RootKit RFC, part 2
Описание технических требований к качественному rootkit, часть 2.
Аналитическая статья.
Version 1.0 .

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 Цель

Данная статья ставит своей целью техническое описание требований к качественному rootkit - архитектуре, протоколам, реализации. В идеале - это документ класса RFC.

Теоретическое обоснование в первой части данного RFC. Тут только техническая детализация

Сделано:

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:

Аналогичное описание для peer2peer модели взаимодействия.
Также см. стр. ??, ?? .

1.2 Использование.

Статья выложена в паблик с целью создания полноценного rootkit RFC.

Статья может быть полезна для специалистов IT, в особенности тех из них, чья работа связана с обеспечением безопасности IT инфраструктуры, либо с ее тестированием.

Статья развивалась усилиями членов клуба, а также «усилиями третьих лиц».

Email текущего maintainer'а 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. Заказчика практической реализации rootkit
2. программиста или группы программистов, реализующих закладку класса «rootkit» для сетей общего пользования, в частности, интернет
3. Любых заинтересованных лиц - как обзорный материал по требованиям, которым должен удовлетворять качественный rootkit.

Глоссарий
Глоссарий пока отсутствует. Статья подразумевает,что вы понимаете используемую терминологию. Однако Вы можете прислать ваши замечания/предложения о том что требуется внести в глоссарий.

1.2.1 Порядок чтения

Предполагается, что Вы заинтересованы прочесть все. Также предполагается, что Вы прочли теоретическое обоснование (первую часть этого RFC).

1.2.2 Замечания по главам

Для того чтобы не было неоднозначностей наборы требований могут характеризоваться так:

- «Обязательно» (английский синоним "MUST")
- «Желательно» (английский синоним "SHOULD")
- «Возможно» (английский синоним "MAY")

?? на стр. ?? может быть интерсна только тем, кто участвует в развитии документа, она полностью посвещена редакторской правке и ведению версий документа, по смысловой нагрузке статьи там практически ничего нет.

Copyright
Текущий copyright - Олег К. Артемьев. См. также описание ©в первой части этого RFC.

Спасибо

Хочется сказать спасибо:

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

2 Требования к языкам программирования

2.0.3 Логика(макро-алгоритм)

- Для облегчения дальнейшей модификации, оптимизации, исправления ошибок, а также с целью возможности портирования логика работы сервера и клиента должны быть реализованы на языке среднего/высокого уровня, предпочтительно одном из диалектов C/CPP.
- При написании когда необходимо иметь ввиду портируемость, как клиента, так и сервера. То есть как минимум межфункциональное взаимодействие должно быть реализовано с возможностью замены функций, блоков функций.

2.0.4 Низкий уровень

клиент

Процедуры низкого уровня на клиенте могут быть написаны на языке ассемблера. Возможна реализация, либо в виде ассемблерных вставок в C-функции, либо в виде отдельных функций assembler'a с интерфейсом работы с языками среднего уровня. В любом случае Ассемблерные вставки в исходный код должны быть прозрачными для замены.

сервер

Во избежание проблем с портированием серверная часть должна быть полностью реализована на языке среднего/высокого уровня. Предпочтительно C/CPP.

2.1 Требования к исходному коду

В общем-то эти требования можно применить к любому программному проекту. Их удовлетворение конечно не обязательно, но должно существенно упростить и сократить разработку.

1. возможность переноса части функционала в библиотеку
2. реализация одного функционала одним кодом
3. написание кода с учетом возможности будущего портирования на другие платформы
4. Минимизация объёма кода написанного на ассемблере по сравнению с C/CPP

3 Требования к архитектуре

Обзор главы.

Описываются требования к реализации архитектуры ботнета - серверной и клиентской части (для не peer2peer схемы). Часть пунктов отсюда естественно будут дублироваться в разделах требований к закладкам и серверу. Тут только архитектурные требования. Частности дальше.

3.1 Глобальные требования

1. модульность
2. event-driven модель
3. поддержжка распределения нагрузки управляющей части
4. вынос ресурсозатратных вычислений на клиента
5. поддержжка профилей(схем) работы
6. поддержка взаимных проверок (бот-сервер, сервер-бот)
7. хранение и передача данных протокола в кросплатформенном формате

Event driven модель - как показывает практика она быстрее многопоточной модели и проще программируется.

поддержжка профилей(схем) работы - в т.ч. по уровням доверия (ступени инкубационный, тестовый, рабочий периоды), самостоятельное профилирование сетевых возможностей закладки в среде функционирования (в т.ч. детект смены сети мобильным пользователем), поддержка профилирования возможностей закладки сервером.

вынос ресурсозатратных вычислений на клиента - когда это не противоречит безопасности.

хранение и передача данных протокола в кросплатформенном формате - независимость от litle/big endian, будет актуально при портировании - меньше придется переделывать.

3.2 Требования и к закладке и серверу

1. Использование в качестве транспорта стандартных протоколов на стандартных портах
2. Шифрование обмена данных
3. Скрытие канала передачи информации
4. Поддержка режима инкубационного периода
5. Поддержка режима тестового периода
6. Все что перечислено в глобальных требованиях

3.3 Индивидуальные требования

3.3.1 Требования к закладке

1. работа, как минимум, части закладки в ring0
2. модульность. Как минимум - деление dropper/loader/payload
3. шифрование/расшифрование частей тела rootkit в памяти («на лету»), части - по ключу с сервера
4. ступенчатая инсталляция
5. in memory only payload
6. реализация методов обнаружения запуска «под отладчиком»
7. профилирование сетевых возможностей в среде функционирования
8. поддержка профилирования возможностей закладки сервером

профилирование сетевых возможностей в среде функционирования подразумевает выявление разрешенных протоколов для работы с интернет.Необходимо, например, для того, чтобы в зараженной сети где запрещен, допустим, ICMP трафик, наружу не могла быть запущена icmp DDoS атака.

1. пассивное - по сниффингу типов протоколов используемых пользователем
2. активное - по результатам запуска тестов на соединения
3. по результатам запроса к серверу

поддержка профилирования возможностей сервером подразумевает поддержку выставления переменных описывающих тип использования данного экземпляра сервером с использованием фильтрации на этой основе исполнения команд выдаваемых всему ботнету.

3.3.2 Требования к серверу

1. Event driven модель

4 Требования к протоколу.

В этой главе описываются требования к протоколу работы в рамках ботнета.

4.1 Глобальные требования

1. Формализуемость без неоднозначностей.
2. Должен использовать шифрование.
3. Должен использовать скрытые каналы передачи данных.
4. Должен позволять масштабируемость инфраструктуры поддержки сети.
5. Должен позволять профилирование закладки сервером
6. Должен иметь возможность отчета о времени в обе стороны
7. Должен иметь проверки на подмену сервера.
8. Должен иметь проверки на подмену клиента.
9. Поддержка индивидуального и массового командного режима

4.1.1 Шифрование

Шифрование должно использоваться как для трафика, так и для payload.

Обязательно

- Шифрование: Данные (в сеть/из сети) не должны передаваться в открытом виде.
- Должен использовать индивидуальный ключ шифрования трафика для каждой закладки
- Прием/передача зашифрованных данных должны идти через скрытые каналы передачи информации

4.1.2 Скрытые каналы передачи данных

- Реализация работы на, как минимум, двух каналах скрытой передачи информации[1]

[1] позволит резервировать один из каналов для информирования о нештатных ситуациях

4.2 Требования и к закладке и серверу

1. Использование в качестве транспорта стандартных протоколов на стандартных портах
2. Шифрование обмена данных
3. Скрытие канала передачи информации
4. Поддержка режима инкубационного периода
5. Поддержка режима тестового периода
6. Поддержка уникальных идентификаторов для модулей системы
7. использование уникальных идентификаторов позволяющих индексировать не менее чем двойной объём IPv6 сети

4.2.1 Требования к серверу

1. Event driven модель обработки событий[2]
2. Набор проверок по отношению к клиентам
3. Защита от DoS атак
4. Классификация клиентов по задачам, которые на них можно выполнять
5. Классификация клиентов по уровню доверия
6. Достаточное быстродействие [3]
7. Поддержка включения журнала событий для каждого клиента
8. Поддержка детализированного журнала событий для каждого подозрительного клиента
9. Отдельный алгоритм работы с пойманными ботами

Защита от DoS атак в протокольном контексте подразумевает построение протокола таким образом, чтобы обнаружение псевдоботов (т.е. реверсеров и закладок не корректно работающих с протоколом - закладок под контролем реверсеров) могло происходить на как можно более ранней стадии обмена, что позволит минимизировать затраты сервера и уменьшит вероятность быстрого DoS. Также часть данных может кэшироваться на закладке для минимизации нагрузки на сервер и исключения существенного увеличения определенных типов трафика в интернет.

[2] как показывает практика - быстрее многопоточной модели

[3] как показывает практика монстры вроде apache нагрузки в 200 тысяч ботов не выдерживают несмотря на то, что мощности сервера и ширина канала позволяют обработку существенно большего

4.2.2 Требования к закладке

Прочее

1. Синхронизации времени запуска payload на закладке с точкой отчета.
2. Должен содержать алгоритм генерации уникальных идентификаторов покрывающих не менее двойного объёма IPv6 сети

Список литературы

[1]
NetHack club teamwork
RootKit RFC, part 1
Рекомендации по созданию програмных закладок.
Аналитическая статья.
Version 1.1 .