New Variants of Agent.BTZ/ComRAT Found: The Threat That Hit The Pentagon In 2008 Still Evolving; Part 2/2

intezer.com/new-variants-of-agent-btz-comrat-found-part-2/

September 13, 2017

Written by Omri Ben Bassat - 13 September 2017

Our previous blog post was a short brief of new Agent.BTZ variants that we found. This second part in the series will demonstrate in greater detail exactly *how* we discovered these new variants.

1. Methodology

To begin, we used our hunting methodology, which consist of four main parts:

- **Collection:** Collect multiple samples from different versions.
- **Analysis:** Mark functions that have stayed consistent across all versions that are likely to be a part of the next version.
- **Creating a signature:** Create a robust yet flexible YARA rule for these functions.
- **Hunting:** Search a large repository of files with that YARA rule (VirusTotal, for example).

2. Why focus on Agent.BTZ?

We chose to focus on Agent.BTZ for several reasons: **First, This is one of the oldest state-sponsored threat, developed and operated by the Turla group since (at least) 2007 for dozens of targeted attacks.**

Second, there is also a lot of public knowledge regarding Agent.BTZ specifically and Turla group in general available online, including intel reports, technical analyses and malware samples, which we used for our research.

Third is the fact that this specific malware has remained out of public view for the last two to three years; **however, we recognized that it wasn't likely to disappear–it has just continued to fly under the radar.**

3. Do the math

We based our research on an earlier publication from three years ago: "Evolution of sophisticated spyware: from Agent.BTZ to ComRAT" by Paul Rascagnères of GDATA (at the time). In this excellent blog post, Paul described the evolution of Agent.BTZ to ComRAT between 2007-2014 by manually diffing (BinDiff) two different candidates from each major internal hard-coded version (referred by the authors as "Ch" or "PVer").

The following table shows the code's similarity between each version to its direct neighbors. By summing up the data in this table we can conclude that, in general, about ~30% of the original code has been used in every version up to the latest known version as of 2014 (marked in red).

Version	1.0	1.5	2.03	2.11	2.14.1	3.00	3.10	3.20	3.25	3.26
1.0	100%	90%	75%	72%	70%	**42%**	38%	35%	32%	30%
1.5	90%	100%	83%							
2.03	75%	83%	100%	96%						
2.11	72%		96%	100%	98%					
2.14.1	70%			98%	100%	60%				
3.00	**42%**				60%	100%	90%			
3.10	38%					90%	100%	93%		
3.20	35%						93%	100%	91%	
3.25	32%							91%	100%	95%
3.26	30%								95%	100%

2007

2014

*** https://www.gdatasoftware.com/blog/2015/01/23927-evolution-of-sophisticated-spyware-from-agent-btz-to-comrat*

Obviously this 30% isn't comprised of totally unique code made by Agent.BTZ's developers; rather, it is a mixture made of mostly common code that can be found in many more software products (both legit and malware). For example, these could include C Runtime Library or any other 3rd party library such as zlib. **Using Intezer's technology, we were able to do a 'deep dive' into this code, automatically mapping all of the common, library and (most importantly) unique pieces of code created by the malware's authors**.

4. Mysterious magic number

After filtering out all of the common and library code, we were left with several unique functions that can be found in every version of the malware since 2007. The most prominent is the following function:

In the picture you can see the two, almost identical functions from the first version (Ch 1.0) on the right and the latest version (Ch 3.26) of Agent.BTZ on the left. This function is initially reading first four bytes of a given file, and then comparing them for the magic number 0xAAFF1290 (marked in red). If it matches, the function will return true; otherwise, it is false.

So, what is this magic number? Which file is it? We were actually unable to find the function that creates this file within the same binary (ver 3.26). The function shown above is the only reference to that magic number. Luckily, while re-reading old ThreatExpert's Agent.BTZ analysis from 2008 we happened to notice the following paragraph:

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).

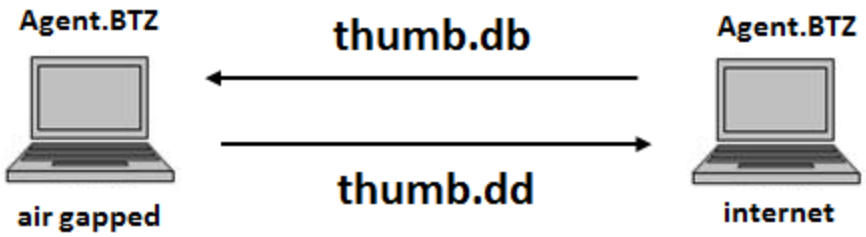
When Agent.btz detects a new drive with the file thumb.dd on it (system info and logs collected from another computer), it will copy that file as %system%\mssysmgr.ocx.

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.

POSTED BY SERGEI SHEVCHENKO AT 5:30 AM
LABELS: AGENT.BTZ

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

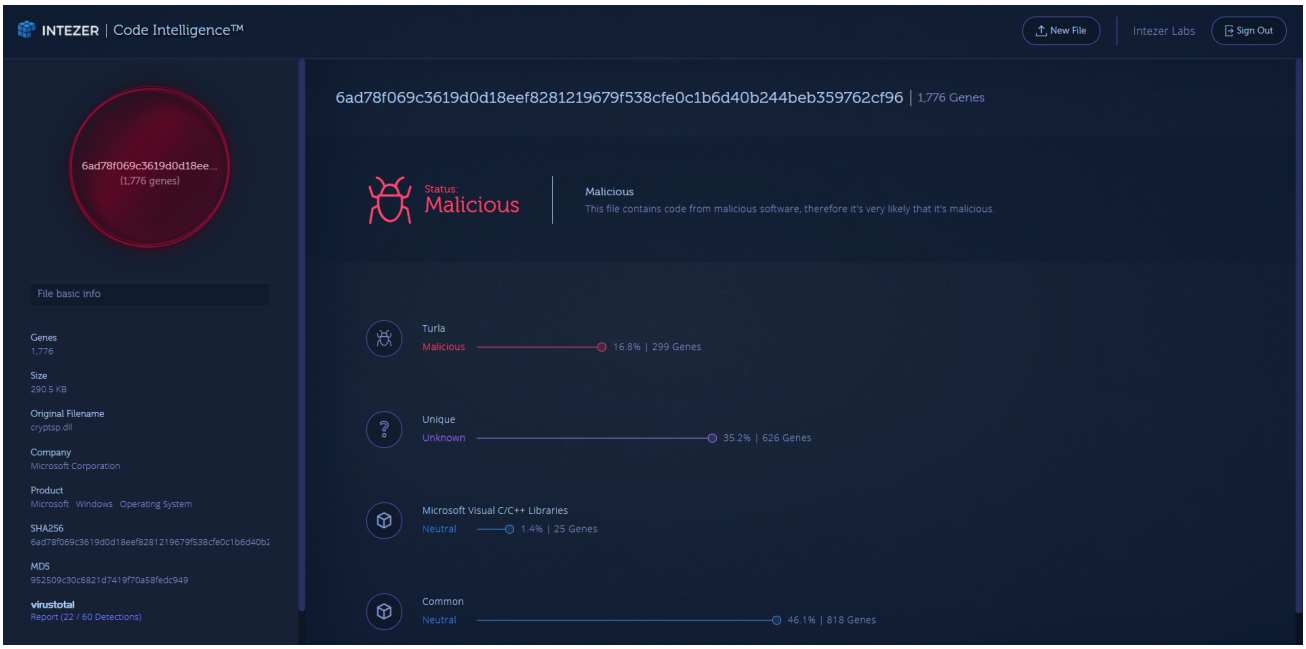
So now we know that this function is used to verify thumb.dd files, which are log/config files leaked over newly-connected USB drives (to overcome air-gapped networks, like those of the Pentagon). But wait… why didn't we found the function which creates these files? **Because the USB-infection vector was removed few years earlier! And yet the adversary is still looking for these valuable thumb.dd files…**



5. Hunting for new variants

So far, we know that this function exists, and it has stayed consistent across all versions of the malware; we know what it does (verify certain magic number) and why (detect thumb.dd files leaked from internal network by older versions of the malware). By that information, we can tell that it's likely to be used in future versions as well.

The next step involves writing a dedicated YARA rule(see appendix) for that specific function and searching for new samples. The rule has to be tolerant to minor changes between versions (mainly due to different compilation flags). Using the VirusTotal Intelligence service, we were able to scan about 2-3 months' worth of file uploads. As soon as the scan finished, we dug into the results and **discovered this first new variant of Agent.BTZ that wasn't yet mentioned in any public report:**



**A screenshot from the Intezer Analyze™ product displaying partial code connections between new sample to old samples of Agent.BTZ(Turla group).

This specific file was supposedly compiled at 2016-07-13 (the timestamp was modified in some of the earlier samples) and uploaded to VT 2017-05-11, which means that this sample is at least *two years newer* than any previous sample.

6. **Main differences between old(3.26) & new samples**

- Filename
  New file names
  - activeds.dll – proxy dll
  - stdole2.tlb – main payload
- Exports
  - AddAtomS removed
    Legacy stub function
  - AddAtomT removed
    Legacy installation function
  - UnInstallW added
    Force delete file. Used by new dropper for self delete.
- PVer(internal version tracking)
  random version id(see following picture) instead of incremental,

might be due to GDATA's 2014 publications.

- For example:
  - 0.8.1529506101
  - 0.9.1528434231
  - …
- Persistence(COM Hijacking)
  New CLSID
  {DFFACDC5-679F-4156-8947-C5C76BC0B67F}

C2 Infrastructure

Both using "Satellite Turla" infrastructure

Config & Log Files

Same 512 bytes encryption key as 3.26

Different file paths:

config: %appdata%\Microsoft\Windows\PrivacIE\High\desktop.ini

log: %appdata%\Microsoft\Windows\PrivacIE\High\index.dat

7. **Indicators of Compromise**

ype	indicator
sha256	69690f609140db503463daf6a3699f1bf3e2a5a6049cefe7e6437f762040e548
sha256	6798b3278ae926b0145ee342ee9840d0b2e6ba11ff995c2bc84d3c6eb3e55ff4
sha256	73db4295c5b29958c5d93c20be9482c1efffc89fc4e5c8ba59ac9425a4657a88
sha256	50067ebcc2d2069b3613a20b81f9d61f2cd5be9c85533c4ea34edbefaeb8a15f
sha256	380b0353ba8cd33da8c5e5b95e3e032e83193019e73c71875b58ec1ed389bdac
sha256	9c163c3f2bd5c5181147c6f4cf2571160197de98f496d16b38c7dc46b5dc1426
sha256	628d316a983383ed716e3f827720915683a8876b54677878a7d2db376d117a24
sha256	f27e9bba6a2635731845b4334b807c0e4f57d3b790cecdc77d8fef50629f51a2
sha256	a093fa22d7bc4ee99049a29b66a13d4bf4d1899ed4c7a8423fbb8c54f4230f3c
sha256	6ad78f069c3619d0d18eef8281219679f538cfe0c1b6d40b244beb359762cf96
sha256	49c5c798689d4a54e5b7099b647b0596fb96b996a437bb8241b5dd76e974c24e

sha256	e88970fa4892150441c1616028982fe63c875f149cd490c3c910a1c091d3ad49
sha256	89db8a69ff030600f26d5c875785d20f15d45331d007733be9a2422261d16cea
ip	81.199.34[.]150
dns	elephant.zzux[.]com
dns	angrybear.ignorelist[.]com
dns	bigalert.mefound[.]com
dns	bughouse.yourtrap[.]com
dns	getfreetools.strangled[.]net
dns	news100top.diskstation[.]org
dns	pro100sport.mein-vigor[.]de
dns	redneck.yourtrap[.]com
dns	savage.2waky[.]com
dns	tehnologtrade.4irc[.]com
ip	81.199.160[.]11
dns	forums.chatnook[.]com
dns	goodengine.darktech[.]org
dns	locker.strangled[.]net
dns	simple-house.zzux[.]com
dns	specialcar.mooo[.]com
dns	sunseed.strangled[.]net
dns	whitelibrary.4irc[.]com
dns	bloodpearl.strangled[.]net
dns	getlucky.ignorelist[.]com
dns	proriot.zzux[.]com
dns	fourapi.mooo[.]com
dns	nopasaran.strangled[.]net

ip	78.138.25[.]29
dns	showme.twilightparadox[.]com
dns	mouses.strangled[.]net
ip	82.146.175[.]69
dns	mouses.strangled[.]net
ip	178.219.68[.]242
dns	ftp.fueldust.compress[.]to
dns	ftp.linear.wikaba[.]com
dns	ftp.mysterysoft.epac[.]to
dns	ftp.scroller.longmusic[.]com
dns	ftp.spartano.mefound[.]com
dns	fueldust.compress[.]to
dns	linear.wikaba[.]com
dns	mysterysoft.epac.to
dns	safety.deaftone[.]com
dns	salary.flnet[.]org
dns	scroller.longmusic[.]com
dns	spartano.mefound[.]com
ip	88.83.25[.]122
dns	robot.wikaba[.]com
ip	41.223.91[.]217
dns	smileman.compress[.]to
dns	decent.ignorelist[.]com
dns	dekka.biz[.]tm
dns	disol.strangled[.]net
dns	eraser.2waky[.]com

dns	filelord.epac[.]to
dns	justsoft.epac[.]to
dns	smuggler.zzux[.]com
dns	sport-journal.twilightparadox[.]com
dns	sportinfo.yourtrap[.]com
dns	stager.ignorelist[.]com
dns	tankos.wikaba[.]com
dns	grandfathers.mooo[.]com
dns	homeric.mooo[.]com
dns	jamming.mooo[.]com
dns	pneumo.mooo[.]com
dns	razory.mooo[.]com
dns	anger.scieron[.]com
dns	gantama.mefound[.]com
dns	letgetbad.epac[.]to
dns	rowstate.epac[.]to
dns	memento.info[.]tm
ip	196.43.240[.]177
dns	bughouse.yourtrap[.]com
dns	news100top.diskstation[.]org
ip	169.255.102[.]240
dns	harm17.zzux[.]com
dns	mountain8.wikaba[.]com
sha256	0e0045d2c4bfff4345d460957a543e2e7f1638de745644f6bf58555c1d287286
sha256	bdcc7e900f10986cdb6dc7762de35b4f07f2ee153a341bef843b866e999d73a3
sha256	fac13f08afe2745fc441ada37120cebce0e0aa16d03a03e9cda3ec9384dd40f2

sha256	bae62f7f96c4cc300ec685f42eb451388cf50a13aa624b3f2a019d071fddaeb1

Related articles:

1. https://www.gdatasoftware.com/blog/2014/11/23937-the-uroburos-case-new-sophisticated-rat-identified
2. https://www.gdatasoftware.com/blog/2015/01/23927-evolution-of-sophisticated-spyware-from-agent-btz-to-comrat
3. http://blog.threatexpert.com/2008/11/agentbtz-threat-that-hit-pentagon.html
4. https://securelist.com/satellite-turla-apt-command-and-control-in-the-sky/72081/
5. https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/waterbug-attack-group.pdf
6. https://securelist.com/the-epic-turla-operation/65545/
7. http://artemonsecurity.com/snake_whitepaper.pdf
8. https://www.gdatasoftware.com/blog/2015/01/23926-analysis-of-project-cobra

Omri Ben Bassat
Ex-officer in the IDF CERT. Malware analyst and a reverse engineer with vast experience in dealing with nation-state sponsored cyber attacks. Omri is the creator of Master of Puppets (MoP)—an open-source framework for reverse engineers who wish to create and operate trackers for new malware found in the wild—which was presented during the Black Hat USA 2019 Arsenal.