Reverse RDP Attack: The Hyper-V Connection

Research by: Eyal Itkin

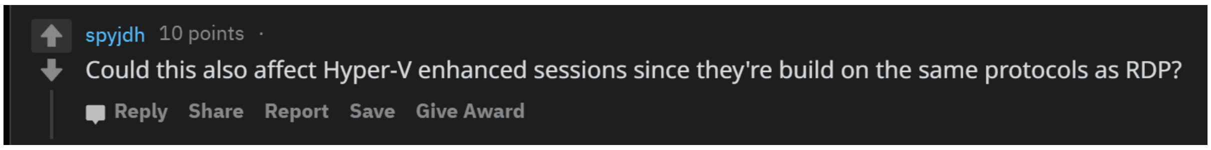
Overview

Earlier this year, we published our research on the Reverse RDP Attack. In our previous blog post, we described how we found numerous critical vulnerabilities in popular Remote Desktop Protocol (RDP) clients. However, our research didn't just end there. In this blog post, we discuss new developments, including a new target: Microsoft's Hyper-V.

TL;DR – The Path-Traversal vulnerability we found in Microsoft's RDP client is also applicable as a guest-to-host VM escape in Hyper-V Manager. Following this revelation, Microsoft issued a CVE (CVE-2019-0887), and released a patch for the vulnerability.

Hyper-V? It isn't an RDP client

Following our publication, we received numerous comments asking if the vulnerabilities in Microsoft's RDP client can affect Microsoft's Hyper-V product. Here is one such question, from the Reddit discussion:


spyjdh 10 points ·

Could this also affect Hyper-V enhanced sessions since they're build on the same protocols as RDP?

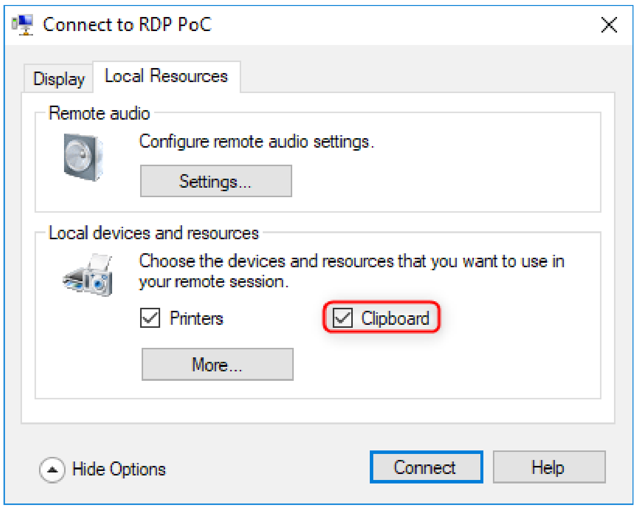
Figure 1: A Reddit comment asking about the Hyper-V implications of the vulnerabilities.

Some background: Microsoft's Hyper-V is a virtualization technology that is used in Microsoft's Azure cloud, and offered as a virtualization product on top of the Windows 10 operating system. Similar to other virtualization technologies, Hyper-V comes with a graphical user interface that lets a user manage the local / remote virtual machine (VM).

After this short lesson on Hyper-V, the question remains: What does Hyper-V have to do with RDP? The answer is found in the term "enhanced sessions", which are Microsoft's equivalent of VMWare's VM tools. Enhanced sessions offer a user an extended functionality that includes **clipboard synchronization** between the guest and the host. As you may recall, we already found a Path-Traversal vulnerability in the clipboard synchronization implemented by Microsoft's RDP client. It's time to check if the same vulnerability will also work out-of-the-box for the Hyper-V case.

To perform the tests, we installed Hyper-V on our machine, and converted our initial virtual machine into a Hyper-V machine. We then used the Hyper-V

Manager program and connected to our virtual machine. Soon enough, we encountered the familiar Settings window seen in Figure 2.


Connect to RDP PoC

Display | Local Resources

Remote audio

Configure remote audio settings.

Settings...

Local devices and resources

Choose the devices and resources that you want to use in your remote session.

☑ Printers ☑ Clipboard

More...

Hide Options | Connect | Help

Figure 2: Settings window for Hyper-V VM, which is the same window as in mstsc.exe.

As we can see, this is exactly the same GUI window that is also used for the settings of an RDP connection when using mstsc.exe. This looks promising. To top it off, the Clipboard resource is shared by default, meaning this is an enhanced session.

Feeling motivated, we ran the same script we initially submitted to Microsoft in the mstsc.exe case, and it **worked**. We just found a **Hyper-V guest-to-host VM escape** over the control interface, using the RDP vulnerability! Here is a video of a paste-only attack for the Hyper-V scenario:

RDP & Hyper-V - Explained

It turns out that RDP is used behind the scenes as the control plane for Hyper-V. Instead of re-implementing screen-sharing, remote keyboard and a synchronized clipboard, Microsoft decided that all of these features are already implemented as part of RDP, so why not use it in this case as well? In our joint BlackHat presentation with Dana Baril from Microsoft, we elaborate more on yet another Virtualization-based security isolation technique, and the custom version of RDP that it relies on.

While it was hard for any security researcher to miss Microsoft's effort to test and improve the security of its Hyper-V technology, we can learn an important lesson from this research. As the saying goes: your system is only as strong as its weakest link. In other words, by depending on other software libraries, Hyper-V Manager inherits all of the security vulnerability that are found in RDP, and in any other software library that it uses.

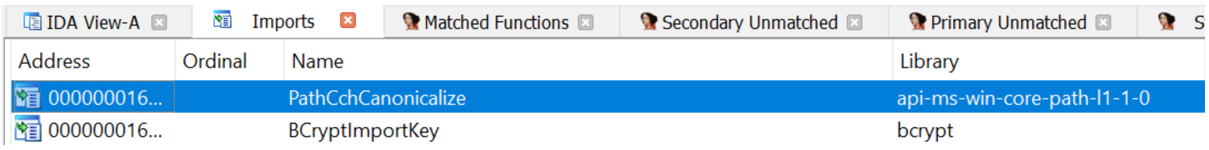
Although the lesson might sound trivial, the problem of updating versions and keeping track of vulnerabilities in external software dependencies is known to be a notoriously hard one.

Microsoft’s Patch

In our previous publication, we quoted Microsoft’s official response. The bottom line of it was that there was no patch for the Path-Traversal vulnerability, and it didn’t even get an official CVE. This time, however, it seems that Microsoft decided to fix the vulnerability. Immediately after we disclosed the Hyper-V implications of the vulnerability to MSRC, we got a reply that they will once again look at it. A few weeks later, a new MSRC ticket was opened for it, later leading to an official CVE, CVE-2019-0887, and a patch.

Before we start analyzing the patch, let’s recap the vulnerability. A malicious RDP server can send a crafted file transfer clipboard content that will cause a Path-Traversal on the client’s machine. Therefore, we expect to see that mstscax.dll will check the received FileGroupDescriptorW clipboard format, and sanitize each file path that is contained inside.

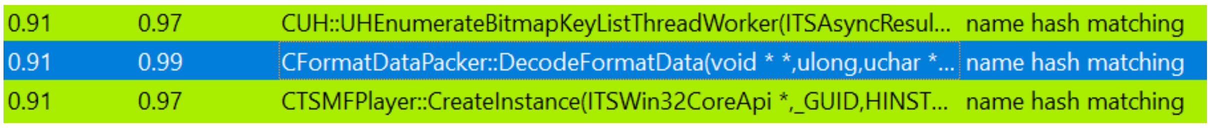
We decided to use BinDiff to analyze Microsoft’s fix, but unfortunately, it found dozens of modified functions. To help clear up the mess, we looked at the import section and saw a new entry that wasn’t there before, as shown in Figure 3.



Address	Ordinal	Name	Library
000000016...		PathCchCanonicalize	api-ms-win-core-path-l1-1-0
000000016...		BCryptImportKey	bcrypt

Figure 3: PathCchCanonicalize() is now used by the patched .dll file.

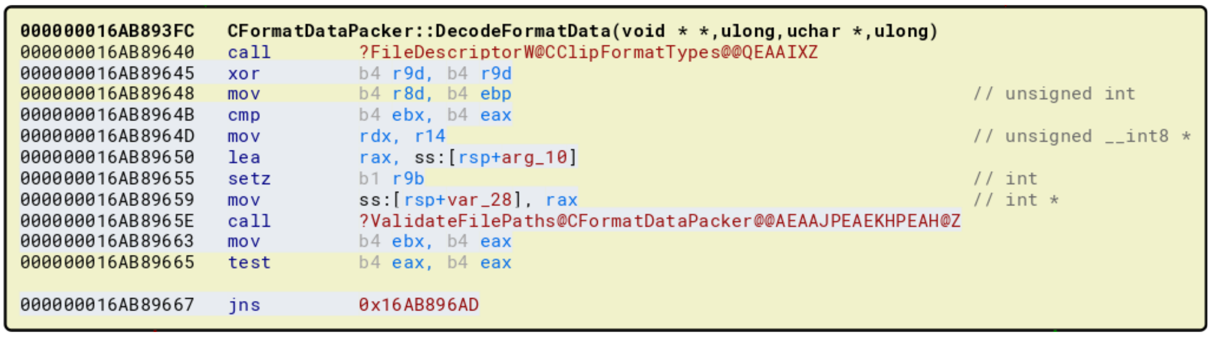
According to MSDN, the function PathCchCanonicalize “Converts a path string into a canonical form.” This is exactly what was missing in the first place. The next question is who invokes this function? Soon enough, we saw the function that introduces the fix:



0.91	0.97	CUH::UHEnumerateBitmapKeyListThreadWorker(ITSAsyncResul...	name hash matching
0.91	0.99	CFormatDataPacker::DecodeFormatData(void * *,ulong,uchar *...	name hash matching
0.91	0.97	CTSMFPlayer::CreateInstance(ITSWin32CoreApi *,_GUID,HINST...	name hash matching

Figure 4: CFormatDataPacket::DecodeFormatData() was changed, as shown in BinDiff.

The change itself can be seen in Figure 5:

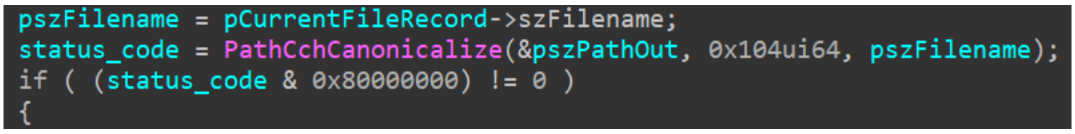


```
000000016AB893FC   CFormatDataPacker::DecodeFormatData(void * *,ulong,uchar *,ulong)
000000016AB89640   call         ?FileDescriptorW@CClipFormatTypes@@QEAAIXZ
000000016AB89645   xor          b4 r9d, b4 r9d
000000016AB89648   mov          b4 r8d, b4 ebp                                // unsigned int
000000016AB8964B   cmp          b4 ebx, b4 eax
000000016AB8964D   mov          rdx, r14                                      // unsigned __int8 *
000000016AB89650   lea          rax, ss:[rsp+arg_10]
000000016AB89655   setz         b1 r9b                                        // int
000000016AB89659   mov          ss:[rsp+var_28], rax                          // int *
000000016AB8965E   call         ?ValidateFilePaths@CFormatDataPacker@@AEAAJPEAEKHPEAH@Z
000000016AB89663   mov          b4 ebx, b4 eax
000000016AB89665   test         b4 eax, b4 eax

000000016AB89667   jns          0x16AB896AD
```

Figure 5: A call to CFormatDataPacker::ValidateFilePaths() was added to the flow.

When handling incoming FileGroupDescriptorW file formats, the client passes the format to a new function that verifies the blob's structure. This new function checks that the data is structured correctly, and then calculates the canonicalized form for each filename:



```
pszFilename = pCurrentFileRecord->szFilename;
status_code = PathCchCanonicalize(&pszPathOut, 0x104ui64, pszFilename);
if ( (status_code & 0x80000000) != 0 )
{
```

Figure 6: Calculate the canonicalized form for every filename.

If successful, the canonicalized output is then compared to the original filename, and any mismatch between the two results in an error. This means that if our filename contains strings of the form "." or "..", it will be changed when converted to the canonicalized form, thus failing the validity check.

It appears that the fix matches our initial expectations, and our Path-Traversal vulnerability is now fixed.

Conclusion

Now that there is a patch for the Path-Traversal vulnerability, we highly recommend all users to install the patch to protect both their RDP connections and their Hyper-V environment.

Our research shows a researcher can never know if a given research project has reached its end. After we "finished" the RDP research (with mixed feelings), the feedback we received from the research community helped us discover the full implications of the vulnerabilities we found, which lay way beyond the initial RDP scope of our research.