Piston: Uncooperative Remote Runtime Patching

Chris Salls, Yan Shoshitaishvili, Nick Stephens, Christopher Kruegel, and Giovanni Vigna
Motivation: Software Bugs

Internet-paralyzing Mirai botnet comes roaring back with new strain

100,000 devices infected in 60 hours by strain that targeted ZyXEL devices.

DAN GOODIN - 11/29/2017, 9:21 AM

Hackers found 47 new vulnerabilities in 23 IoT devices at DEF CON

The results from this year's IoT hacking contest are in and it's not a pretty picture
Motivation: Need Automated Patching

Fixing, upgrading and patching IoT devices can be a real nightmare

The recall of almost half a million St. Jude Medical pacemakers highlights the growing importance—and huge risks—of the Internet of Things.

FTC sets $25,000 prize for automatic IoT patching

Feds cite use of internet-connected cameras to launch botnet attack as proof that better security is needed
Hotpatching

- Needs builtin support.

Service@Wansview.com
Sun 10/29, 6:19 PM

Hi Aravind,

Really sorry for your inconvenience, the K2 camera has no new firmware. And it also can't upgrade the firmware. I have no the K2 camera's initial firmware.

Best regards
Jerry

Service@Wansview.com
Piston

- Critical bugs are often security vulnerabilities that can lead to code execution.

- Can we use the bug to patch the device?
Piston: Overview

Patch Generation

- Function Matching
- Replacement Function Placement

Patch set

Remote Patching

Repair Routine

Exploit Effect Reconstruction
Hijacked Function Analysis
Caller State Recovery

Rollback Routine

Repair Planning

Optional Input from Analyst

Exploit Specification
Remote Configuration

Patching Stub
Piston: Patch Generation

Function Matching

Replacement Function Placement

Original and Replacement binaries

Patch set
Patch Generation: Function Matching

- Identify functions to be updated.
- Filter out superficial differences.

Old Function

```
push ebp
mov ebp,esp
sub esp, 0x18
mov eax, 0x804a02c
: 0x804a02c "Hello %s"
```

New Function

```
push ebp
mov ebp,esp
sub esp, 0x18
mov eax, 0x805a084
: 0x805a084 "Hello %s"
```
Piston: Patch Generation

- Original and Replacement binaries
- Function Matching
- Replacement Function Placement

Patch set
Patch Generation: Replacement Function Placement

- Identify location for the new functions.
- Fix-up relative references.
- Create Jump-out stubs in the old functions.
Patch Generation: Replacement Function Placement

- Jump-out Stub:

```plaintext
; Old Function
oldloc:
jmp newloc
...
...

; Patched Function
newloc:
push ebp
mov ebp, esp
...
...
ret
```
Piston: Patch Generation

Patch Generation

- Function Matching
- Replacement Function Placement

Remote Patching

- Repair Routine
  - Exploit Effect Reconstruction
  - Hijacked Function Analysis
  - Caller State Recovery

Rollback Routine

Patching Stub

Optional Input from Analyst

Exploit Specification

Remote Configuration
Piston: Repair Planning

Original and Replacement binaries

Repair Routine

Rollback Routine

Exploit Effect Reconstruction

Hijacked Function Analysis

Caller State Recovery

Repair Planning

Exploit Specification

Remote Configuration
Repair Planning: Exploit Effect Reconstruction

- Trace execution of exploit:
  - Hijacked Function and Caller Function.

- Detect Exploitation Point:
  - Use simple heuristics to detect the instruction where the buffer overflow occurs.

- Mark all overflowed data as corrupted or Tainted.
Piston: Repair Planning

- Original and Replacement binaries
- Repair Routine
- Rollback Routine
- Exploit Effect Reconstruction
- Hijacked Function Analysis
- Caller State Recovery
- Exploit Specification
- Remote Configuration

Repair Planning
Repair Planning: Hijacked Function Analysis

- Does Hijacked Function needs to be restarted?
  - Tainted data influence control or data flow?

- Recover local and global state.

- Repeatable system calls.
Repair Planning: Hijacked Function Analysis

- **Recovering global state:**
  - Recover the data read from global state using non-corrupted data.
  - Use Under-Constrained symbolic execution (UCSE) to construct the symbolic expressions.

- **Rollback Routine.**
Piston: Repair Planning

- Original and Replacement binaries
- Repair Routine
- Rollback Routine
- Exploit Effect Reconstruction
- Hijacked Function Analysis
- Caller State Recovery

Exploit Specification
Remote Configuration
Repair Planning: Caller State Recovery

- Recover Local state of Caller Function:
  - Live callee-saved Registers.
  - Hijacked function parameters.
Repair Planning: Caller State Recovery

- Recover from redundant stack data:

  ```
  mov eax, [ebp + var_14]
  mov edx, [ebp + var_8]
  sub eax, edx
  mov [ebp + var_3C], eax
  call hijacked_func()
  ```

  If \( \text{var}_3C \) is corrupted it can be recovered as \( \text{var}_14 - \text{var}_8 \)

- **Repair Routine.**
Piston: Repair Planning

Patch Generation
- Function Matching
- Replacement Function Placement

Patch set

Remote Patching

Repair Planning
- Exploit Effect Reconstruction
- Hijacked Function Analysis
- Caller State Recovery

Rollback Routine

Patching Stub

Optional Input from Analyst

Exploit Specification

Remote Configuration
Piston: Remote Patching

Patch Generation
- Function Matching
- Replacement Function Placement

Remote Patching
- Patch set
- Repair Routine
- Rollback Routine

Repair Planning
- Exploit Effect Reconstruction
- Hijacked Function Analysis
- Caller State Recovery

Patching Stub

Original and Replacement binaries

Optional Input from Analyst

Exploit Specification

Remote Configuration
Piston: Remote Patching

- Patching Stub:
  - Launch Exploit: Gain Control.
  - Repair Routine.
  - Rollback Routine.
  - Apply Patch Set.
Evaluation: Dataset

- Cyber Grand Challenge (CGC) binaries:
  - Stack-based buffer overflow: 24 binaries

- NGINX 1.4.0:
  - CVE-2013-2028
Evaluation: Recovery

- 2 Exploit types:
  - Shellcode stub: 23 bytes.
    - Successfully recovered for **22/24 (91%)** Binaries.
  - ROP stub: Handles NX stack: 40 bytes.
    - Successfully recovered for **20/24 (83%)** Binaries
Evaluation: End-End

- Recover, Patch and Restart.
- 5 CGC binaries.
- NGINX 1.4.0
## Evaluation: End-End

<table>
<thead>
<tr>
<th>Binary Name</th>
<th>Function Interrupted</th>
<th>Automated Rollback?</th>
<th>Automated Repair?</th>
<th>Caller stack recovered (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CROMU_00017</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>144</td>
</tr>
<tr>
<td>CROMU_00020</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>52</td>
</tr>
<tr>
<td>CROMU_00037</td>
<td>NO</td>
<td>N/A</td>
<td>YES</td>
<td>4</td>
</tr>
<tr>
<td>CROMU_00038</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>4</td>
</tr>
<tr>
<td>CROMU_00039</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>303</td>
</tr>
<tr>
<td>NGINX</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
<td>28</td>
</tr>
</tbody>
</table>
Limitations

- Demonstrated for only stack-buffer overflows.
- Recovering from other type of exploits need analyst input.
- Data recovery needs redundancy.
Conclusions

- Automated Patching Mechanism for Uncooperative Processes.
- Automated Rollback and Recovery.
- Empirical Evaluation.
BACKUP
Piston: Overview

- Patch Generation
  - Function Matching
  - Replacement Function Placement

- Repair Planning
  - Repair Routine
    - Exploit Effect Reconstruction
    - Hijacked Function Analysis
    - Caller State Recovery

- Remote Patching
  - Remote Configuration
  - Exploit Specification
  - Optional Input from Analyst

- Patch set
  - Original and Replacement binaries

- Patching Stub