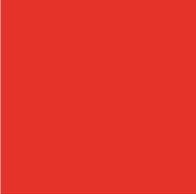


# iOS/macOS 0-day<sup>w</sup>48-hours

from sandbox to kernel

 Présenté 31/05/2018

Pour BeeRumP

Par Eloi Vanderbeken



# Who?

- Eloi Vanderbeken
- @elvanderb
- Synacktiv



- **Coordinateur du pôle reverse de Synacktiv**
  - développement bas niveau
  - reverse
  - recherche et exploitation de vulnérabilités
  - 12 personnes (31 pour tout Synacktiv)
- **On recrute !**

# XNU



- **Kernel de macOS / iOS / watchOS**
- **BSD based**
- **OpenSource**
  - ne contient pas tous les kexts
- **Surface d'attaque intéressante**
  - large partie accessible depuis un processus sandboxé
  - étonnamment relativement peu (publiquement) audité (par rapport aux kext)

# Où taper ?



## ■ Réseau

- compliqué
- beaucoup de data user controlled
- async

## ■ Nouveau code

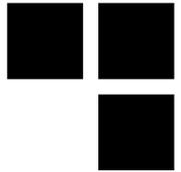
- Apple n'est pas connu pour ses tests unitaires de qualité...
  - cf. CVE-2017-2370...
- Facile à identifier

## ■ MPTCP !

# MPTCP



- **MultiPath TCP**
- **Sert à faire du TCP sur plusieurs canaux**
  - cellulaire + wifi par exemple
- **Meilleurs perfs + résilience**
- **Utilisé par iOS depuis iOS 7 pour Siri**
- **Interface publique depuis iOS 11**
- **Dans XNU depuis ~ xnu-2422.1.72 (fin 2013)**



# Concrètement...

## ■ `bsd/kern/uipc_domain.c`

```
706 void
707 domaininit(void)
708 {
709     struct domain *dp;
710     domain_guard_t guard;
711
712     ● ● ●
713
739     attach_domain(&inetdomain_s);
740 #if INET6
741     attach_domain(&inet6domain_s);
742 #endif /* INET6 */
743 #if MULTIPATH
744     attach_domain(&mpdomain_s);
745 #endif /* MULTIPATH */
746     attach_domain(&systemdomain_s);
```

# Concrètement...



## ■ `bsd/netinet/mp_proto.c`

```
59  struct domain mpdomain_s = {
60      .dom_family =      PF_MULTIPATH,
61      .dom_flags =      DOM_REENRANT,
62      .dom_name =      "multipath",
63      .dom_init =      mp_dinit,
64  };
65
66  /* Initialize the PF_MULTIPATH domain, and add in the pre-defined protos */
67  void
68  mp_dinit(struct domain *dp)
69  {
70      VERIFY(!(dp->dom_flags & DOM_INITIALIZED));
71
72      net_add_proto(&mpsw, dp, 1);
73  }
```

# Concrètement...



## ■ `bsd/netinet/mp_proto.c`

```
45  static struct protosw mpsw = {
46      .pr_type =          SOCK_STREAM,
47      .pr_protocol =     IPPROTO_TCP,
48      .pr_flags =        PR_CONNREQUIRED|PR_MULTICONN|PR_EVCONNINFO|
49                          PR_WANTRCVD|PR_PCBLCK|PR_PROTOLOCK|
50                          PR_PRECONN_WRITE|PR_DATA_IDEMPOTENT,
51      .pr_ctloutput =    mptcp_ctloutput,
52      .pr_init =         mptcp_init
53      .pr_usrreqs =       &mptcp_usrreqs,
54      .pr_lock =         mptcp_lock,
55      .pr_unlock =       mptcp_unlock,
56      .pr_getlock =      mptcp_getlock,
57  };
--
```

# Concrètement...



## ■ `bsd/netinet/mptcp_usrreq.c`

```
86  struct pr_usrreqs mptcp_usrreqs = {
87      .pru_attach =          mptcp_usr_attach,
88      .pru_connectx =       mptcp_usr_connectx,
89      .pru_control =        mptcp_usr_control,
90      .pru_detach =         mptcp_usr_detach,
91      .pru_disconnect =     mptcp_usr_disconnect,
92      .pru_disconnectx =    mptcp_usr_disconnectx,
93      .pru_peeraddr =       mp_getpeeraddr,
94      .pru_rcvd =           mptcp_usr_rcvd,
95      .pru_send =           mptcp_usr_send,
96      .pru_shutdown =       mptcp_usr_shutdown,
97      .pru_sockaddr =       mp_getsockaddr,
98      .pru_sosend =         mptcp_usr_sosend,
99      .pru_soreceive =      soreceive,
100     .pru_socheckopt =      mptcp_usr_socheckopt,
101     .pru_preconnect =      mptcp_usr_preconnect,
102 };
```

# Concrètement...



## ■ `bsd/netinet/mptcp_usrreq.c`

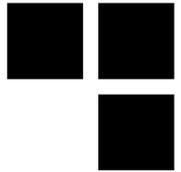
```
285  /*
286   * User-protocol pru_connectx callback.
287   */
288  static int
289  mptcp_usr_connectx(struct socket *mp_so, struct sockaddr *src,
290                   struct sockaddr *dst, struct proc *p, uint32_t ifscope,
291                   sae_associd_t aid, sae_connid_t *pcid, uint32_t flags, void *arg,
292                   uint32_t arglen, struct uio *auio, user_ssize_t *bytes_written)
293  {
294  #pragma unused(p, aid, flags, arg, arglen)
```

# SANITY CHECKS!!!!



## ■ `bsd/netinet/mptcp_usrreq.c`

```
322     if (dst->sa_family == AF_INET &&
323         dst->sa_len != sizeof(mpte->__mpte_dst_v4)) {
324         mptcplog((LOG_ERR, "%s IPv4 dst len %u\n", __func__,
325                 dst->sa_len),
326                 MPTCP_SOCKET_DBG, MPTCP_LOGLVL_ERR);
327         error = EINVAL;
328         goto out;
329     }
330
331     if (dst->sa_family == AF_INET6 &&
332         dst->sa_len != sizeof(mpte->__mpte_dst_v6)) {
333         mptcplog((LOG_ERR, "%s IPv6 dst len %u\n", __func__,
334                 dst->sa_len),
335                 MPTCP_SOCKET_DBG, MPTCP_LOGLVL_ERR);
336         error = EINVAL;
337         goto out;
338     }
```

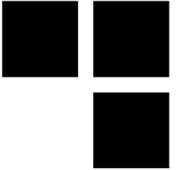


# else ?

## ■ `bsd/netinet/mptcp_usrreq.c`

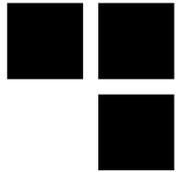
```
337         goto out;
338     }
339
340     if (!(mpte->mpte_flags & MPTE_SVCTYPE_CHECKED)) {
341         if (mptcp_entitlement_check(mp_so) < 0) {
342             error = EPERM;
343             goto out;
344         }
345
346         mpte->mpte_flags |= MPTE_SVCTYPE_CHECKED;
347     }
348
349     if ((mp_so->so_state & (SS_ISCONNECTED|SS_ISCONNECTING)) == 0) {
350         memcpy(&mpte->mpte_dst, dst, dst->sa_len);
351     }
352
```

# else ?



## ■ `bsd/netinet/mptcp_usrreq.c`

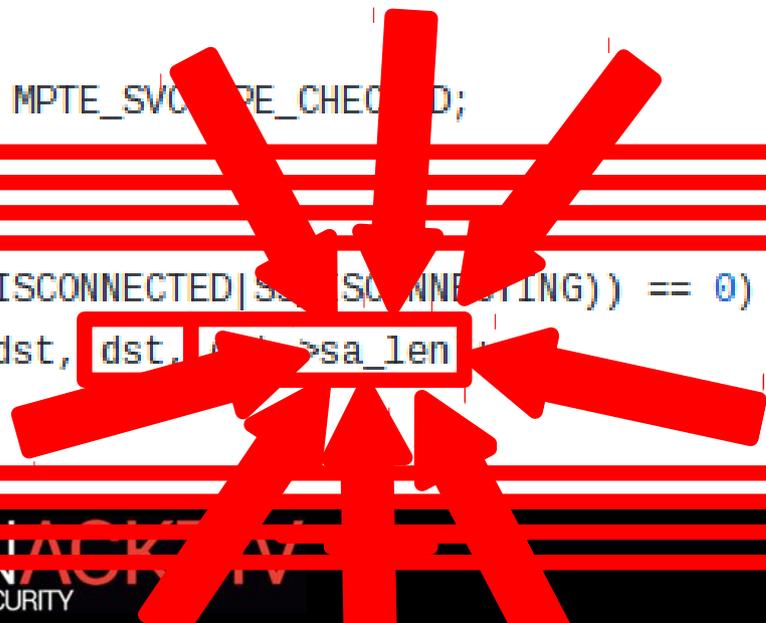
```
337         goto out;
338     }
339
340     if (!(mpte->mpte_flags & MPTE_SVCTYPE_CHECKED)) {
341         if (mptcp_entitlement_check(mp_so) < 0) {
342             error = EPERM;
343             goto out;
344         }
345
346         mpte->mpte_flags |= MPTE_SVCTYPE_CHECKED;
347     }
348
349     if ((mp_so->so_state & (SS_ISCONNECTED|SS_ISCONNECTING)) == 0) {
350         memcpy(&mpte->mpte_dst, dst, dst->sa_len);
351     }
```



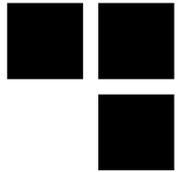
# else ?

## ■ `bsd/netinet/mptcp_usrreq.c`

```
337         goto out;
338     }
339
340     if (!(mpte->mpte_flags & MPTE_SVCTYPE_CHECKED)) {
341         if (mptcp_entitlement_check(mp_so) < 0) {
342             error = EPERM;
343             goto out;
344         }
345
346         mpte->mpte_flags |= MPTE_SVCTYPE_CHECKED;
347     }
348
349     if ((mp_so->so_state & (SS_ISCONNECTED|SS_ISCONNECTING)) == 0) {
350         memcpy(&mpte->mpte_dst, dst, mp_so->sa_len);
351     }
352 }
```



# Est-ce que c'est vraiment grave...



## ■ `bsd/netinet/mptcp_var.h`

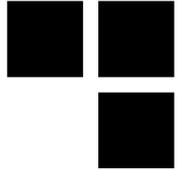
```
52  /*
53   * MPTCP Session
54   *
55   * This is an extension to the multipath PCB specific for MPTCP, protected by
56   * the per-PCB mpp_lock (also the socket's lock);
57   */
58  struct mptses {
59      struct mppcb    *mpte_mppcb;          /* back ptr to multipath PCB */
60      struct mptcb    *mpte_mptcb;        /* ptr to MPTCP PCB */
61
62
63
64
65
66
67
68
69
70
71
72
73      };
74
75      union {
76          /* Destination address of initial subflow */
77          struct sockaddr mpte_dst;
78          struct sockaddr_in __mpte_dst_v4;
79          struct sockaddr_in6 __mpte_dst_v6;
80      };
81
82      struct mptsub    *mpte_active_sub;    /* ptr to last active subf */
```



**28 bytes MAX**



# Patché :'(



## ■ Kernel iOS 11.4

```
84 sa_family = dst_>sa_family;
85 if ( sa_family == AF_INET6 )
86 {
87     if ( dst_>sa_len != 28 )
88     {
89         if ( !(dword_FFFFFFFF00762AA38 & 2) || !(byte_FFFFFFFF00762AA90 & 1) )
90             return 22LL;
91         goto LABEL_15;
92     }
93 }
94 else
95 {
96     if ( sa_family != AF_INET )
97         return EAFNOSUPPORT;
98     if ( dst_>sa_len != 16 )
99     {
100         if ( !(dword_FFFFFFFF00762AA38 & 2) || !(byte_FFFFFFFF00762AA90 & 1) )
101             return 22LL;
102         goto LABEL_15;
103     }
```

# Blame!

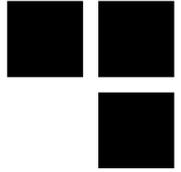


- **Vulnérabilité introduite par xnu-4570.1.46**
  - d'après les sources...
- **Avant la destination était placée dans un buffer malloc**
- **Les nouvelles vulns ne vivent pas vieilles**
  - ~ 8 mois
  - y'a plus d'jeunesse ma bonne dame

# WEN JB ETA?



- **Pas le temps de parler de l'exploitation**
  - pas eu le temps de regarder
  - très nombreuses mitigations
  - R/W arbitraire to root + entité : long
  - de nombreuses publications sur le sujet
- **Vulnérabilité exploitable que depuis une app « normale »**
  - pas depuis un renderer sans accès au réseau :)
- **Ça reste une rump préparée en 1h :)**

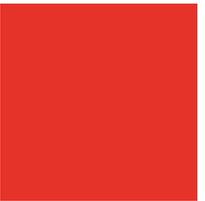


# Conclusion

- **Vulnérabilité simple à trouver**
- **Probablement simple à exploiter**
- **Atteignable depuis la sandbox**
  - sous condition
- **Qui a vécu 8 mois**
  - en même temps peu et beaucoup
  - a survécu à plusieurs updates
- **Si vous voulez faire des trucs cools...**
  - Synaktiv recrute



AVEZ-VOUS  
DES QUESTIONS ?



MERCI DE VOTRE ATTENTION,