



iService: Detecting and Evaluating the Impact of Confused Deputy Problem in AppleOS

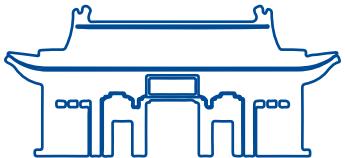
Yizhuo Wang, Yikun Hu, Xuangan Xiao, Dawu Gu

Shanghai Jiao Tong University

The Annual Computer Security Applications Conference (ACSAC 2022)

9 December, 2022

CONTENTS



- 01 Background**
- 02 Challenges**
- 03 Design of iService**
- 04 Evaluation**

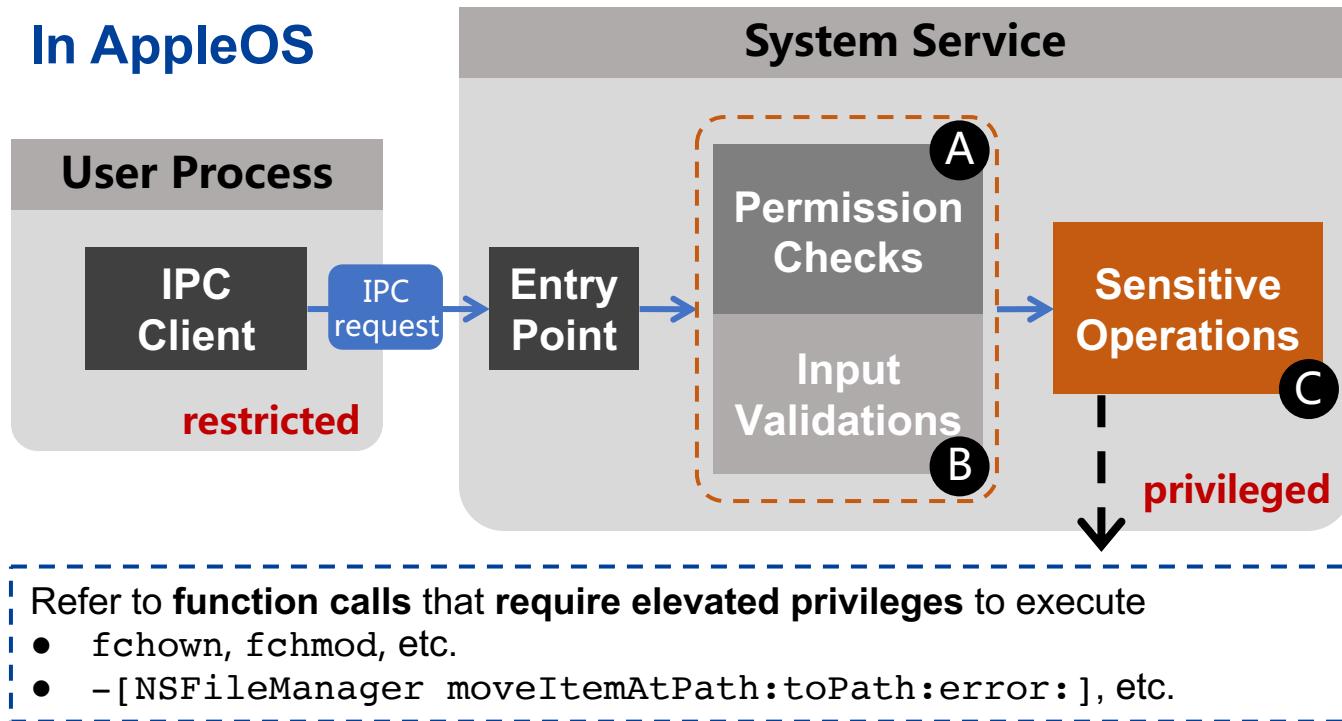
Confused Deputy Problem



Confused deputy, a specific type of privilege escalation

A **restricted entity abuses a more-privileged entity to perform a sensitive operation that should not be allowed.**

In AppleOS



Root Cause

Missing or lacking proper permission checks and input validations



Arbitrary file overwriting, code execution, etc.

Detect Confused Deputies

- Identify sensitive operations in system services
- Extract protections for sensitive operations

Challenges for detection



Motivating example: CVE-2021-30774

```
1 listener = xpc_connection_create_mach_service(Init  
    "com.apple.osanalytics.osanalyticshelper", 0, 1);  
2 service = objc_msgSend(&OBJC_CLASS__OSAXPCServices, "init");  
3 xpc_connection_set_event_handler(listener, &block1); call  
  
4 if (xpc_get_type(conn) == XPC_TYPE_CONNECTION){ handler_1  
5     xpc_connection_set_event_handler(conn, &block2); call  
  
6     service=objc_loadWeakRetained(block2->lvar3); handler_2  
7     if (xpc_get_type(req) == XPC_TYPE_DICTIONARY){  
8         pid = xpc_connection_get_pid(block2->lvar2);  
9         objc_msgSend(service, "serviceRequest:fromPID:forReply:",  
            req, pid, reply); } call  
  
10    op = xpc_dictionary_get_uint64(req, "operation"); dispatcher  
11    switch (op){  
12        case 6:  
13            ret = objc_msgSend(&OBJC_CLASS_$_OSALogHelper,  
                "createForSubmissionWithXPCRequest:forReply:", req, reply);  
                ↓ through multiple calls to code snippets in another binary  
14            dst = objc_msgSend(req, "objectForKeyedSubscript:",  
                CFSTR("override filePath"));  
15            src = objc_msgSend(req, "objectForKeyedSubscript:",  
                CFSTR("move file"));  
/* Eliminate a lot of tedious processes */  
16            fchown(src,-1,250u) && fchmod(src,0660);  
17            objc_msgSend(fileMgr, "moveItemAtPath:toPath:error:",  
                src, dst, &err);
```

Sensitive operations:

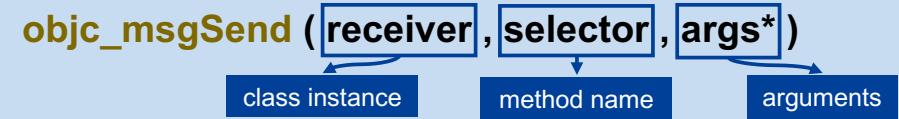
1. Change permission of file **src**
2. Move file **src** to **dst**

✓ Overwrite arbitrary files
✓ Obtain root privilege

Input fields without validations

Challenge 1: Resolving Objective-C messages

Dynamically determined indirect call: message send



Necessary to infer the type of message receiver

Line 2: type **OSAXPCServices** → var **service**

Lines 3 and 5: callback function invocation

Line 6: assign **OSAXPCServices** → var **service**

Challenges for detection



Motivating example: CVE-2021-30774

```
1 listener = xpc_connection_create_mach_service(Init  
  "com.apple.osanalytics.osanalyticshelper", 0, 1);  
2 service = objc_msgSend(&OBJC_CLASS__OSAXPCServices, "init");  
3 xpc_connection_set_event_handler(listener, &block1); call  
  
4 if (xpc_get_type(conn) == XPC_TYPE_CONNECTION){ handler_1  
5   xpc_connection_set_event_handler(conn, &block2); call  
  
6 service=objc_loadWeakRetained(block2->lvar3); handler_2  
7 if (xpc_get_type(req) == XPC_TYPE_DICTIONARY){  
8   pid = xpc_connection_get_pid(block2->lvar2);  
9   objc_msgSend(service, "serviceRequest:fromPID:forReply:",  
    req, pid, reply); } call  
  
10 op = xpc_dictionary_get_uint64(req, "operation"); dispatcher  
11 switch (op){  
12   case 6:  
13     ret = objc_msgSend(&OBJC_CLASS_$_OSALogHelper,  
      "createForSubmissionWithXPCRequest:forReply:", req, reply);  
      through multiple calls to code snippets in another binary  
14 dst = objc_msgSend(req, "objectForKeyedSubscript:",  
    CFSTR("override filePath"));  
15 src = objc_msgSend(req, "objectForKeyedSubscript:",  
    CFSTR("move file"));  
/* Eliminate a lot of tedious processes */  
16 fchown(src,-1,250u) && fchmod(src,0660);  
17 objc_msgSend(fileMgr, "moveItemAtPath:toPath:error:",  
  src, dst, &err);
```

Sensitive operations:

1. Change permission of file **src**
2. Move file **src** to **dst**

✓ Overwrite arbitrary files
✓ Obtain root privilege

Input fields without validations

Challenge 2: Identify sensitive input validations

Input validations have no fixed pattern

Only validations for **sensitive input** matters

Sensitive inputs on which sensitive operations depend

Set **req** of **xpc_dictionary**

"operation": 6

"move-file": src

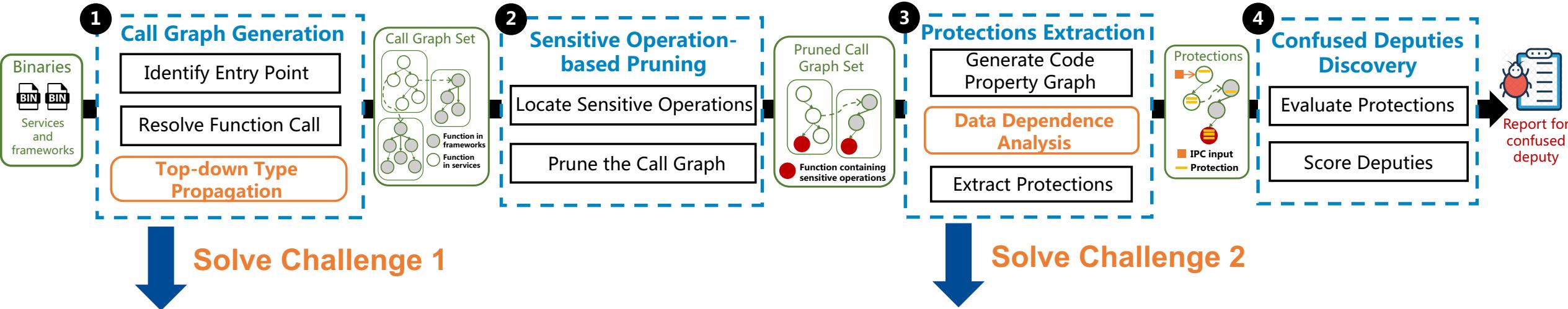
"override-file": dst

"LogType": don't care

Design of iService



Propose **iService**, a static analysis framework to detect confused deputies in AppleOS



Top-down Type Propagation

Insight: type information exists in binary, which are introduced at definition.

Solution: record types, propagate **intra-** and **inter-procedural** until call sites.

Sensitive operation-oriented data dependence analysis

Insight: input validations should restrict the value range of key parameters.

Solution: identify data dependence among **inputs** and **key parameters** of sensitive operations.

Top-down Type Propagation



Maintain a object-type map recording inferred types

Intra-procedural Propagation

Along the control flow

Introduce type information

- Function parameters
- Function prototypes
- Recovered structures

Standard library,
Instantiation functions

Propagate type information

- Assignment-like statement
- Prototypes of private function resolved

Inter-procedural Propagation

Start at entry points, from caller to callee

Parameters passed at call sites

Layout of structures used for callback

Statement level

```
1 obj1 = objc_msgSend(&OBJC_CLASS__OSAXPCServices, "init");
2 block1.isa = _NSConcreteStackBlock;
3 block1.invoke = &handler_1;
4 objc_copyWeak(block1.lvar2, obj1);
5 xpc_connection_set_event_handler(listener, &block1); call Init
```



```
6 if (xpc_get_type(conn) == XPC_TYPE_CONNECTION){ handler_1
7   block2.isa = _NSConcreteStackBlock;
8   block2.invoke = &handler_2;
9   objc_copyWeak(block2.lvar3, block1.lvar2);
10  xpc_connection_set_event_handler(conn, &block2); call handler_2
```



```
11 obj2 =objc_loadWeakRetained(block2->lvar3);
12 if (xpc_get_type(req) == XPC_TYPE_DICTIONARY){
13   objc_msgSend(obj2, "serviceRequest:fromPID:forReply:",
req,pid,reply);}
```

+[OSAXPCServices init]

introduce

obj1 → block1.lvar2 → block1.lvar2 → block2.lvar3 → block2.lvar3 → obj2

resolve objc_msgSend

-[OSAXPCServices serviceRequest:fromPID:forReply]

Data Dependence Analysis



Generate Code Property Graph (CPG)
to abstract binary code

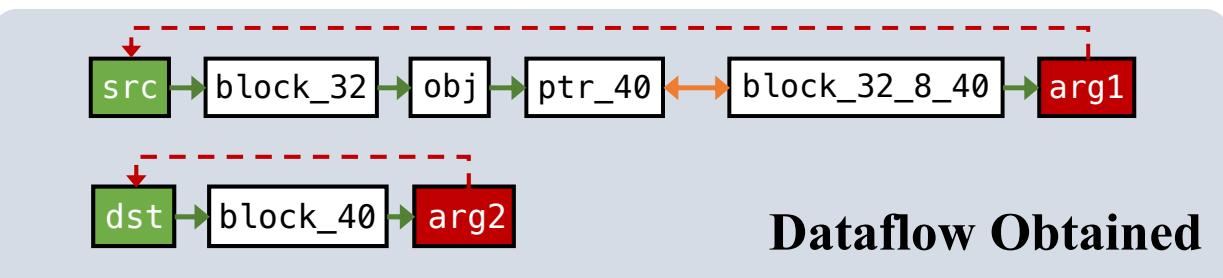
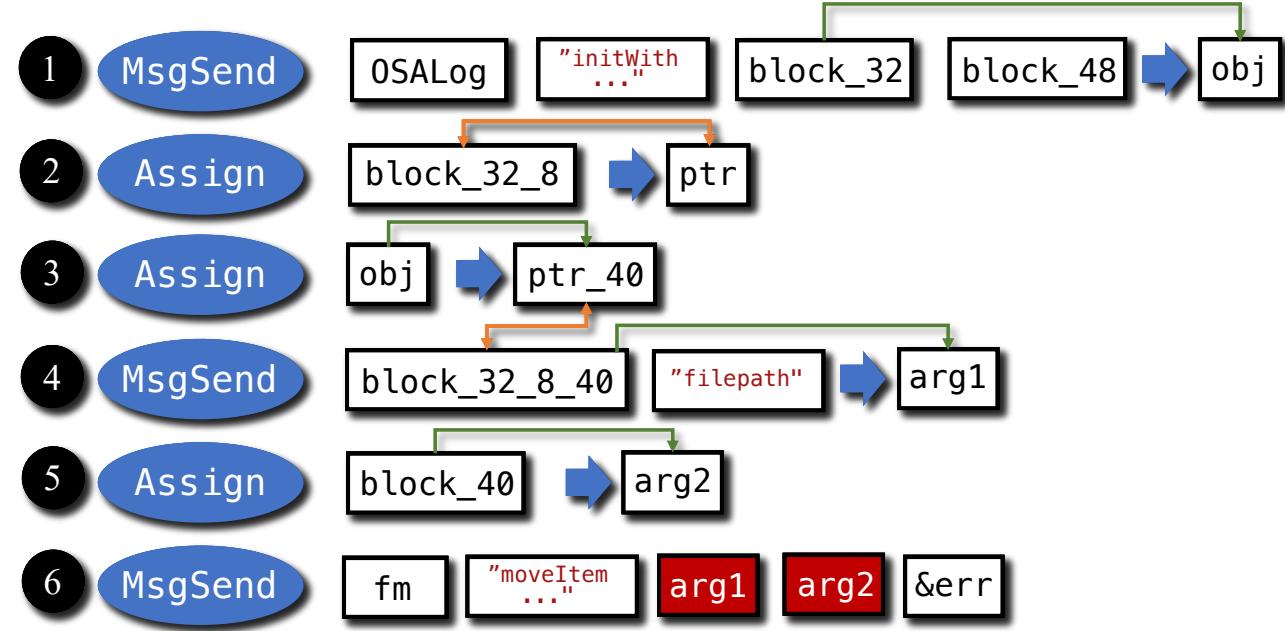
- abstract syntax $G = (V, E, \lambda, \mu)$
- control flow && control&data dependency

Perform alias analysis
to fulfill field-sensitive dataflow

- nested base & offset to identify pointers
- infer alias relationship of sub-fields

Apply summary-based method
to fulfill inter-procedural dataflow

- generate dataflow summary for functions
- produce summary in a bottom-up manner



Evaluation – Overall Result



Implementation: about 4,300 LOC of Python based on the IR of IDA Pro and Neo4j graph database

Dataset: 1,256 daemon binaries collected from 4 AppleOS, macOS 10.14.3, 10.15.7, 11.4, and 12.4

1,256 daemons → 439 system services → 830 entry points → 431 sensitive operations

Table 1: Result of the sensitive operation identification

AppleOS	Daemon	System Service			Entry Points			Sensitive Operation							
		XPC	NSXPC	Total	XPC	NSXPC	Total	A	B	C	D	In service	In Framework	Total	controllable
macOS 10.14.3	293	82	16	98	80	25 (76)	105 (156)	7	38	11	58	105	11	114	31
macOS 10.15.7	301	88	19	107	88	27 (72)	115 (160)	10	39	10	59	106	12	118	27
macOS 11.4	325	96	24	120	113	64 (166)	177 (279)	3	40	11	53	117	–	117	17
macOS 12.4	337	99	15	114	113	69 (122)	182 (235)	2	32	14	21	82	–	82	12
Total	1256	365	74	439	394	185 (436)	579 (830)	29	127	37	194	368	23	431	87

A: refers to the sensitive operation of *File Permissions Manipulation*.

B: refers to the sensitive operation of *System and User Files Manipulation*.

C: refers to the sensitive operation of *Preferences Management*.

D: refers to the sensitive operation of *Process Management*.

Controllable: the sensitive operation has at least one parameter depending on the IPC input.

87 sensitive operations are controllable

Finally, iService reported 20 confused deputies, of which **11** were confirmed to be exploitable (true positives)

Evaluation – Bug Findings



Table 2: Summary of confused deputy vulnerability found.

Vulnerable Service	OS	Cause	Impact	Status
osanalyticshelper	macOS 10.15, 11	Missing permission checks and input validations	Gain root privilege	CVE-2021-30774
fbahelperd	macOS 10.12-14	Weak permission checks and input validations	Overwrite arbitrary files	CVE-2019-8521
fbahelperd	macOS 10.12-14	Weak permission checks and input validations	Gain root privilege	CVE-2019-8565
timemachinehelper	macOS 10.12-14	Missing permission checks and input validations	Execute arbitrary shell command	CVE-2019-8513
timemachinehelper	macOS 10.12-14	Missing permission checks and input validations	Overwrite arbitrary files	CVE-2019-8530
bluetoothhelper	macOS 10.12-14	Missing permission checks and input validations	Overwrite arbitrary files	Repaired
getmobilityinfohelper	macOS 10.12-14	Missing permission checks and input validations	Overwrite arbitrary files	Repaired
fud	macOS 10.14-15	Weak permission checks and input validations	File access and DoS	Repaired
storelegacy	macOS 10.14-15	Missing permission checks and input validations	File access and DoS	Repaired
coresymbolicationd	macOS 10.15, 11	Missing permission checks and input validations	May lead to DoS	Repaired
wifihelper	macOS 10.12-14	Missing permission checks and input validations	File access and DoS	Repaired

11 are true positives, which are confirmed to be exploited

- ✓ **5** are 0-day bugs with CVE numbers assigned
- ✓ **6** are 1-day bugs

Impact of confused deputies found:

- ✓ **4** may lead to arbitrary file writing
- ✓ **2** may lead to root privilege
- ✓ **1** may lead to arbitrary shell command execution

Evaluation – Case Study



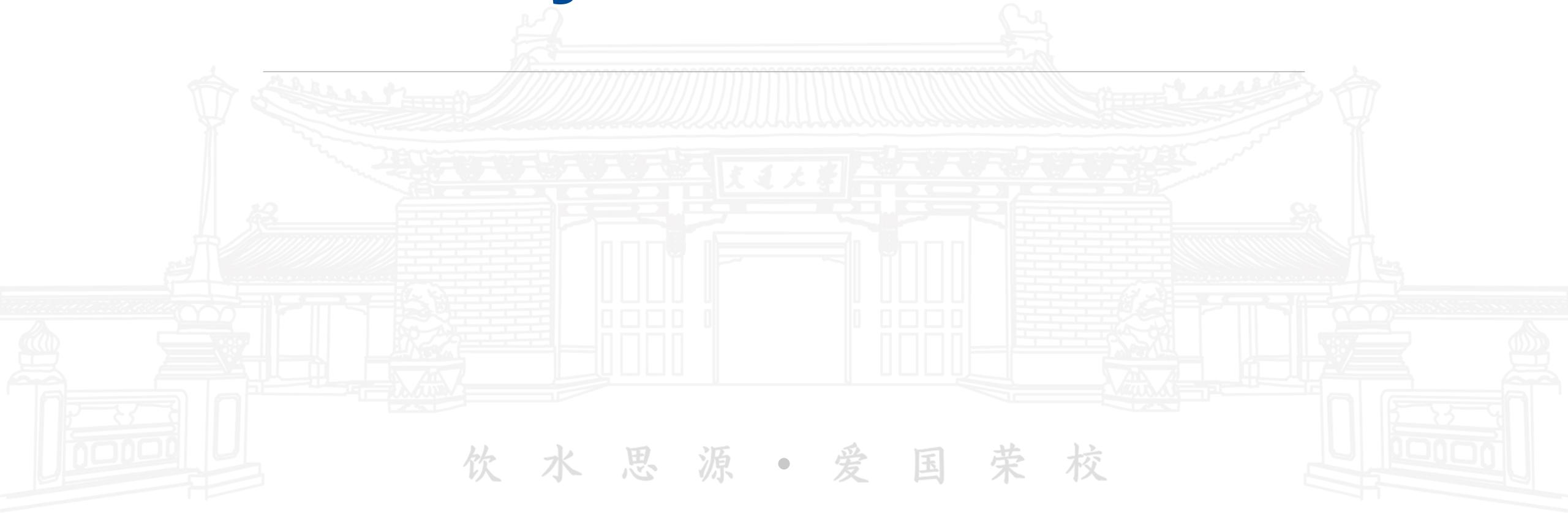
CVE-2019-8521 Arbitrary file overwriting

```
-[FBAPrivilegedDaemon copyLogFiles:] @"$PATHtoEXP"
1 if ([src hasPrefix:@"/Library/Logs"] || [src hasPrefix:@"/var/log"]){
2     if (![self canModifyPath:dst]) {
3         result[src] = [NSString stringWithFormat:@"Invalid
4             destination: %@", dst];
5     } else {
6         result[src] = @"File must be copied from a log directory";
7     }
8 }
... // Skip some complex code
9 [fileMg copyItemAtPath:src toPath:dst error:&err]
10 [self fixPermissionsOfURL:dst recursively:1]
```

```
-[FBAPrivilegedDaemon canModifyPath:] "/tmp/$PATHtoEXE"
11 if ([dst hasPrefix:@"/var/folders/"] ||
12     [dst hasPrefix:@"/private/var/"] || [dst hasPrefix:@"/tmp/"]) {
13     return TRUE;
14 } else {
15     return [dst rangeOfString:@"Library/Caches/
16         com.apple.appleseed.FeedbackAssistant"] != 0;
```



Thank you & Questions?



饮水思源 · 爱国荣校