Agent.btz - A Threat That Hit Pentagon

blog.threatexpert.com/2008/11/agentbtz-threat-that-hit-pentagon.html

According to this publication, the senior military leaders reported the malware breach incident that affected the U.S. Central Command network, including computers both in the headquarters and in the combat zones.

The threat involved into this incident is referred as Agent.btz. This is a classification from F-Secure. Other vendors name this threat mostly as Autorun. Some of the aliases assigned to this threat might seem confusing. There is even a clash with another threat that is also detected as Agent.btz by another vendor – but that's a totally different threat with different functionality. This post is about F-Secure-classified Agent.btz – the one that was involved into the aforementioned incident.

At the time of this writing, ThreatExpert system has received and processed several different samples of this threat – further referred as Agent.btz. All these builds exhibit common functionality.

Agent.btz is a DLL file. When loaded, its exported function DllEntryPoint() will be called automatically. Another exported function of this DLL, InstallM(),is called during the initial infection stage, via a command-line parameter for the system file rundll32.exe.

Infection Vector

The infection normally occurs via a removable disk such as thumb drive (USB stick) or any other external hard drive. Once a removable disk is connected to a computer infected with Agent.btz, the active malware will detect a newly recognized drive. It will drop its copy on it and it will create autorun.inf file with an instruction to run that file. When a clean computer recognizes a newly connected removable drive, it will (by default) detect autorun.inf file on it, it will then open it and follow its instruction to load the malware.

Another infection vector: when a clean computer attempts to map a drive letter to a shared network resource that has Agent.atz on it and the corresponding autorun.inf file, it will (by default) open autorun.inf file and follow its instruction to load the malware. Once infected, it will do the same with other removable drives connected to it or other computers in the network that attempt to map a drive letter to its shared drive infected with Agent.atz – hence, the replication.

The autorun.inf file it creates contains the following command to run rundll32.exe:

rundll32.exe .\\[random_name].dll,InstallM

Functionality

When Agent.btz DLL is loaded, it will decrypt some of the strings inside its body. Agent.btz file is not packed. The strings it decrypts are mostly filenames, API names, registry entries, etc.

After decrypting its strings, Agent.btz dynamically retrieves function pointers to the following kernel32.dll APIs: WriteProcessMemory(), VirtualAllocEx(), VirtualProtectEx(). It will need these APIs later to inject malicious code into Internet Explorer process.

Agent.btz spawns several threads and registers window class "zQWwe2esf34356d".

The first thread will try to query several parameters from the values under the registry key:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\StrtdCfg

Some of these parameters contain such details as time out periods, flags, or the name of the domain from which the additional components can be downloaded.

The first thread will spawn 2 additional threads. One of them will wait for 5 minutes, and then it will attempt to download an encrypted binary from the domain specified in the parameters.

For example, it may attempt to download the binaries from these locations:

http://biznews.podzone.org/update/img0008/[random digits].jpg

or

http://worldnews.ath.cx/update/img0008/[random digits].jpg

The downloaded binary will be saved under the file name $1F.dll into the temporary directory.

Once the binary is saved, Agent.btz signals its threads with "wowmgr_is_loaded" event, saves new parameters into the registry values under the key "StrtdCfg", loads Internet Explorer process, decrypts the contents of the downloaded binary, injects it into the address space of Internet Explorer and then spawn a remote thread in it.

At the time of this writing the contents of the binary is unknown as the links above are down. Thus, it's not known what kind of code could have been injected into the browser process. The only assumption can be made here is that the remote thread was spawned inside Internet Explorer process in order to bypass firewalls in its attempt to communicate with the remote server.

Installation

Agent.btz drops its copy into %system% directory by using a random name constructed from the parts of the names of the DLL files located in the %system% directory.

It registers itself as an in-process server to have its DLL loaded with the system process explorer.exe. The CLSID for the in-process server is also random - it is produced by UuidCreate() API.

This threat may also store some of its parameters by saving them into the values nParam, rParam or id under the system registry key below:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\CrashImage

On top of that, Agent.btz carries some of its parameters in its own body – stored as an encrypted resource named CONFIG. Agent.btz locates this resource by looking for a marker 0xAA45F6F9 in its memory map.

File wmcache.nld

The second spawned thread will wait for 10 seconds. Then, it'll save its parameters and some system information it obtains in an XML file %system%\wmcache.nld.

The contents of this file is encoded by XOR-ing it with the following mask:

1dM3uu4j7Fw4sjnbcwlDqet4F7JyuUi4m5lmnxl1pzxl6as80cbLnmz54cs5Ldn4ri3do5L6gs923HL34x2f5cvd0fk6c1a0s

Below is the decoded fragment of the XML file, provided as example:

```
<?xml version="1.0" encoding="unicode"?>
<Cfg>
<Ch>
<add key="Id" value="3024688254" />
```

```
<add key="PVer" value="Ch 1.5" />
<add key="Folder" value="img0008" />
<add key="Time" value="29:11:2008 18:44:46" />
<add key="Bias" value="4294967285" />
<add key="PcName" value="%ComputerName%" />
<add key="UserName" value="%UserName%" />
<add key="WinDir" value="%windir%" />
<add key="TempDir" value="%temp%" />
<add key="WorkDir" value="%system32%" />
<add key="Cndr" value="0" />
<add key="List" value="">
<add key=" 0" value="2" />
</add>
<add key="NList" value="">
</add>
</Ch>
...
</Cfg>
```

Besides the basic system information above, Agent.btz contains the code that calls GetAdaptersInfo() and GetPerAdapterInfo() APIs in order to query network adapter's IP and MAC address, IP addresses of the network adapter's default gateway, primary/secondary WINS, DHCP and DNS servers. The collected network details are also saved into the log file.

File winview.ocx

The second spawned thread will log threat activity into the file %system32%\winview.ocx.

This file is also encrypted with the same XOR mask. Here is the decrypted example contents of that file:

```
18:44:44 29.11.2008 Log begin:
18:44:44 Installing to C:\WINDOWS\system32\[random_name].dll
18:44:44 Copying c:\windows\system32\[threat_file_name].dll to C:\WINDOWS\system32\[random_name].dll (0)
18:44:44 ID: {7761F912-4D09-4F09-B7AF-95F4173120A6}
18:44:44 Creating Software\Classes\CLSID\{7761F912-4D09-4F09-B7AF-95F4173120A6}
18:44:44 Creating Software\Classes\CLSID\{7761F912-4D09-4F09-B7AF-95F4173120A6}\InprocServer32\
18:44:44 Set Value C:\WINDOWS\system32\[random_name].dll
18:44:44 Creating SOFTWARE\Microsoft\Windows\CurrentVersion\ShellServiceObjectDelayLoad\
18:44:44 Native Id: 00CD1A40
18:44:44 Log end.
```

The thread will be saving its parameters and system information into the aforementioned encrypted XML file in the loop – once in every 24 hours.

File mswmpdat.tlb

The original thread will then attempt to start 2 processes: tapi32d.exe and typecli.exe – these attempts are logged. Whenever Agent.btz detects a newly connected removable disk, it will also log the device details into the same log file %system%\mswmpdat.tlb.

The contents of this log file is encrypted the same way – here is the decrypted fragment of it:

18:44:45 29.11.2008 Log begin:
18:44:45 Creating ps C:\WINDOWS\system32\tapi32d.exe (2)
18:44:45 Creating ps C:\WINDOWS\system32\typecli.exe (2)
18:44:45 Log end.
19:02:48 29.11.2008 Log begin:
19:02:49 Media arrived: "D:" Label:"" FS:FAT SN:00000000
19:02:49 Log end.

It is not clear what these 2 files are: tapi32d.exe and typecli.exe - the analyzed code does not create them. It is possible however that the missing link is in the unknown code it injects into Internet Explorer which can potentially download those files.

Files thumb.db

When Agent.btz detects a new drive of the type DRIVE_REMOVABLE (a disk that can be removed from the drive), it attempts to create a copy of the file %system%\1055cf76.tmp in the root directory of that drive as thumb.db.

In opposite, if the newly connected drive already contains file thumb.db, Agent.btz will create a copy of that file in the %system% directory under the same name. It will then run %system%\thumb.db as if it was an executable file and then delete the original thumb.db from the connected drive.

The analyzed code does not create 1055cf76.tmp, but if it was an executable file downloaded by the code injected into Internet Explorer (as explained above), then it would have been passed into other computers under the name thumb.db. Note: an attempt to run a valid thumb.db file, which is an OLE-type container has no effect.

Files thumb.dd and mssysmgr.ocx

Agent.btz is capable to create a binary file thumb.dd on a newly connected drive. The contents of this file starts from the marker 0xAAFF1290 and is followed with the individual CAB archives of the files winview.ocx (installation log), mswmpdat.tlb (activity log), and wmcache.nld (XML file with system information).

When Agent.btz detects a new drive with the file thumb.dd on it (system info and logs collected from another computer), it will copy that file as %system%\mssysmgr.ocx.

This way, the locally created files do not only contain system and network information collected from the local host, but from other compromised host (or hosts) as well.