ENDGAME.

Another 0day, Another Prevention

SERIES ONE, VOLUME ONE

CODY PIERCE

A new 0day against the popular browser Firefox was revealed yesterday which specifically targets the popular "Tor Browser" project, a favorite of Tor users. The Endgame Vulnerability Research & Prevention team quickly analyzed the exploit from the original post, as well as a clean version of reduced JavaScript.

This vulnerability appears to be a fairly typical Use-After-Free (UAF) when animating SVG content via JavaScript. In general, when exploiting UAF vulnerabilities there are a few key steps to gain code execution, which also occur here. The code first must create a situation where an object is allocated and used. Next, the memory of the process must be arranged so that when the "use after free" happens, code execution is gained.

In this exploit the initial HTML page includes JavaScript to first set up some global variables used to communicate to the initial page as well as a Web Worker thread responsible for the majority of the exploit, including heap grooming, memory reading/writing, ROP creation, and payload storage.

In the cleaned version of the worker JavaScript it is evident that the exploit first creates three utility objects. First, the Memory class is used to precisely read and write arbitrary virtual memory. By exposing a few functions it gives the exploit writer a clean interface to bypass ASLR, as well as store the ROP and payload into memory. The PE class is next. This is designed to dynamically resolve import functions for use in the ROP chain. The final helper class is ROP. Its purpose is to dynamically scan memory for ROP gadget bytes and generate a ROP chain. This is done so the author can target multiple versions of Firefox without requiring version-specific information beforehand.

The bulk of the worker thread is designed to drive the exploitation in the following steps:

1. Prepare memory by allocating a large number of objects to address 0x30000030.

2. Trigger the usage of the free memory in the original exploit function which will corrupt the allocated heap blocks.

3. Use the memory class to leak the base address of XUL.dll.

4. Use the PE class to resolve the import address of Kernel32.dll!CreateThread.

5. Dynamically construct the ROP chain using the ROP class.

6. Store the addresses of the ROP chain and payload in memory.

7. Trigger the control flow hijack that calls the stack pivot to initiate the ROP chain.

8. Gain code executation and execute the payload stored in the JavaScript variable "thecode".

While this vulnerability was surely discovered while fuzzing, the exploit writer wasn't a novice. The use of the helper functions to dynamically build the payload is of moderate skill.

Prevention

After analyzing the exploit we quickly wanted to determine how our Endgame exploit prevention techniques would do against this never-before-seen vulnerability and exploit.

Endgame has two distinct approaches to exploit prevention. One uses our novel Hardware-Assisted Control-Flow-Integrity (HA-CFI) that can predict when a branch misprediction is about to execute malicious instructions. The other consists of dynamically inserted checks that are performed during program execution.

When testing HA-CFI against exploits, we really want to catch the exploit before that ROP chain. This gives the defender an earlier place to determine whether something

"bad" is about to happen. In this instance, because we know this UAF will eventually alter control flow, we were able to successfully detect that first change. Using IDA, we have visualized the detection information straight from HA-CFI's alert. The source is the exact location of the control-flow hijack, and the destination is the beginning of the ROP chain. It is important to note that we block the destination address from ever executing *(see below).*

Pretty neat! As an added bonus it helps us when reverse engineering exploits because we know exactly where the bug is.

As mentioned earlier, we also have a dynamic ability to detect exploits as well. We call this DBI, for Dynamic Binary Instrumentation. When talking about exploit prevention, there is early and late stage detection. Late detection is typical of most exploit prevention software. For an experienced exploit writer, the later a detection the easier it is to bypass. This is why it is essential to detect exploits as early as possible with innovative techniques such as HA-CFI and new DBI checks.

Our recently released DBI prevention check has proven to detect early against this vulnerability. The new protection is designed to detect when an attacker is attempting to dynamically resolve PE imports, and construct a ROP chain. By doing this we actually catch the exploit in Step 4 above. Below is some output from a debug version of the software showing exactly what happened. *(see next page).*

HA-CFI Source:

```
.text:6A68DDDD    call    dword ptr [eax+14Ch]
.text:6A68DDE3    test    eax, eax
.text:6A68DDE5    jnz     short loc_6A68DDED
```

HA-CFI Destination:

```
.text:69C60C9B    xchg    eax, esp
.text:69C60C9C    retn
```

```
[DBI debug]
HeaderProtect DETECTION
Image Type: MEM_PRIVATE
Image Type: MEM_IMAGE
Image File: C:\Users\vrp32\Desktop\Tor Browser\Browser\xul.dll
VRP_ALERT_ADDRESS: 0364D2C2 => 695C003C (0x00000000ffffff000)
```

You can see that the exploit tried to read the header of xul.dll (base address 0x695c0000). And if we disassemble the source of the read we see an executable JIT page making the offending read *(see below)*.

```
0364D2C2                mov     esi, [esi+ebx*4] ; read xul.dll header + 0x3C
0364D2C5                test    esi, esi
0364D2C5 ; ---------------------------------------------------------------------------
```

Stay tuned for the next release, where some of our latest research is designed to catch exploits even earlier, in Stage 1.

Conclusion

At this point, 0days should not surprise anyone. They have been, and will be, a regular occurrence. However, in this case targeting Firefox and Tor is particularly unique, as Tor is often used for illegal purposes, and exploiting the user's browser is a quick way to "de-anonymize" the user. Also, unlike most exploit kits, this exploit doesn't appear to be highly obfuscated, which also differentiates it from other 0days.

Mozilla should be releasing a patch today or tomorrow, and users are urged to upgrade if they are using Firefox. In the meantime Endgame users are protected, and everyone else should disable JavaScript.

ENDGAME.