Finding 1-Day Vulnerabilities in Trusted Applications using Selective Symbolic Execution

Marcel Busch, Kalle Dirsch

2020-02-23

Friedrich-Alexander-University Erlangen-Nürnberg, Germany

BAR'20

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Motivation

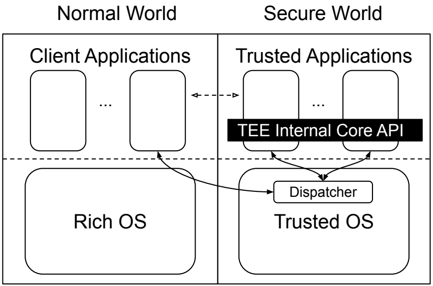
- How secure are Trusted Execution Environments (TEEs)?
- What errors do vendors make?
- In 2016 Huawei's TEE got exploited
  - CVE-2016-8764 [2]
  - Type confusion bug in the *Secure Storage* Trusted Application (TA)
- How to facilitate binary-diff-based analyses of 1-days in TAs?
  ⇒ Filter patches dealing with user input
  ⇒ Compare constraints introduced by patches

[a]

[a] https://www.youtube.com/watch?v=XjbGTZrg9DA

Background

- Two "Worlds"
- Two OSs
- Two user spaces
- Client Application (CA) logically interacts with TA
- Logical channel is carried out by Rich Operating System (Rich OS) and Trusted Operating System (Trusted OS)
- GlobalPlatform (GP) specification defines "*libc*" of TAs



Challenges and Related Work

Challenges

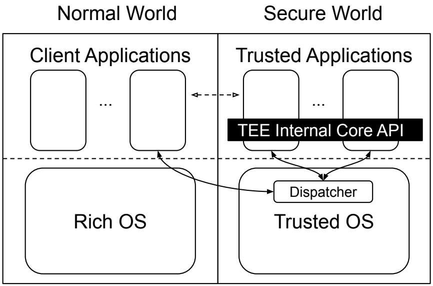
- TAs are closed source
- No dynamic analysis (*i.e.*, debugging)
- No TA modifications (*i.e.*, instrumentation)

Related Work

- PartEmu [1]
- TEEGris Usermode [4]

Our prototype, *SimTA*, focuses on

- GP Internal Core API



TA Lifecycle

- `TA_CreateEntryPoint`: Constructor
- `TA_OpenSessionEntryPoint`: Opens client session
- `TA_InvokeCommandEntryPoint`: Invocation of TA commands
- `TA_CloseSessionEntryPoint`: Closes client session
- `TA_DestroyEntryPoint`: Destructor

```
while ( 1 ) {
  LifecycleData* data = MsgRcv();

  switch ( data->lifecycle_cmd ) {
    case OPEN_SESS:
      if (data->init) {
        TA_CreateEntryPoint();
      }
      TA_OpenSessionEntryPoint(...);
      break;
    case INVOKE_CMD:
      TA_InvokeCommandEntryPoint(...);
      break;
    case CLOSE_SESS:
      TA_CloseSessionEntryPoint(...);
      if (data->deinit) {
        TA_DestroyEntryPoint();
      }
      break;
    default:
      break;
  }
  MsgSnd(data);
}
```

TA Parameters

```
TEE_Result TA_OpenSessionEntryPoint(
    uint32_t paramTypes,
    [inout] TEE_Param params[4],
    [out][ctx] void** sessionContext
);

TEE_Result TA_InvokeCommandEntryPoint(
    [ctx] void* sessionContext,
    uint32_t commandID,
    uint32_t paramTypes,
    [inout] TEE_Param params[4]
);
```

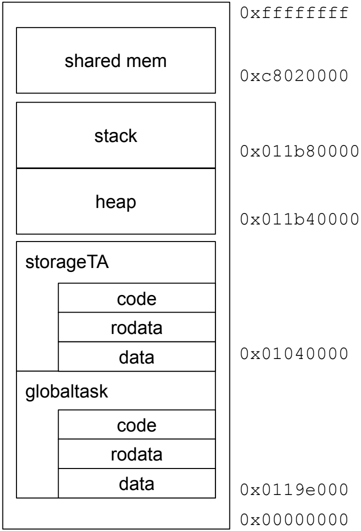
```
typedef union {
    struct {
        unsigned int buffer;
        unsigned int size;
    } memref;
    struct {
        unsigned int a;
        unsigned int b;
    } value;
} TEE_Param;
```

TA CmdId-Handler

```
TA_InvokeCommandEntryPoint(sessCtx, cmdId, paramTypes, params) {
  switch ( cmdId ) {
    case FOPEN:
      if (paramTypes != FOPEN_PTYPES)
        goto ptype_error;

      char* path; size_t pathsz;
      uint32_t flags;
      TEE_ObjectHandle obj;

      path = params[0]->memref.buffer;
      pathsz = params[0]->memref.size;
      flags = params[1]->value.a;

      TEE_OpenPersistentObject(TEE_STORAGE_PRIVATE, path, pathsz, flags, &obj);
      ...
      break;
    case FREAD:
    ...
  }
  return;
ptype_error:
  log("bad param types");
  return;
}
```

TA Address Space

- Address space retrieved via CVE-2016-8764 exploit
- `globaltask` implements GP Internal Core API
- `globaltask` is the only library
- TA does not perform syscalls
- `shared mem` contains `params`

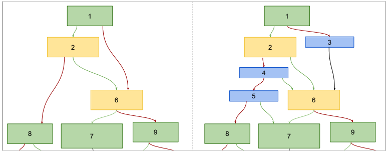


SimTA

- Maps memory according to our analysis using *angr* [3]
- Hooks input/output of lifecycle via angr-SimProcedures
  - Modular implementation of call sequences
  - Allows for selectively chosen symbolic inputs
- Hooks GP Internal Core API via angr-SimProcedures
  - Specification of functions available from GP
  - Implements all functions used by `storageTA`
- Can be found on GitHub: `https://github.com/teesec/simta`

Evaluation – Approach

- Analysis of Secure Storage TA
- `VNS-L21C432B130` vs `VNS-L21C432B160`
- Used Zynamic's BinDiff to identify patches
- SimTA provides
  - *filter mode* – identifies patches dealing with user-controlled input
  - *exec mode* – runs both versions with selectively chosen symbolic inputs
- Found three 1-days



Evaluation – CVE-2016-8764 Re-Discovery

- Type confusion

```
enum TEE_ParamType {
    TEE_PARAM_TYPE_NONE = 0x0,
    TEE_PARAM_TYPE_VALUE_INPUT = 0x1,
    TEE_PARAM_TYPE_VALUE_OUTPUT = 0x2,
    TEE_PARAM_TYPE_VALUE_INOUT = 0x3,
    TEE_PARAM_TYPE_MEMREF_INPUT = 0x5,
    TEE_PARAM_TYPE_MEMREF_OUTPUT = 0x6,
    TEE_PARAM_TYPE_MEMREF_INOUT = 0x7,
};
```

```
TA_InvokeCommandEntryPoint(sessCtx, cmdId,
    paramTypes, params) {
  switch ( cmdId ) {
    case FOPEN:
      ...
      break;
    case FREAD:
      // if (paramTypes != FOPEN_PTYPES)
      //  goto ptype_error;
      char *dst = params[0]->buffer;
      int sz = params[0]->size;
      ...
      TEE_ReadObjectData(obj, dst, sz);
      break;
      ...
    ...
  }
  return;
ptype_error:
  log("bad param types");
  return;
}
```

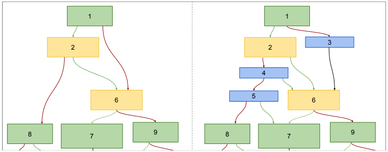
Evaluation – Heap-based buffer overflow

- Missing length check
- Passing attacker provided buffer length to `MemMove` operation

```
TA_InvokeCommandEntryPoint(sessCtx, cmdId,
    paramTypes, params) {
  switch ( cmdId ) {
    case FOPEN:
      ...
      char* path;
      param0_buf = params[0]->memref.buffer;
      param0_sz = params[0]->memref.size;

      // if(strlen(param0_buf) != param0_sz)
      //   return -1

      path = malloc(strlen(param0_buf));

      ...

      MemMove(path, param0_buf, param0_sz);
      ...
      break;
    case FREAD:
      ...
    ...
  }
  return;
}
```

Future Work and Limitations

- Support more Trusted Core (TC) TAs
- Larger analysis covering different versions and more TC TAs
- Investigate compatibility with other TEEs



THE END

Questions?

References

Lee Harrison, Hayawardh Vijayakumar, Rohan Padhye, Koushik Sen, and Michael Grace.
Partemu: Enabling dynamic analysis of real-world trustzone software using emulation.
In *Proceedings of the 29th USENIX Security Symposium (USENIX Security 2020) (To Appear)*, August 2020.

NIST.
Cve-2016-8764.
https://nvd.nist.gov/vuln/detail/CVE-2016-8764, 2017.
Accessed: 2019-08-28.

Yan Shoshitaishvili, Ruoyu Wang, Christopher Salls, Nick Stephens, Mario Polino, Audrey Dutcher, John Grosen, Siji Feng, Christophe Hauser, Christopher Kruegel, and Giovanni Vigna.
SoK: (State of) The Art of War: Offensive Techniques in Binary Analysis.
In *IEEE Symposium on Security and Privacy*, 2016.

Alexander Tarasikov.
Qemu teegris usermode.
https://github.com/astarasikov/qemu/tree/teegris_usermode, 2019.
Accessed: 2019-11-30.