spyware** - термин используемый для обозначения закладок, которые инсталлируются вместе с каким либо прикладным ПО (иногда самовольно, иногда - согласно лицензионному соглашению о использовании программы, которое, впрочем, мало кто читает) и занимаются сбором и отправкой разработчику ПО различной информации личного характера о пользователе. Например,

отсылают информацию о истории его серфинга сети. Используется, в лучшем случае, для контекстной рекламы.

**клиент-серверная архитектура** - термин применяемый для описания взаимодействия между объектами (в IT это компьютеры или программы), при котором, упрощённо говоря, один компьютер использует ресурсы другого. Чаще всего количество компьютеров(программ) серверов меньше, чем количество компьютеров клиентов(обычно, как минимум, в несколько раз).

**сниффинг, sniffing** - прослушивание, в частности, трафика или клавиатурного ввода или любого другого доступного программно устройства.

**контент** - наполнение. Чаще всего речь идёт о содержимом web-страницы.

**хост, host** - в общем случае - компьютер подключенный к сети.

**загрузка** - в контексте спама - покупка заражений через hosting-сервера. Вообще в IT, в зависимости от контекста - либо используется для обозначения процесса запуска программы (загрузка в ОЗУ компьютера), либо для обозначения использования некоторого устройства в смысле синонимичном слову «нагрузка» (загрузка процессора, загрузка канала доступа к интернет).

**covert channel, covered channel,hidden channel, скрытый канал** - в IT подразумевается канал передачи информации путем непредусмотренного(альтернативного) использования каналов связи. В качестве примеров использования скрытых каналов можно дать следующий, далеко не полный, список:

- туннелирование полное или частичное(один и более протоколов) IPv4/IPv6 в один из его «подпротоколов» - ICMP, UDP и др.
- туннелирование полное или частичное(один и более протоколов) IPv4/IPv6 в протоколы работающие на OSI application layer: http, dns и др.

bruteforce, полный перебор, лобовая атака - в криптографии/криптоанализе под этим подразумевается поочередная подстановка всех возможных паролей или ключей, которые могут быть использованы для расшифрования. Чаще всего это самый неэффективный метод, однако для некоторых алгоритмов зашифрования не опубликовано иных алгоритмов атаки.

fishing,фишинг получение обманом конфиденциальных данных пользователя. Данные в дальнейшем используются для различных махинаций, например - воровство денег из электронных систем оплаты. Довольно часто используется перенаправление (например трояном) на сайт внешне похожий или даже идентичный тому, куда пользователь вводит свои данные.

## Список литературы


[1] Брюс Шнайер
Прикладная криптография.
Протоколы, алгоритмы, исходные тексты на языке Си
Applied Cryptography. Protocols, Algorithms, and Source Code in C
Издательство: Триумф, 2002 г. Твердый переплет, 816 стр.
Тираж: 3000 экз.
Формат: 70x100/16

[2] Гарфинкель Симсон
Все под контролем: Частная жизнь под угрозой
Пер. с английского В Мяснянкина.
Екатеринбург: У-Фактория, 2003. - 432 с.
(Серия <<Кибертime/non-fiction>>), ISBN 5-94799-270-1

[3] Open System Interconnection (OSI) reference model
http://www.webopedia.com/quick_ref/OSI_Layers.asp

[4] Lance Spitzner
Презентации с сайта проекта "The Honeynet Project"
http://project.honeynet.org/misc/project.html

[5] Олег Артемьев, Владислав Мяснянкин
Опасные деревья в сетевых лесах
Журнал "LAN" 01/2002

[6] Botnets - the Killer Web App (2007)- ничего особенного, но хороша тем, что есть обзор CC и общеизвестных альтернативных способов. Также есть обзор имевшихся на 2007 ботов. Также будет полезна описанием методов работы anomaly detection tools ( в частности ourmon tcp wight & other). Также рассматриваются сандбоксы для исследования малвари.

Craig A. Schiller
Jim Binkley
David Harley
Gadi Evron
Tony Bradley
Carsten Willems
Michael Cross

Botnets: The Killer Web App
Syngress Publishing Inc, 2007

[7] «THE ART OF COMPUTER VIRUS RESEARCH AND DEFENSE» - неплохой обзор технологий используемых для обнаружения вирусного и прочего

зловредного кода и трюков используемых для того чтобы избежать этого в главе «Armored Viruses».

THE ART OF COMPUTER VIRUS RESEARCH AND DEFENSE
By Peter Szor

Publisher: Addison Wesley Professional
Pub Date: February 03, 2005
ISBN: 0-321-30454-3
Pages: 744

# 15 Предметный указатель

## Предметный указатель