ЗОМБИ-СКРИПТИНГ

ИСПОЛЬЗУЕМ BEEF ДЛЯ ПРОДВИНУТЫХ XSS-АТАК

Михаил Фирстов,
HeadLight Security
@cyberpunkych
cyber-punk@xakep.ru

XSS-атаки (cross-site scripting) уже давно вышли за рамки угона сессии через document.cookie на сниффер злоумышленника. Думаю, многие хорошо помнят даже различные онлайн-сервисы, которые позволяли с легкостью вставлять нужный пейлоад и ждать, пока на e-mail придет уведомление с данными жертвы. Сейчас на дворе 2016 год, так что предлагаю оставить все вышеописанное где-то в 2007 и вернуться в реальность. Сегодня речь в статье пойдет о популярном инструменте, который помогает выжать максимум из простого внедрения JS-кода. Итак, знакомься — BeEF.

BeEF — это фреймворк, позволяющий централизованно управлять пулом зараженных через XSS клиентов, отдавать команды и получать результат. Он работает следующим образом:

- злоумышленник внедряет на уязвимый сайт скрипт **hook.js**;
- **hook.js** сигналит C&C (BeEF) о том, что новый клиент онлайн;
- злоумышленник входит в панель управления BeEF и удаленно «рулит» зараженными браузерами: исполняет пейлоад и получает ответ.

Жертве достаточно выполнить в своем браузере hook.js, и она станет очередным «зомби», которому можно будет посылать различные команды, подсовывать вредоносные экзешники и так далее. А от злоумышленника требуется просто запустить beef-xss и открыть в браузере панель управления, в которой, собственно, все самое интересное и происходит.

BeEF «из коробки» содержит уйму встроенных техник, эксплойтов, плагинов и значительно облегчает одновременную работу со множеством клиентов. Конечно же, зараженный пользователь может покинуть страницу с внедренным скриптом, но специально для этих целей существует атака Man-in-The-Browser, которая позволяет следить за всеми действиями клиента в контексте одного домена, пока он остается на сайте или собственноручно не поменяет адрес в строке URL.

УСТАНОВКА BEEF

Начать надо, конечно, с установки. Я рассматриваю ситуацию, в которой у нас есть своя впс/вдс где-то в облаках с белым айпишником и четырнадцатой Бубунтой:

```
root@myownvpn:~# uname -a
Linux myownvpn 3.13.0-52-generic #85-Ubuntu SMP Wed Apr 29 16:44:17
UTC 2015 x86_64 x86_64 x86_64 GNU/Linux
```

Действуем по официальной инструкции, которую можно найти в репозитории на гитхабе. BeEF написан на Ruby, поэтому установка стандартная для большинства небольших руби-приложений: rvm, ruby, bundler и **bundle install**. В общем, проблем возникнуть не должно.

Кстати, BeEF также включается в дефолтную поставку Kali Linux 2, поэтому если ты используешь творение Offsec'а, то BeEF у тебя уже установлен.

ИСПОЛЬЗОВАНИЕ

После запуска beef-xss открываем в нашем браузере ссылку **http://127.0.0.1:3000/ui/authentication** и логинимся как **beef:beef**. Отлично, у нас есть все, что нужно. Чтобы протестировать работу BeEF'а, предлагаю «захукать» свой же браузер, перейдя на страницу **/demos/basic.html**. Как видно в исходном коде, у нас подгружается тот самый hook.js, и теперь в разделе Online Browsers появился хост и список «захуканных» клиентов. Это и есть наши зараженные зомби.

Если выбрать нужного клиента и перейти на вкладку Commands, нашему взору предстанет широкий спектр различных техник, сплоитов, атак и трюков, которые можно использовать буквально за пару кликов.

СОЦИАЛЬНАЯ ИНЖЕНЕРИЯ

Для начала выберем что-нибудь простое и понятное — открываем раздел Social Engineering и попробуем провернуть фишинг логина в гугл-аккаунт. Для этого нам достаточно указать страницу, на которую будет перенаправлен пользователь, и задержку (delay) до редиректа. Заполняем (или, в нашем случае, оставляем как есть), нажимаем кнопку Execute — волшебным образом содержимое страницы меняется на знакомый и привычный глазу интерфейс авторизации в гугл-почте!

Притворимся, что мы жертва, и введем какие-нибудь данные. После нажатия кнопки Sign in нас перенаправляет на настоящую авторизацию Гугла и заодно перекидывает на указанную ранее в настройках страницу. Как можно увидеть, нажав в BeEF'е на отправленную команду, мы успешно перехватили введенные пользователем данные.

Фишинговая страница логина в GMail, показанная жертве с помощью BeEF

Перехваченные через BeEF данные от Гмейла

САМЫЕ ОСНОВНЫЕ КОМАНДЫ

Где же старый добрый document.cookie, который всем так нужен? Да тут же, под рукой — в разделе Hooked Domain найдется достаточно команд для почти мгновенного доступа к различной информации из браузера жертвы, в том числе и вожделенным «печенькам».

Этим несложным образом можно получить список подключенных на странице скриптов и ссылок или весь исходный код страницы. Отдельное внимание я советую уделить такой фиче, как замена ссылок — изменить всё так, чтобы каждая ссылка на странице вела на нужный нам адрес. Среди всех доступных команд есть даже перенаправление пользователя на ролик с песней «Never gonna give you up» — классика как-никак!

You've been rickroll'd!

ФИНГЕРПРИНТ

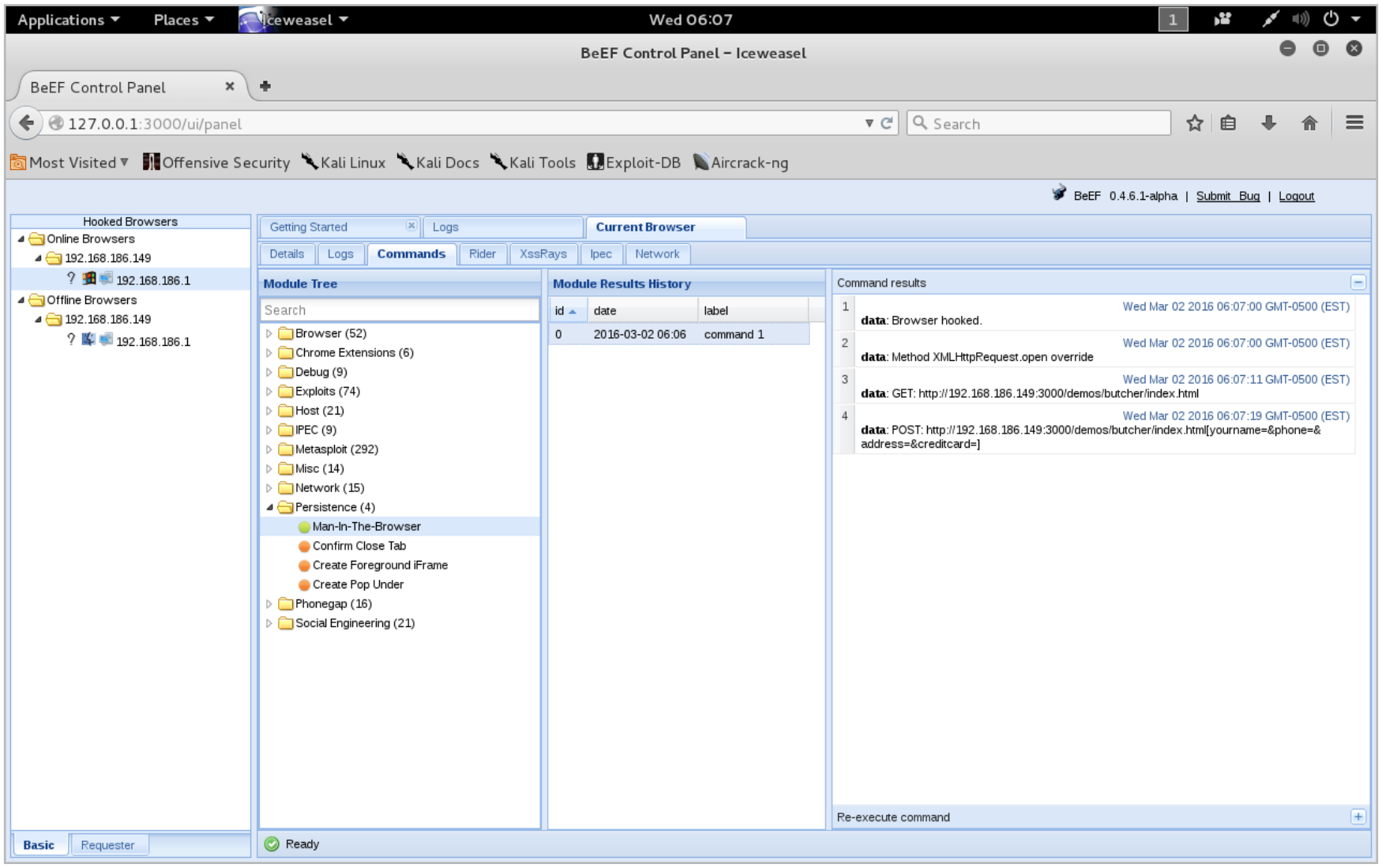
Помимо всего вышеописанного есть раздел Browser, в котором собрано множество полезных фич для максимального фингерпринта браузера жертвы. Можно узнать список установленных расширений в браузере, наличие LastPass, FireBug и т.д. С помощью команды «Webcam» можно попробовать подсунуть пользователю Flash-плагин для работы с вебкой, жаль только, что данная техника уже устарела, как и Flash в целом. Не буду перечислять и описывать все фичи, а посоветую тебе изучить их самостоятельно, заодно проверив актуальность в современных браузерах.

эксплойты

Следующий интересный нам раздел — Exploits, как ни странно. В нем найдется множество готовых векторов для эксплуатирования различных уязвимостей. Прежде всего я советую обратить внимание на сплойты для различных роутеров. Мы уже не раз писали, какие фатальные баги иногда содержат роутеры: часто от безобидных XSS'ок получается раскрутить их до полноценного RCE через CSRF. К слову, тот же сценарий повторяется и с многими другими устройствами, которые находятся в сети потенциальной жертвы, будь то NAS, камеры, локальные сервисы и так далее. Список эксплойтов, конечно, не может похвастаться зиродеями или огромным выбором, но ведь никто не запрещает писать кастомные плагины для бифа под свои уязвимости ;)

атака на браузер, сниффинг отправляемых данных

Одной из самых интересных атак при использовании XSS можно назвать Man-in-The-Browser. Она позволяет отслеживать различные действия инфицированного пользователя на всем ресурсе, а не только на странице с пейлоадом. Эффект достигается за счет фоновой подгрузки запрошенной страницы по клику и подмены DOM, без изменения реального location'а браузера (примерно как в рельсовом `turbolinks`). Естественно, эта фишка сработает только при соблюдении SOP: запрошенная страничка должна иметь тот же протокол, порт и домен, а иначе загрузки просто не произойдет.



BeEF отлавливает попытки перехода по другую страницу и подгружает данные динамически

Это крайне удобная фича, избавиться от которой можно только путем прямого изменения URL адреса в строке браузера. Как можно увидеть, в логах отображаются все действия, выполненные пользователем, такие как ввод и отправка POST/GET запросов, открытие других сайтов в новом окне и т.д.

РАБОТА С СЕТЬЮ

Многие функции из секции Network могут не работать, например, детект социальных сетей у меня упорно отказывался работать. Подозреваю, что там использовалась баги/фичи, которые были прикрыты сервисами. Но не надо расстраиваться: всегда стоит помнить, что в некоторых ситуациях даже одна XSS в браузере жертвы может привести к компрометации всей домашней сети. Поэтому обрати внимание на различные сетевые сканнеры, представленные в BeEF'e. Как показывает практика, далеко не все из них будут работать в современных браузерах, но команда **`GET HTTP servers`** у меня отработала и честно нашла в локальной сети старенький Zyxel с веб-интерфейсом. Конечно, детект серверов очень простой и основывается на попытке подгрузить favicon с каждого из указанных в диапазоне апишников.

Еще советую посмотреть на такую замечательную функцию, как сканирование сети с использованием словаря DNS-имен. Кто знает, вдруг вы для удобства назвали свой роутер «router»? :)

MSF

Детект и фингерпринт внутренней сети это, конечно, замечательно, как и попытки использовать различные сплойты под роутеры со всякими CSRF/XSS/etc. Но что, если мы хотим стать крутыми хакерами и попробовать сразу пробить браузер жертвы? Нет ничего проще, ведь BeEF поддерживает использование Metasploit'a. Это, конечно, не приватные связки с 0day, но никто и не гарантирует, что у пользователя будет последняя версия браузера/плагинов :)

Чтобы подружить metasploit и beef, достаточно выполнить несколько простых действий:

1. Открыть главный **`config.yaml`** (у меня в Kali он был расположен по адресу **`/usr/share/beef-xss/`**), найти раздел **`Extension`** и у Метасплойта изменить значение параметра **`enable`** на **`true`**.
2. Открыть файл настроек плагина метасплойта (например, **`/usr/share/beef-xss/extensions/metasploit/config.yaml`**) и изменить параметры **`host`**, **`callback_host`** (остальное на твое усмотрение).
3. Запустить метасплойт и выполнить команду:

```
msf > load msgrpc ServerHost=192.168.186.149 User=msf Pass=abc123
```

Перезапуск BeEF'a — и нашему взору открываются несколько сотен модулей

метасплойта, которые можно применять по своему усмотрению. Давай рассмотрим простой пример использования связки BeEF + Metasploit. Возьмем модуль **browser_autopwn2**, настроим и запустим его.

```
msf > use auxiliary/server/browser_autopwn2
msf auxiliary(browser_autopwn2) > show options
                …тут указываем опции, например SRVHOST…
msf auxiliary(browser_autopwn2) > run
```

Для удобства использования и настройки в качестве URIPATH я указал /autopwn. Теперь идем в BeEF, выбираем раздел Misc и используем команду на добавление невидимого фрейма. В качестве адреса просто указываем что-то в духе **http://наш_ip:8080/autopwn**. После отправки команды видно, что в браузере жертвы фрейм с связкой подгрузился, а все остальное уже на совести метасплойта. Точно таким же образом можно использовать и другие модули Metasploit'а и SET'а (отличный инструмент широкого спектра возможностей для атак, связанных с соц. инженерией).

Атаки на расширения браузера

Еще один интересный и часто использующийся вектор атаки — атака на расширения пользователя. В случае успеха она гарантирует нам гораздо большие возможности, нежели простое внедрение скрипта на страничку сайта, ведь у расширений куда больше прав. Специально для этих целей был создан аналог BeEF`a, который в итоге перерос в самостоятельный полноценный фреймворк — Chrome Extension Exploitation Framework, или просто ChEF. Среди его возможностей выделяются такие фичи, как:

- мониторинг открытых вкладок в браузере;
- глобальное выполнение JS кода на любой из вкладок (global XSS);
- чтение http-only кукисов;
- получение и изменение истории браузера;
- снятие скриншотов окон браузера;
- доступ к файловой системе через **file://**;
- внедрение BeEF.

Этот инструмент стоило упомянуть, но он заслуживает отдельного материала, поэтому в данной статье не будем расписывать все, на что ChEF способен.

выводы

Конечно, можно было бы описать еще часть функциональности BeEF'a, рассказать про phonegap, инфицирование мобильных приложений, применение при MiTM'е и т.д., но все это, к сожалению, не умещается в формат одной статьи. Кратко говоря, BeEF — это действительно самый удобный и самый крутой (во всех смыслах) фреймворк для работы с XSS, который выжимает из простого внедрения JS-кода абсолютный максимум, оставляя место для творчества (BeEF — проект опенсорсный), и к тому же поддерживает кастомные плагины. Такой инструмент прекрасно подходит как и для проведения пентестов, так и для решения различных CTF-тасков и всего, на что еще способна твоя фантазия.