AlphaEXP: An Expert System for Identifying Security-Sensitive Kernel Objects

Ruipeng Wang, Kaixiang Chen, Chao Zhang, Zulie Pan
Qianyu Li, Siliang Qin, Shenglin Xu, Min Zhang, Yang Li
Motivation

Memory corruption vulnerabilities are one of the major threats to software.

Source: Google Project Zero, 0day "In the Wild" spreadsheet. Last updated: 2023-04-20
Motivation

There are three types of solutions proposed and deployed in practice:

1. Vulnerability patching
   - It cannot mitigate unknown 0-day vulnerabilities

2. Software and system hardening
   - Such solutions would introduce performance costs to the system

3. Object-specific protections
   - Object-specific protection has a good balance between security and performance.
Motivation

There are three types of solutions proposed and deployed in practice:

1. Vulnerability patching
   - It cannot mitigate unknown 0-day vulnerabilities
   
   **How to identify sensitive objects that need to be protected?**

2. Software and system hardening
   - Such solutions would introduce performance costs to the system

3. Object-specific protections
   - Object-specific protection has a good balance between security and performance.
Motivation

How to identify sensitive objects that need to be protected?

◆ Analyzing publicly exposed exploits to find out objects that are abused
  this solution heavily relies on the human experience, and cannot find sensitive data that have not been abused yet

◆ Classifying objects based on developers’ intentions and the program’s semantics
  Its results (i.e., sensitive objects) may deviate from the adversary’s

◆ Analyzing the target code following specific attack knowledge
  SLAKE （CCS ’19）ELOISE （CCS ’20）
  they are not generic solutions for identifying sensitive objects, and cannot distinguish the sensitivity of the objects.
Motivation

Buffer overflow

- Arbitrary Address Writing (AAW)
- Arbitrary Address Reading (AAR)
- RIP hijacking

Capability Upgrade

Vulnerability Trigger

Exit Capabilities

Capability Stitching

Exploitation Finish

AAW

AAR

RIP hijacking

GetShell privilege escalation

… …
Motivation

(a) Vulnerable Function

```c
static bool tipc_crypto_key_rcv(struct tipc_crypto *rx, struct tipc_msg *hdr)
{
    ...
    struct tipc_aead_key *skey = NULL;
    ...
    u16 size = msg_data_sz(hdr);
    u8 *data = msg_data(hdr);
    ...
    skey = kmalloc(size, GFP_ATOMIC);
    ...
    skey->keylen = ntohl((__be32 *)(data + TIPC_AEAD_ALG_NAME));
    memcpy(skey->alg_name, data, TIPC_AEAD_ALG_NAME);
    memcpy(skey->key, data + TIPC_AEAD_ALG_NAME + sizeof(__be32), skey->keylen);
    ...
}
```

(b) Vulnerability

Packet (from userland):
- Header size: X => X*4
- Message size: Y
- Content:  … dead beef dead beef dead beef dead beef fff

(c) Structure Definition

```c
struct tipc_aead_key {
    char alg_name[TIPC_AEAD_ALG_NAME];
    unsigned int keylen;
    char key[ ];
};
```

```c
struct msg_msg {
    struct list_head m_list;
    long m_type;
    size_t m_ts;
    struct msg_msgseg *next;
    void *security;
};
```

```
struct tty_struct {
    int magic;
    struct kref kref;
    struct device *dev;
    struct tty_driver *driver;
    const struct tty_operations *ops;
    int index;
    ...
};
```

(d) Exploitation

- Overread
- Addr. leaking
- Heap manipulation
- Fake function table
- RIP hijacking
Motivation

```
struct tty_struct {
    int magic;
    struct kref kref;
    struct device *dev;
    struct tty_driver *driver;
    const struct tty_operations *ops;
    int index;
    ...
};
```

```
tty_operations (function table)
```

```
tty_ioctl
Fake function table
```

```
tty_struct
key
ops
```

```
RIP hijacking
```

Weak Cap.  
Data  
Control  
Functionality  
Powerful Cap.
Motivation

- Attack Action
- Selector
- Inference
- Knowledge Graph
- Scheduler

Vulnerability Analysis

- Knowledge of Target Program
- Input Guess
- Extrapolating
- Result
- Practice

EXP

- Attack Action Selector
- Scheduler

Inference

Knowledge Graph

Scheduler

Attacker

Scheduling

Inferencing

Sensitive Object

Kernel Knowledge Base

Result
Our Solution: AlphaEXP

Knowledge Graph Construction
- Ontology Construction
- Knowledge Extraction
- Mechanism
- Capability Rules

Knowledge Graph

Attack Path Generation
- Inference Engine
  - Memory Inference
  - Capability Inference
- Rules
- Schemas
- Scheduler
- Attack Action Selector

PoCs
- Capabilities
- Attack Paths

Sensitive Object Classifying
- Applied Conditions Assessment
- Effect Achieved Assessment
- Classifying Results
Our Solution: AlphaEXP

Knowledge Graph Construction
Our Solution: AlphaEXP

Attack Path Generation

1. Choose an state.
2. Choose an attack action.
3. Inference attack action effect.
4. Adding the converged state to state pool.

✓ If the attack action is as expected, add it to the attack path.

Inference Rules based on Datalog
### Our Solution: AlphaEXP

**Sensitive Object Classifying**

<table>
<thead>
<tr>
<th>Perspective</th>
<th>Factor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Applied Conditions</strong></td>
<td>kmem-cache</td>
<td>Can be applied in the exploitation of different vulnerability object memory kmem-cache.</td>
</tr>
<tr>
<td></td>
<td>entry capability</td>
<td>Modification of sensitive object requires unintended writing capability over 0x80 size</td>
</tr>
<tr>
<td></td>
<td>vulnerability type</td>
<td>Sensitive object can both be applied in the exploitation of overflows and UAF</td>
</tr>
<tr>
<td><strong>Effect Achieved</strong></td>
<td>writing capability</td>
<td>Sensitive object can be used to upgrade writing capability in exploitation</td>
</tr>
<tr>
<td></td>
<td>executing capability</td>
<td>Sensitive object can be used to upgrade executing capability in exploitation</td>
</tr>
<tr>
<td></td>
<td>reading capability</td>
<td>Sensitive object can be used to upgrade reading capability in exploitation</td>
</tr>
</tbody>
</table>
Evaluation

• RQ1: How effective is AlphaEXP in sensitive objects identifying and classifying?

• RQ2: Is AlphaEXP better at identifying sensitive objects compared to current SOTA techniques?

• RQ3: What is the cost of building a knowledge graph?

• RQ4: How effective is attack path generation?
Evaluation

• RQ1: How effective is AlphaEXP in identifying and classifying sensitive objects?
• RQ2: Is AlphaEXP better at identifying sensitive objects compared to current SOTA techniques?

### Sensitive Objects

<table>
<thead>
<tr>
<th>Write</th>
<th>Read</th>
</tr>
</thead>
</table>
| keyctl_update_key\*, msg_msg\*, add_key\*, ip_options\* get_from_user\*, scsi_request, hiddev_ioctl_usage, proc_ioctl, ksecx_segment, do_ipv6_setsockopt | ipv6_opt_hdr\*, sock_fprog_kern\*, policy_load_memory\*, lidt_struct\*, ip_options\*, seq_file\*, xhrm_policy\*, xhrm_algo_aced\*, xhrm_algo\* cfg80211_pkt_pattern\*, user_key_payload\*, xhrm_reply_state_exn\*, ip_sf_socklist\*, proc_dir_entry\*, ext4_dir_entry\*; station_info\* cache_reader\*, tc_cookie\*, cfg80211_bss_ies\*, sg_header\*, intimid_event_info\*, audit_rule_data\*, fb_info\*, cfg80211_sched_scan_request\*, fc_map_user\*, cache_request\*, frame\*, ieee80211_mgd_auth_data\*, mon_reader_bin\*, mon_reader_text\*, cfg80211_scan_request\* tcp_fastopen_context\*, request_key_auth\*, xhrm_algo_auth\*, cfg80211_wowlan_tcp\*, msg_msg\*, tcp_sock\*, user_element\*, neighbour pnegh_entry, net_device, netdev_phys_item_id, netlink_ext_ack, cfg80211_nan_match_params, wiphy, wiphy_iftype_ext_cap, wireless_dev hidraw_report, hid_device, sg_request, fb_map, usb_device, usb, usbpl, drm_crct, drm_plane, cfg80211_connect_res_params, kobj_uvent_env beacon_data, probe_resp, cfg80211 roam_info, cfg80211_wowlan_wakeup, cfg80211_ssid, cfg80211_mgmt_tx_params, ieee80211_mgd_assoc_data | 50 objects that could be abused to get writing capability, 81 objects with reading capability, and 112 objects with execution capability

---

\* Identified by SLAKE as well, \#: Identified by ELOISE as well, \#: Identified by KOBOE as well, \#: Not present in v4.15, \#: False Positives
**Evaluation**

- **RQ1:** How effective is AlphaEXP in sensitive objects identifying and classifying?

---

**Legend**

- **All Data**
  - WX
  - XR
  - WR
  - X
  - W
  - R

- **Cap&Cap&T-WR**
- **Cap&T-X**
- **Cap&T-R**
- **T-WX**
- **T-X**
- **Cap&W**
- **Cap&T-W**
- **T-W**
- **T-R**

---

**Cache&Cap&T-WR**

- sg_scsi_ioctl
- keyctl_update_key
- sendmsg
- create_entry
- add_key
- setxattr
- fb_write
- proc_bulk
- memfd_create

- drm_property_blob
- msg_msg
Evaluation

- RQ3: What is the cost of building a knowledge graph?

The static knowledge extraction process takes 19 minutes.

The dynamic knowledge extraction process takes 72 hours.

100,723 entities and 180,204 relationships
Evaluation

• RQ4: How effective is attack path generation?
Conclusion

◆ AlphaEXP can identify sensitive kernel objects and classify their sensitivity, able to help defenders build cost-effective defenses.

◆ AlphaEXP constructs a knowledge graph of the kernel.

◆ AlphaEXP reports several hundreds of sensitive kernel objects and classifies them into 12 sensitivity levels.

◆ AlphaEXP provides a new idea for automated exploit generation (AEG).
Thanks for listening!

Q&A

Contact: Ruipeng Wang, wangruipeng@nudt.edu.cn