Digging deep for PLATINUM

blogs.technet.microsoft.com/mmpc/2016/04/26/digging-deep-for-platinum/

April 26, 2016

This blog introduces our latest report from the Windows Defender Advanced Threat Hunting team. You can read the full report at:

PLATINUM: Targeted attacks in South and Southeast Asia

There is no shortage of headlines about cybercriminals launching large-scale attacks against organizations. For us, the activity groups that pose the most danger are the ones who selectively target organizations and desire to stay undetected, protect their investment, and maximize their ROI. That's what motivated us – the Windows Defender Advanced Threat Hunting team, known as hunters – when we recently discovered a novel technique being used by one such activity group.

We have code named this group PLATINUM, following our internal practice of assigning rogue actors chemical element names. Based on our investigations, we know PLATINUM has been active since 2009 and primarily targets governmental organizations, defense institutes, intelligence agencies, and telecommunication providers in South and Southeast Asia. The group has gone to great lengths to develop covert techniques that allow them to conduct cyber-espionage campaigns for years without being detected.

Uncovering these kinds of techniques is true detective work, and finding them in the wild is a challenge, but with the wealth of anonymized information we can utilize from over 1 billion Windows devices, a broad spectrum of services, Microsoft's intelligent security graph as well as advanced analytics and machine algorithms to surface suspicious behaviors, Microsoft is in the best position to do so.

Digging up the nugget

Through our advanced and persistent hunting, we discovered PLATINUM is using hotpatching as a technique to attempt to cloak a backdoor they use. Using hotpatching in the malicious context has been theorized [1], [2], but has not been observed in the wild before. Finding such techniques is a focus of the Microsoft APT hunter team, and we want to provide some brief insights on how the team dug up this PLATINUM "nugget".

In the first part of this methodology, a hunter carves out some rough data sets from existing information and data that can be further analyzed. This could be based on rough heuristics, such as looking for files with high entropy, that were first observed recently, and that are confined to a geographic region that fits the profile of the activity group being investigated.

Carving the data still yields large data sets that can't be manually analyzed, and advanced threat analytics can help in sorting through the data for meaningful information in the second step. Graph inferences through the Microsoft intelligent security graph can bubble pieces of information to the top of the queue for a hunter to choose from. In the PLATINUM investigation, we identified 31 files.

Lastly, the hunter works directly with the resulting set. During this stage of the PLATINUM investigation, a hunter found a file with unusual string ("*.hotp1*"). The hunter's experience and intuition drove him to dig deeper. In this case, that further investigation led us to the malicious use of hotpatching by this activity group and the "nugget" was uncovered.

Deconstructing the attack

So what is hotpatching? Hotpatching is a previously supported OS feature for installing updates without having to reboot or restart a process. It requires administrator-level permissions, and at a high level, a hotpatcher can transparently apply patches to executables and DLLs in actively running processes.

Using hotpatching in a malicious context is a technique that can be used to avoid being detected, as many antimalware solutions monitor non-system processes for regular injection methods, such as CreateRemoteThread. Hotpatching originally shipped with Windows Server 2003 and was used to ship 10 patches to Windows Server 2003. Windows 10, our most secure operating system ever, is not susceptible to this and many other techniques and attack vectors.

What this means in practical terms is that PLATINUM was able to abuse this feature to hide their backdoor from the behavioral sensors of many host security products. We first observed a sample employing the hotpatching technique on a machine in Malaysia. This allowed PLATINUM to gain persistent access to the networks of companies it targeted and victimized over a long period without being detected.

Thwarting the bad guys

The Microsoft APT hunter team actively tracks activity groups like PLATINUM. We proactively identify these groups and the techniques they use and work to address vulnerabilities and implement security mitigations. The team builds detections and threat intelligence that are utilized by many of our products and services. Beta users of Windows Defender ATP can take advantage of this additional layer of protection and intelligence for a broad set of activity groups.

We've included a more technical exploration of our research and detection of the hotpatching technique in the remainder of this blog.

You can also see a closer look at the PLATINUM activity group in our report PLATINUM: Targeted attacks in South and Southeast Asia. Windows Defender Advanced Threat Protection beta and preview users can also find the report, along with other APT activity group reports, in the Windows Defender ATP portal.

We continue to dig for PLATINUM.

The Windows Defender Advanced Threat Hunting Team

Hotpatching – a case study

We first observed the sample (Sample1) that is capable of utilizing hotpatching on a machine in Malaysia (which matches the general target profile of PLATINUM) on January 28, 2016 . The portable executable (PE) timestamp, which can be arbitrarily set by the adversary, dates back to August 9, 2015, while the unpacked version contains a PE timestamp for November 26, 2015.

It is a DLL that runs as a service and serves as an injector component of a backdoor. Interestingly, this sample not only supported the hotpatching technique described in this post, but was able to apply more common code-injection techniques, including the following, into common Windows processes (primarily targeting winlogon.exe, lsass.exe and svchost.exe):

- CreateRemoteThread
- NtQueueApcThread to run an APC in a thread in the target process
- RtlCreatUserThread
- NtCreateThreadEx

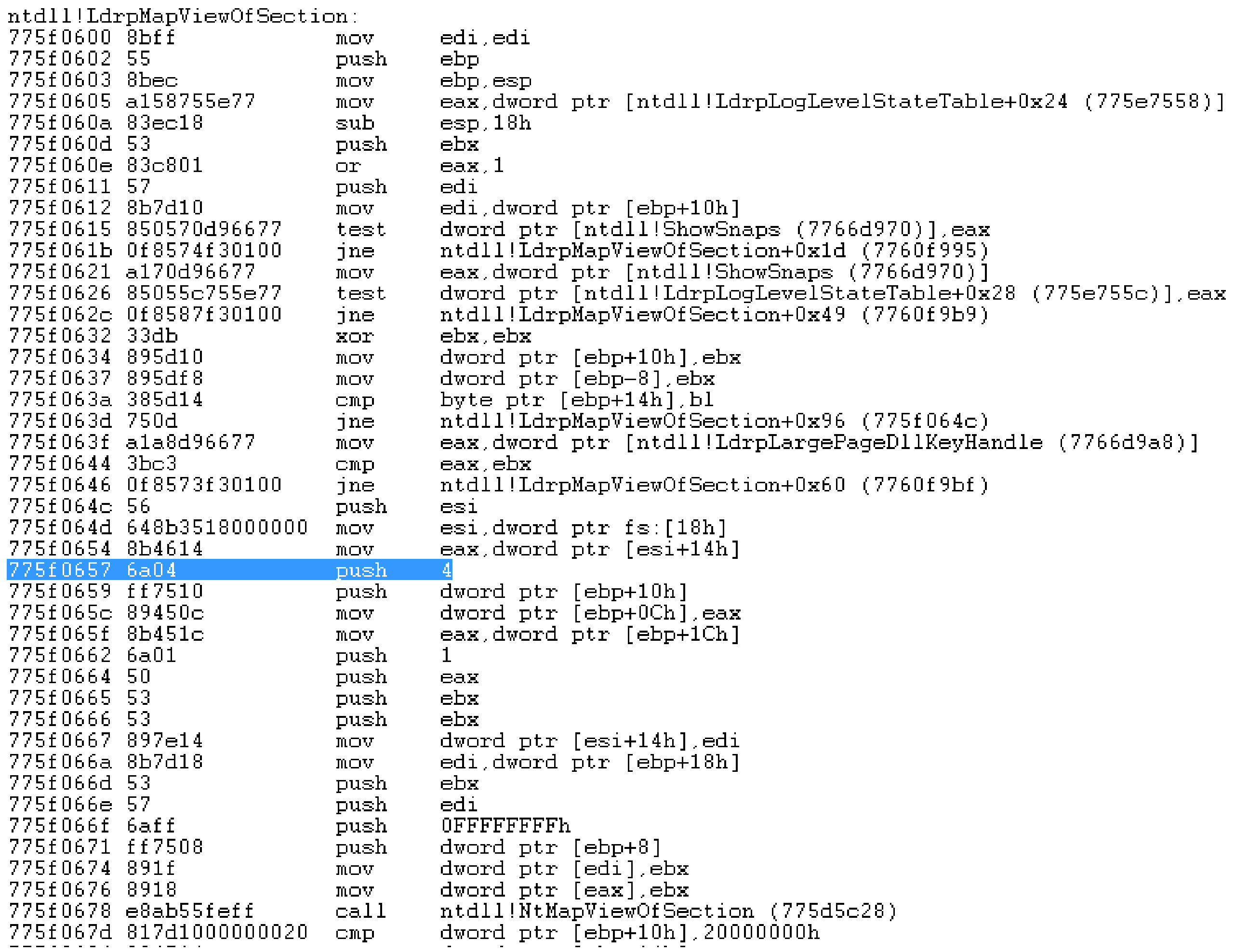
Hotpatching technique

For hotpatching, the sample goes through the following steps:

1. It patches the loader with a proper hotpatch to treat injected DLLs with execute page permissions. This step is required for DLLs loaded from memory (in an attempt to further conceal the malicious code).
2. The backdoor is injected into svchost using the hotpatch API.

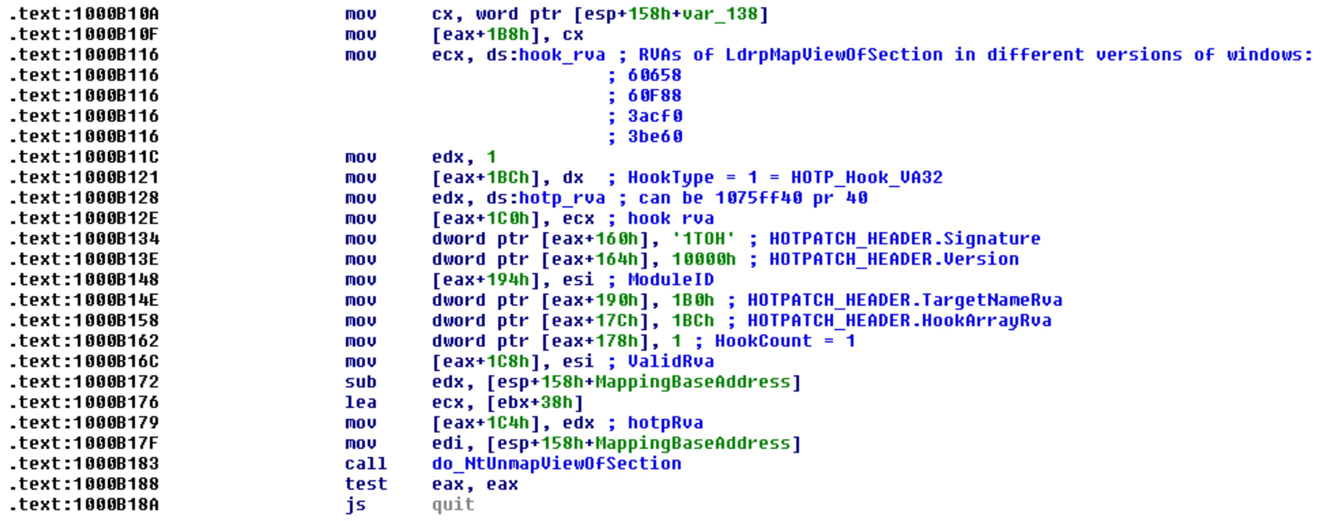
Patching the loader is done by creating a section named “\knowndlls\mstbl.dll”. This DLL does not reside on-disk, but is rather treated as a cached DLL by the session manager.

It then proceeds to write a PE file within that section. The PE file will have one section (“.hotp1 “) with the hotpatch header structure. This structure contains all the information necessary to perform the patching of the function “ntdll!LdrpMapViewOfSection” used by the loader, such that the loader will treat created sections as PAGE_EXECUTE_READWRITE instead of PAGE_READWRITE. The patch is successfully applied by invoking NtSetSystemInformation.



```
ntdll!LdrpMapViewOfSection:
775f0600 8bff            mov     edi,edi
775f0602 55              push    ebp
775f0603 8bec            mov     ebp,esp
775f0605 a158755e77      mov     eax,dword ptr [ntdll!LdrpLogLevelStateTable+0x24 (775e7558)]
775f060a 83ec18          sub     esp,18h
775f060d 53              push    ebx
775f060e 83c801          or      eax,1
775f0611 57              push    edi
775f0612 8b7d10          mov     edi,dword ptr [ebp+10h]
775f0615 850570d96677    test    dword ptr [ntdll!ShowSnaps (7766d970)],eax
775f061b 0f8574f30100    jne     ntdll!LdrpMapViewOfSection+0x1d (7760f995)
775f0621 a170d96677      mov     eax,dword ptr [ntdll!ShowSnaps (7766d970)]
775f0626 85055c755e77    test    dword ptr [ntdll!LdrpLogLevelStateTable+0x28 (775e755c)],eax
775f062c 0f8587f30100    jne     ntdll!LdrpMapViewOfSection+0x49 (7760f9b9)
775f0632 33db            xor     ebx,ebx
775f0634 895d10          mov     dword ptr [ebp+10h],ebx
775f0637 895df8          mov     dword ptr [ebp-8],ebx
775f063a 385d14          cmp     byte ptr [ebp+14h],bl
775f063d 750d            jne     ntdll!LdrpMapViewOfSection+0x96 (775f064c)
775f063f a1a8d96677      mov     eax,dword ptr [ntdll!LdrpLargePageDllKeyHandle (7766d9a8)]
775f0644 3bc3            cmp     eax,ebx
775f0646 0f8573f30100    jne     ntdll!LdrpMapViewOfSection+0x60 (7760f9bf)
775f064c 56              push    esi
775f064d 648b3518000000  mov     esi,dword ptr fs:[18h]
775f0654 8b4614          mov     eax,dword ptr [esi+14h]
775f0657 6a04            push    4
775f0659 ff7510          push    dword ptr [ebp+10h]
775f065c 89450c          mov     dword ptr [ebp+0Ch],eax
775f065f 8b451c          mov     eax,dword ptr [ebp+1Ch]
775f0662 6a01            push    1
775f0664 50              push    eax
775f0665 53              push    ebx
775f0666 53              push    ebx
775f0667 897e14          mov     dword ptr [esi+14h],edi
775f066a 8b7d18          mov     edi,dword ptr [ebp+18h]
775f066d 53              push    ebx
775f066e 57              push    edi
775f066f 6aff            push    0FFFFFFFFh
775f0671 ff7508          push    dword ptr [ebp+8]
775f0674 891f            mov     dword ptr [edi],ebx
775f0676 8918            mov     dword ptr [eax],ebx
775f0678 e8ab55feff      call    ntdll!NtMapViewOfSection (775d5c28)
775f067d 817d1000000020  cmp     dword ptr [ebp+10h],20000000h
```

Figure 1: The malware builds the information describing the first patch



```
.text:1000B10A                 mov     cx, word ptr [esp+158h+var_138]
.text:1000B10F                 mov     [eax+1B8h], cx
.text:1000B116                 mov     ecx, ds:hook_rva ; RVAs of LdrpMapViewOfSection in different versions of windows:
.text:1000B116                                         ; 60658
.text:1000B116                                         ; 60F88
.text:1000B116                                         ; 3acf0
.text:1000B116                                         ; 3be60
.text:1000B11C                 mov     edx, 1
.text:1000B121                 mov     [eax+1BCh], dx  ; HookType = 1 = HOTP_Hook_VA32
.text:1000B128                 mov     edx, ds:hotp_rva ; can be 1075ff40 pr 40
.text:1000B12E                 mov     [eax+1C0h], ecx ; hook rva
.text:1000B134                 mov     dword ptr [eax+160h], '1TOH' ; HOTPATCH_HEADER.Signature
.text:1000B13E                 mov     dword ptr [eax+164h], 10000h ; HOTPATCH_HEADER.Version
.text:1000B148                 mov     [eax+194h], esi ; ModuleID
.text:1000B14E                 mov     dword ptr [eax+190h], 1B0h ; HOTPATCH_HEADER.TargetNameRva
.text:1000B158                 mov     dword ptr [eax+17Ch], 1BCh ; HOTPATCH_HEADER.HookArrayRva
.text:1000B162                 mov     dword ptr [eax+178h], 1 ; HookCount = 1
.text:1000B16C                 mov     [eax+1C8h], esi ; ValidRva
.text:1000B172                 sub     edx, [esp+158h+MappingBaseAddress]
.text:1000B176                 lea     ecx, [ebx+38h]
.text:1000B179                 mov     [eax+1C4h], edx ; hotpRva
.text:1000B17F                 mov     edi, [esp+158h+MappingBaseAddress]
.text:1000B183                 call    do_NtUnmapViewOfSection
.text:1000B188                 test    eax, eax
.text:1000B18A                 js      quit
```

Figure 2: The highlighted “push 4” is patched to “push 0x40”, meaning that the parameter for the following API call NtMapViewOfSection is changed from PAGE_READWRITE to PAGE_EXECUTE_READWRITE.

Now that the memory permission issue has been solved, the injector can proceed with injecting the malicious DLL into svchost. Again, it creates a (now executable) section named “knowndlls\fgrps.dll” and invokes NtSetSystemInformation, causing the final payload to be loaded and executed within the target process (svchost).

Trying to hide the payload using hotpatching also falls in line with the last functional insights we have on the sample. It seems to have an expiry date of January 15, 2017 – at that point in time, the DLL will no longer perform the injection, but rather execute another PLATINUM implant:

C:\program files\Windows Journal\Templates\Cpl\jnwmon.exe –ua

This implant may be related to an uninstall routine. Note that we observed the sample last on the machine on September 3, 2015, which may indicate PLATINUM pulled the trigger earlier.

---

[1] http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Sotirov.pdf

[2] https://www.yumpu.com/en/document/view/14255220/alexsyscan13

Talk to us

Questions, concerns, or insights on this story? Join discussions at the Microsoft community and Windows Defender Security Intelligence.