Скрытые каналы в протоколе HTTP

[Хакерство в Интернете]

В любом наборе информации, будь то исполняемая программа, графическое изображение или сетевой протокол есть пути переноса дополнительных «скрытых» данных. Такая возможность есть и практически на всех уровнях модели OSI. Широко известны инструменты туннелирования, использующие служебные заголовки протоколов сетевого уровня TCP/UDP. Однако практическое применение таких туннелей ограничено из-за их низкой пропускной способности. Более широкие перспективы для построения скрытых коммуникационных каналов основываются на использовании прикладных протоколов. Здесь есть возможность быстрой передачи больших объемов произвольной информации в качестве полезной нагрузки самого протокола, а не кода внутри его служебных заголовков. Одним из таких протоколов является HTTP.

Основное поле для использования скрытых каналов это локальные сети, имеющие доступ в Интернет. Как правило, работа с сетевыми сервисами за пределами домашней сети жестко регламентируется различными системами контроля доступа. Эта статья является практической, в ней мы рассмотрим два способа использования протокола HTTP для туннелирования произвольных данных протоколов сетевого уровня. Оба метода могут быть использованы как для доступа из локальной сети за ее пределы, так и для обратной задачи.

CCTT

CCTT – (“Covert Channel Tunneling Tool”) Инструмент Туннелирования в Скрытых Канналах. CCTT позволяет создавать коммуникационные каналы через системы контроля сетевого доступа с потоками данных, которые позволяют: получить shell на внешнем сервере из внутренней сети. предоставить shell с сервера находящегося во внутренней сети внешнему серверу. установить канал позволяющий TCP соединения (SSH, SMTP, POP и т.д.) между внешним сервером и компьютером во внутренней сети.

Для использования CCTT необходимо установить его клиентскую часть в локальной сети и серверную во внешней сети. В качестве канала передачи между ними могут выступать TCP, UDP соединения и TCP соединения с использованием метода HTTP CONNECT через локальный HTTP прокси сервер или через цепь HTTP прокси серверов. Одна из интересных особенностей CCTT это работа в режиме реверсивного прокси. CCTT клиент устанавливает соединение с CCTT сервером (TCP, UDP, HTTP CONNECT) и регистрируется на нем как реверсивный прокси. CCTT сервер, находящийся за пределами локальной сети, принимает соединения от клиентов, находит соответствующую запись в таблице зарегистрированных реверсивных прокси и использует уже открытое соединение с соответствующим прокси, как канал для передачи данных. CCTT клиент, принимает эти данные и устанавливает соединение с нужным сервисом внутри локальной сети. Так же есть возможность кодировать поток данных между CCTT клиентом и сервером.

FIREPASS

В большинстве локальных сетей использование метода HTTP CONNECT запрещено на внутреннем HTTP прокси сервере. В этом случае может работать технология, реализованная в проекте Firepass. Здесь используется метод HTTP POST. Firepass клиент располагается на сервере внутри локальной сети и «слушает» несколько TCP/UDP портов. Как только устанавливается соединение с ним, Firepass клиент отправляет HTTP/1.1 POST запрос на веб сервер во внешней сети. Он может сделать это как напрямую, так и с использованием локального HTTP прокси сервера. Ресурс, на который он отправляет POST запрос, это Firepass сервер. В заголовке первого HTTP запроса указываться адрес, порт и протокол сервиса во внешней сети с которым необходимо установить соединение. Firepass сервер, получив первый запрос, порождает новый процесс – “менеджер соединения”, который и будет устанавливать дальнейшее соединение со внешним

сервисом.. Далее Firepass клиент с некоторым интервалом отправляет серверу HTTP POST запросы с TCP/UDP данными, полученными от своего клиента. Эти данные отправляются на целевой сервис с использованием «менеджера соединения». Данные же полученные от сервиса проходят обратный путь и возвращаются Firepass клиенту в качестве HTTP ответа на очередной POST запрос. В некоторых случаях, если позволяет конфигурация сети и DMZ возможно проникновение извне в локальную сеть через, например, корпоративный веб сервер. В этом случае Firepass сервер работает на корпоративном веб сервере, а Firepass клиент общается с ним из внешнего мира. Одно из достоинств технологии использованной в Firepass это множество вариантов размещения его серверной части. Требуется только веб сервер и возможность устанавливать с него исходящие TCP/UDP соединения.

ЗАКЛЮЧЕНИЕ

Приведенные здесь методы, показывают только несколько вариантов использования протокола HTTP в качестве среды передачи произвольных данных. В целом же можно сказать, что если в локальной сети есть клиент с возможностью отправлять/принимать HTTP запросы (методы GET, POST, CONNECT), то для него так же всегда есть возможность использования HTTP как туннеля для протоколов сетевого и прикладного уровня.

alex@gray-world.net