

Forum:[Hack Forums](#)

Thread: [AV Evasion](#)

CyberStrike - March 6, 2018, 6:02 am

Disclaimers: I am not an expert on either AV solutions for both end-user as well as enterprise solutions, but have picked up a little bit.

How AV works in general:

Software has is categorized in one of the following lists :

BlackListing - blocked

WhiteListing - allowed to run

Malware is generally detected using one of the following:

Signature Based - Signature based detections uses specific header/strings/file size or patterns (packer used/byte sequence ..etc) within a file to check if it is good or bad. These signatures are often found from malware samples. Of course there can be false positives here.

Disadvantage: If you test enough you can find the signature and slightly change it, also compiling with a different compiler can change the signature.

Behavior Based - here the AV will try to determine the behavior of the software to see if it matches known techniques used by security tools or malware. This is where often we may see a "trojan.gen" it is matching a heuristic pattern found in other malware.

Disadvantage: The software has to be analyzed within a sandbox and often encrypted/compressed or self modifying code is not analyzed during this. (Due to this you often have encrypted payloads being flagged)

Something else to note is that your AV requires a lot of resources and cannot analyze a program indefinitely, so if malicious actions are carried out afterwards. So also while sandboxes try to be hidden from malware often malware can scan for them and sleep until after the evaluation is complete.

(Note is that generally AV products do not do anything about exploitation, but McAfee Enterprise solution does)

Not All AV's are the same:

Often we will find on VT (VirusTotal) the engines are more geared for heuristics and have beta signatures so it's not quite like your home AV solution. Also false positives are more common.

In Memory Detections:

Below are common detection techniques pretty much consider them as "Tells" as well as tools to detect such odd behavior

- Memory Detection

- ? Process
 - ? Parent Process
 - ? Process Name
 - ? Process Signer
 - ? Valid Signature - code signing some MS products are not scanned since false positives will break an entire network
 - ? Command Line Args
 - ? Current working directory- Temp files are often flagged
- ? Thread
 - ? Thread ID
 - ? Start Address
 - ? Start Address Module
 - ? Priority
- ? Memory Properties
 - ? Base Address
 - ? Type of Memory
 - ? State of Memory
 - ? Size
 - ? File
 - ? Initial Permissions (not in process hacker)
 - ? Permissions

Tools for Detections:

Process Hacker

LOKI

YARA (used by some AV's)

Get-InjectedThread

Something to note here, "odd" things should not translate to "Intent" of an application being malicious. AV solutions generally give a free pass to system critical as well as other security tools, because if a critical system file is flagged and it is a false positive that's going to be a lot of angry customers. We have our tools and we have an idea of what we are looking for but what exactly?

- Detection

- ? Thread Start Address - No module associated with the start address. Look for DLL that is not mapped in the current process mapped files. Programs call functions which are run under the program or a function in the DLL as a memory mapped DLL.

- ? Memory Permissions - RWX permissions common pattern or AllocateProtect and Protect Pairs, for example starting in RW then adding the X flag is a common indicator in offensive tools.

- ? Memory Content - Headers of a PE not associated with files on disk or the strings with that file. EG MZ This program cannot be run in DOS mode.

- Memory Detection Applications

- ? Point IN TIme Analysis - non persistent software deployed once. runs through current processes and threads, investigates content and properites reports signs of malicious activity allows human to make determination.. some false positives are OK

- ? RealTIme Detection/prevention - Less false positives are OK it can break the network or machine. Persistent software; Even Driven Investigates content and properties correlating multiple signs. From there is can do nothing or send an alert or terminate the action. Some AV and other solutions use these techniques

Now one should note that memory detection applications especially the ones preventing if they

have a false positive remember those angry customers well they will be back. So now that we have a rough idea as to what exactly that we are dealing with how do we avoid it?

First let me ask when you start up chrome or firefox are you downloading a portable version and running it each time from a temporary directory? No that'd be crazy so stageless files on disk for the initial payload would help it would better blend in and appear more natural. As we know nowadays IR (Incident Response) always should and will dump the memory, so consider encrypting that memory. Remember that each payload has it's own heuristics :

Evasion:

Before going to in memory first let us look at common methods for AV evasion :

- Polymorphism/Metamorphism - new signature on each run
- Encryption
- Sanbox Detection
- Process Injection
- Applocker Bypasses

Shellcoder loaders, are perhaps not commonly used but you can use python/c/powershell and more to load shellcode into memory during execution of your program or script. Using ShellForge we can quickly take C source code, perhaps our dropper with a sleep function have it in assembly and have that shell code injected as part of a larger staged payload. The dropped file maybe a powershell script which will use an applocker bypass and inject our payload into an existing process.

Detection is done by detecting anomalies and are not a indicator of intent: So not doing the things that we are going to be detected is important

- Thread Start Address
 - ? EXE/DLL OK
 - ? Process Hollowing: stick to architecture
 - ? Avoid Hijacking Existing Remote Threads
 - Do not start a new thread with an abitrary pointer.
 - ? Instead hijack existing threads use SetThreadContext/LoadLibrary/CreateRemoteThread.. For LoadLibrary you will have to drop to disk.
 - Memory Permissions:
 - ? Avoid RWX instead map a DLL and over write the memory
 - Memory Content:
 - ? Don't look like a DLL unless expected
 - ? Obfuscate or remove strings that can be used in IR
 - ? avoid stagers
 - ? Harder to notice the attacker by the content in the memory
- Append NOP to start of payload

- Process Injection
 - ? Allocate memory in remote process
 - ? copy program to remote process
 - ? Create thread to run program
- Process Hollowing
 - ? Allocate memory in remote process
 - ? Copy program to remote process
 - ? Hijack existing thread to run program

A word on OPSEC:

After your payload is somewhat not quite ready to head out to the target say before you have used your code signing certificate we have to see what can be found if that file is captured.

Against your payload run (consider saving it as a bin here)

```
strings -e {s,S,b,l,B,L} payload.exe
strings payload.exe
hexdump -C payload.exe
```

In Metasploit using post modules such as "payload_inject" "reflective_dll_inject" and executing a new payload.

ProcessHollowing is fine for same architecture.

(References and most will be posted @ 8000 character limit)

Let's discuss a bit the future of AV, in recent tests with using signature based detection researchers studying worm variants detected 323 features and were able to identify 5 more variants that were previously unknown. Using only 20 of the features they had a success rate of greater than 90%. AntiVirus researchers have also started to analyze OPCODE, assembly language instructions which interact with the CPU, as of right now researchers dealing with obfuscated malware are just over 85% successful, using this technique for detection.

(The Effects of Traditional Anti-Virus Labels on Malware Detection Using Dynamic Runtime Opcodes)

References:

https://en.wikipedia.org/wiki/Antivirus_software
https://en.wikipedia.org/wiki/Static_program_analysis
https://en.wikipedia.org/wiki/Dynamic_program_analysis
https://en.wikipedia.org/wiki/Heuristic_analysis
<https://www.symantec.com/connect/articles/windows-anti-debug-reference>

The Antivirus Hacker's Handbook
AntiVirus Firewall Evasion Techniques Evolution Of Download Deploy Shellcode
WhitePaper: Bypassing Antivirus with a Sharp Syringe
Outsmarted: Why Malware Works in the face of Antivirus Software
The Effects of Traditional Anti-Virus Labels on Malware Detection Using Dynamic Runtime Opcodes

YAK_CODER - March 6, 2018, 9:25 am

Perfect explanation
thanks man

CyberStrike - March 6, 2018, 2:07 pm

(Mar 6, 2018 10:25 AM)crypt4u Wrote: [\[CITE\]](#) Perfect explanation
thanks man

Thank you I know it is a long read, nearly forgot payload heuristics

Payload Heuristics :

Quote:

Payload	Migrates	RWX	Module-less Thread
EXE	X	X	X
DLL	X	X	X
DLL(x64->x86)		X	
PowerShell	X		
Python	X		
Service-EXE	X		
Service-EXE (x64->x86)	X		X
VBA	X	X	X
Java	X		
Java(x64)	X		X

splutter - March 7, 2018, 12:06 am

It's a nice addition to the community but perhaps this is more convenient and in-depth in terms of explanation: <https://blog.cobaltstrike.com/2018/02/08/in-memory-evasion/>

CyberStrike - March 7, 2018, 4:05 am

Yes this is a a very good resource I missed that one in my original post and when I try to go back and edit it gives me an error that the post is far too long, I tried to stay away from just focusing on cobalt strike payloads and tried to keep it more general so with a bit of work users can apply these methods of evasion to their toolset. I did not get into stageless or staged payloads and how they work but the Cobalt Strike video series on this subject covers this as well as other mainly Cobalt Strike techniques. Also "Taking Back NetCat" really is what started me on this path when I created the following thread <https://hackforums.net/showthread.php?tid=5767518> .

emmabee - March 8, 2018, 9:16 am

Great...many thanks learning all the time

CyberStrike - March 8, 2018, 11:49 pm

(Mar 8, 2018 10:16 AM)emmabee Wrote: [\[CITE\]](#) Great...many thanks learning all the time
Awesome keep it up knowledge is very important never stop learning and don't be scared to get out of your comfort zone.

denedene123 - March 9, 2018, 4:28 pm

Dammn boi, thanks for the education.

CyberStrike - March 10, 2018, 1:59 am

You're Welcome, I hope that people test their payloads, doing things manually really is possible while it may take a little bit more than just running a tool it works rather well. I tested multiple payloads with process hacker with python/cs/powershell/cs/ shellcode loaders custom and including common solutions.

I also did some work with carving various injected processes while loaded in RAM to recover the file. Perhaps we need an anti-forensics discussion at some point.

LordDragon - August 9, 2018, 5:40 pm

HQ post,bookmarking this for future reference.

991z3r0 - August 9, 2018, 5:49 pm

Posting to come back to, can be useful for dev

T0pi4ry - August 10, 2018, 10:20 am

HQ Thread, from HQ Member .

This article need to be PINNED .

Thnaks a lott for your big effort ;) :)

CyberStrike - August 11, 2018, 6:04 am

Thank you for all of the positive comments I appreciate it. If you want check out the references I made sure to include them so one can read and learn a bit more.

sara1005 - August 20, 2018, 7:50 pm

Great post and much appreciated. So much to learn . . . it never ends.

Last_viking - August 20, 2018, 7:50 pm

HQ tutorial mate keeping contributing to community.

Well written.

CyberStrike - August 20, 2018, 8:20 pm

(Aug 20, 2018 07:50 PM)sara1005 Wrote: [\[CITE\]](#) Great post and much appreciated. So much to learn . . . it never ends.

It literally is never ending I spend at least 2 maybe 4 hours per day just "keeping up" tactics and techniques are always evolving much like security.

(Aug 20, 2018 07:50 PM)last viking Wrote: [\[CITE\]](#) HQ tutorial mate keeping contributing to community.

Well written.

Thank you, I just hope that people read the sources which I used as this is just a very brief summary, when you look to how many pages the sources are I really don't think I could do them justice.

HiddenEyE - October 10, 2018, 3:39 am

very HQ post. But OP have you coded a crypter,rat or botnet to check this out?

CyberStrike - October 10, 2018, 2:54 pm

(Oct 10, 2018 03:39 AM)HiddenEyE Wrote: [\[CITE\]](#) very HQ post. But OP have you coded a crypter,rat or botnet to check this out?

A simple reverse shell from scratch in cpp and golang (not so simple anymore my payloads depend on various things enumerating values for say the domain/date/user/ maybe software on the target have to kind of expect things there based upon what you have learned about a target, for payloads to decrypt and run. And I have been doing these things at work against corporate networks for a bit now, like code signing.

This said cpp allows you to specify the parent when using createprocess by using InitializeProcThreadAttributeList and _STARTUPINFOEXA, of course you own it, it's your process or have debug rights. Child processes inherit handles from the parent.

<https://docs.microsoft.com/en-us/windows/desktop/api/processthreadsapi/nf-processthreadsapi-initializeprocthreadattributelist>

https://docs.microsoft.com/en-us/windows/desktop/api/winbase/ns-winbase-_startupinfoexa

Edit: Let me just add the following I work from source code first I don't just autogenerate executables (yes the many forms which can execute code not just an exe or dll).

xbutterflyeffect - *October 11, 2018, 1:27 pm*

HQ post. Thanks for the tip.