# Piston: Uncooperative Remote Runtime Patching

Chris Salls, Yan Shoshitaishvili, Nick Stephens, Christopher Kruegel, and Giovanni Vigna

seclab
THE COMPUTER SECURITY GROUP AT UC SANTA BARBARA

# Motivation: Software Bugs

Internet-paralyzing Mirai botnet comes roaring back with new strain

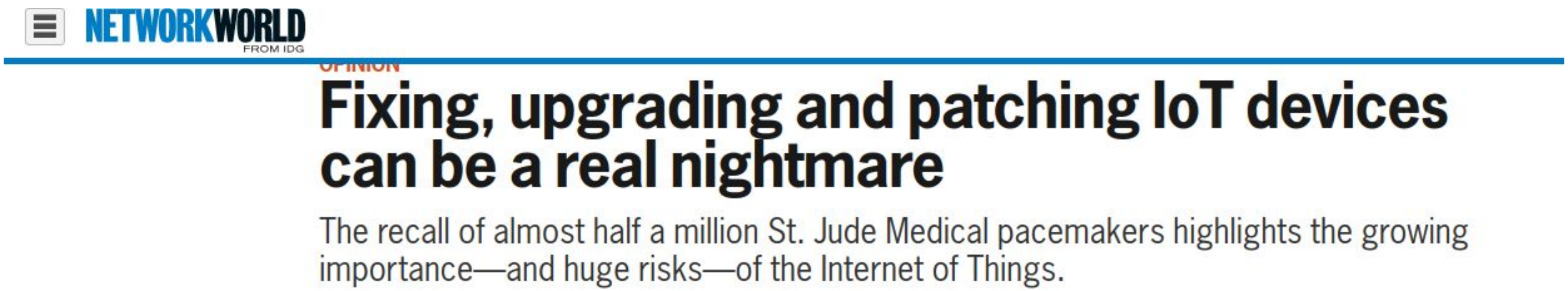
100,000 devices infected in 60 hours by strain that targeted ZyXEL devices.

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

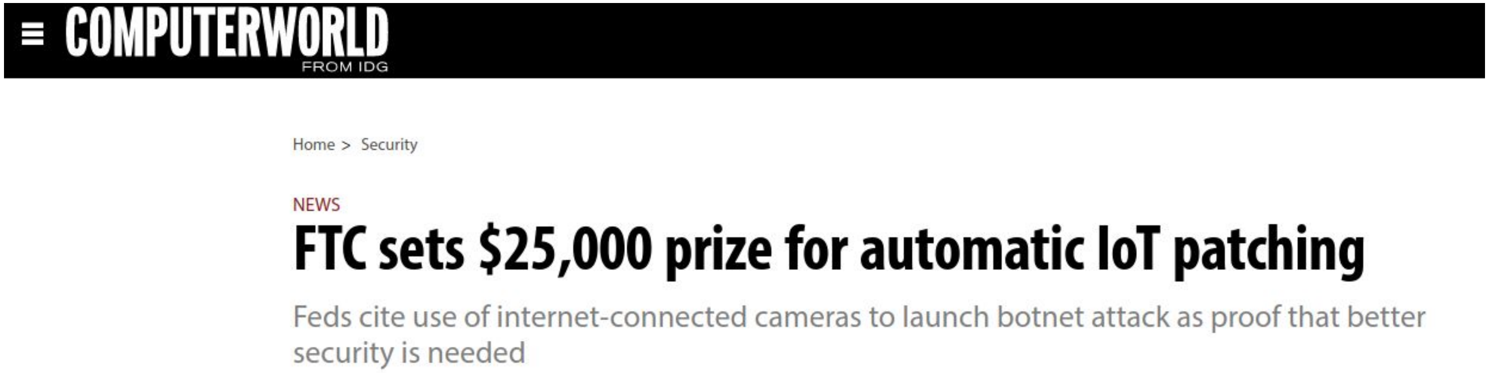


NETWORKWORLD FROM IDG

OPINION

## 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.



COMPUTERWORLD FROM IDG

Home > Security

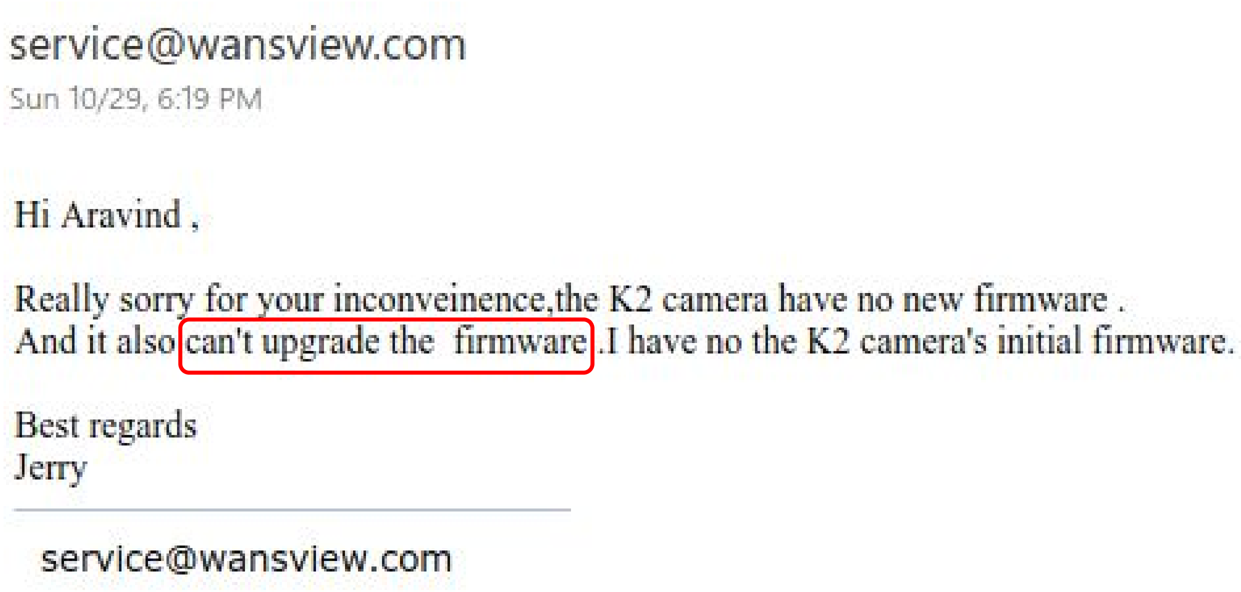
NEWS

## 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 inconveinence,the K2 camera have no new firmware .
And it also can't upgrade the firmware.I have no the K2 camera's initial firmware.

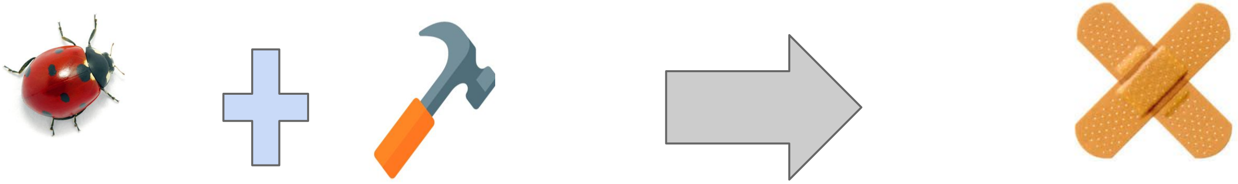
Best regards
Jerry

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?

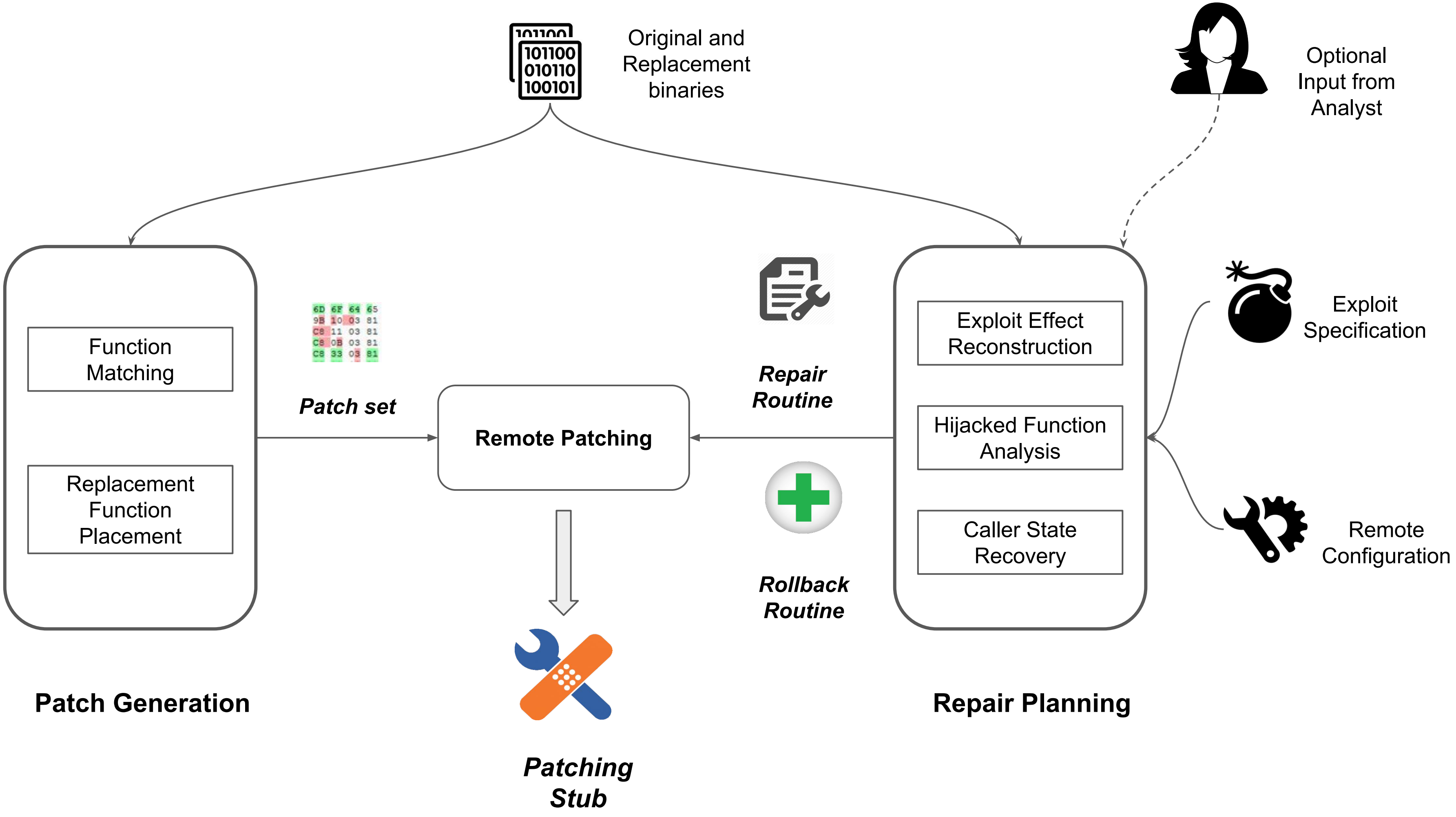


Bug

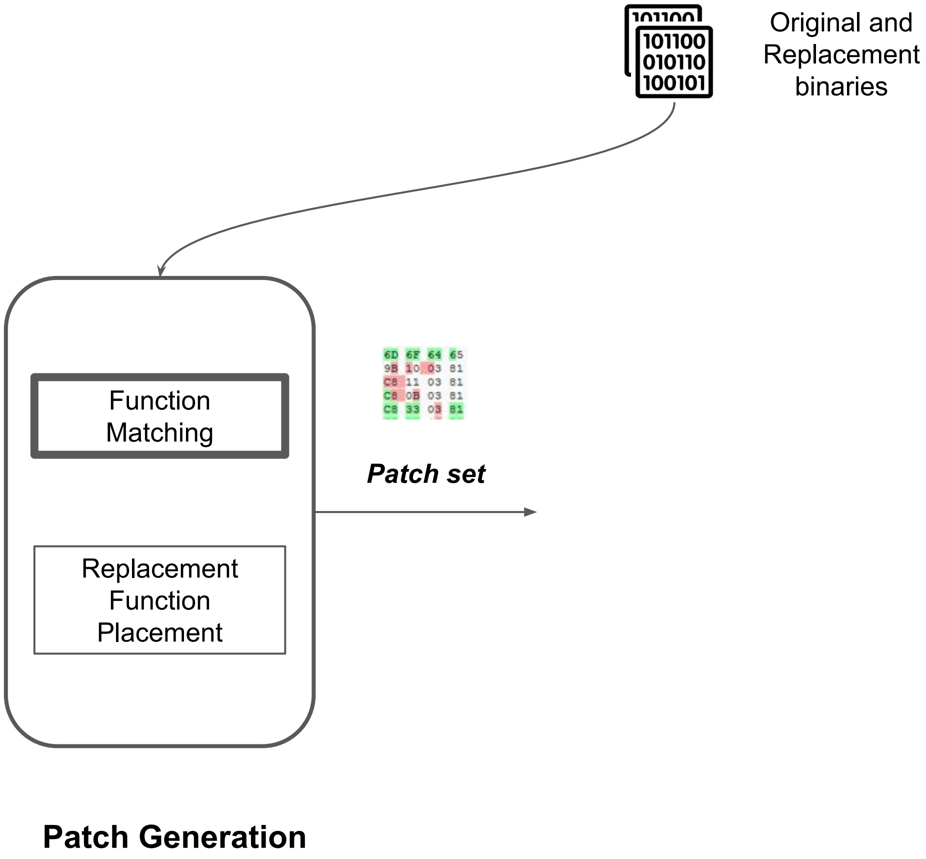
Exploit

Patch

# Piston: Overview



# Piston: Patch Generation



# Patch Generation: Function Matching

- Identify functions to be updated.

- Filter out superficial differences.

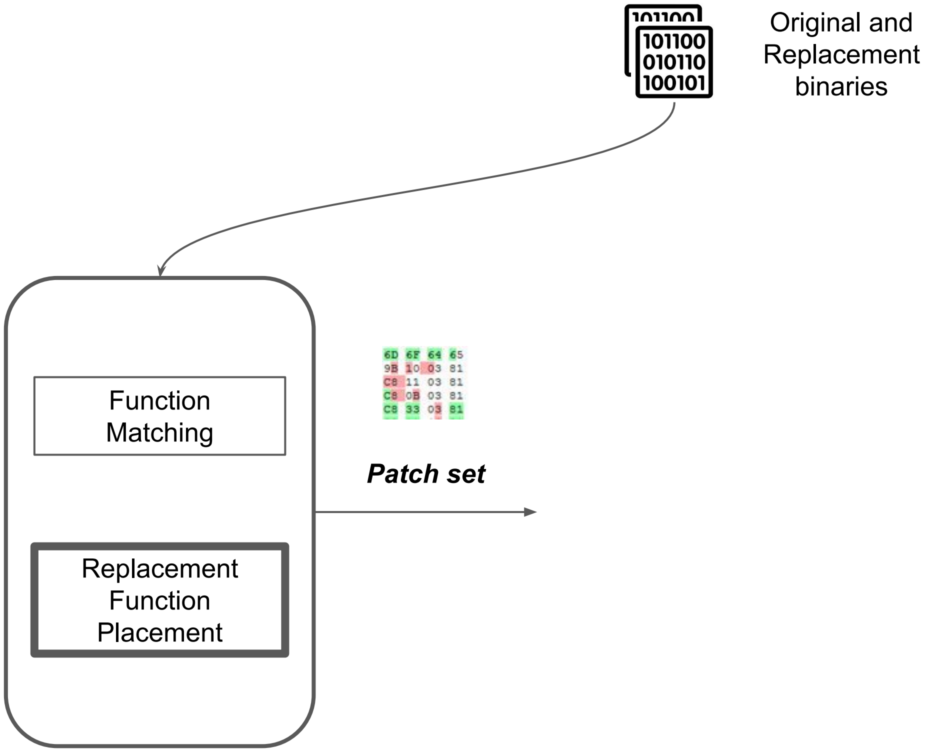
```
push ebp
mov ebp,esp
sub esp, 0x18
mov eax, 0x804a02c
:
0x804a02c “Hello %s”
```

Old Function

```
push ebp
mov ebp,esp
sub esp, 0x18
mov eax, 0x805a084
:
0x805a084 “Hello %s”
```

New Function

# Piston: Patch Generation



Original and Replacement binaries

Function Matching

Replacement Function Placement

```
6D 6F 64 65
9B 10 03 81
C8 11 03 81
C8 0B 03 81
C8 33 03 81
```

***Patch set***

**Patch Generation**

# Patch Generation: Replacement Function Placement

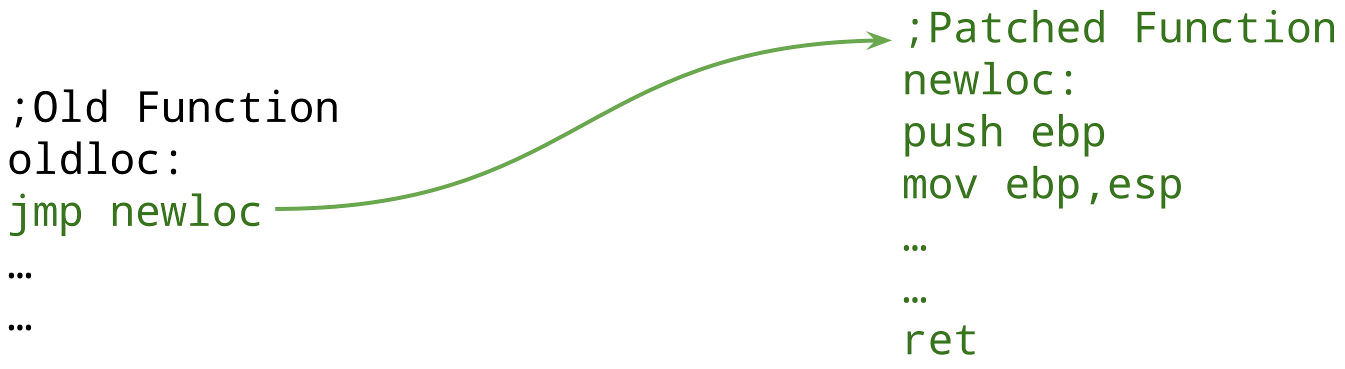
- Identify location for the new functions.

- Fix-up relative references.

- Create Jump-out stubs in the old functions.

# Patch Generation: Replacement Function Placement

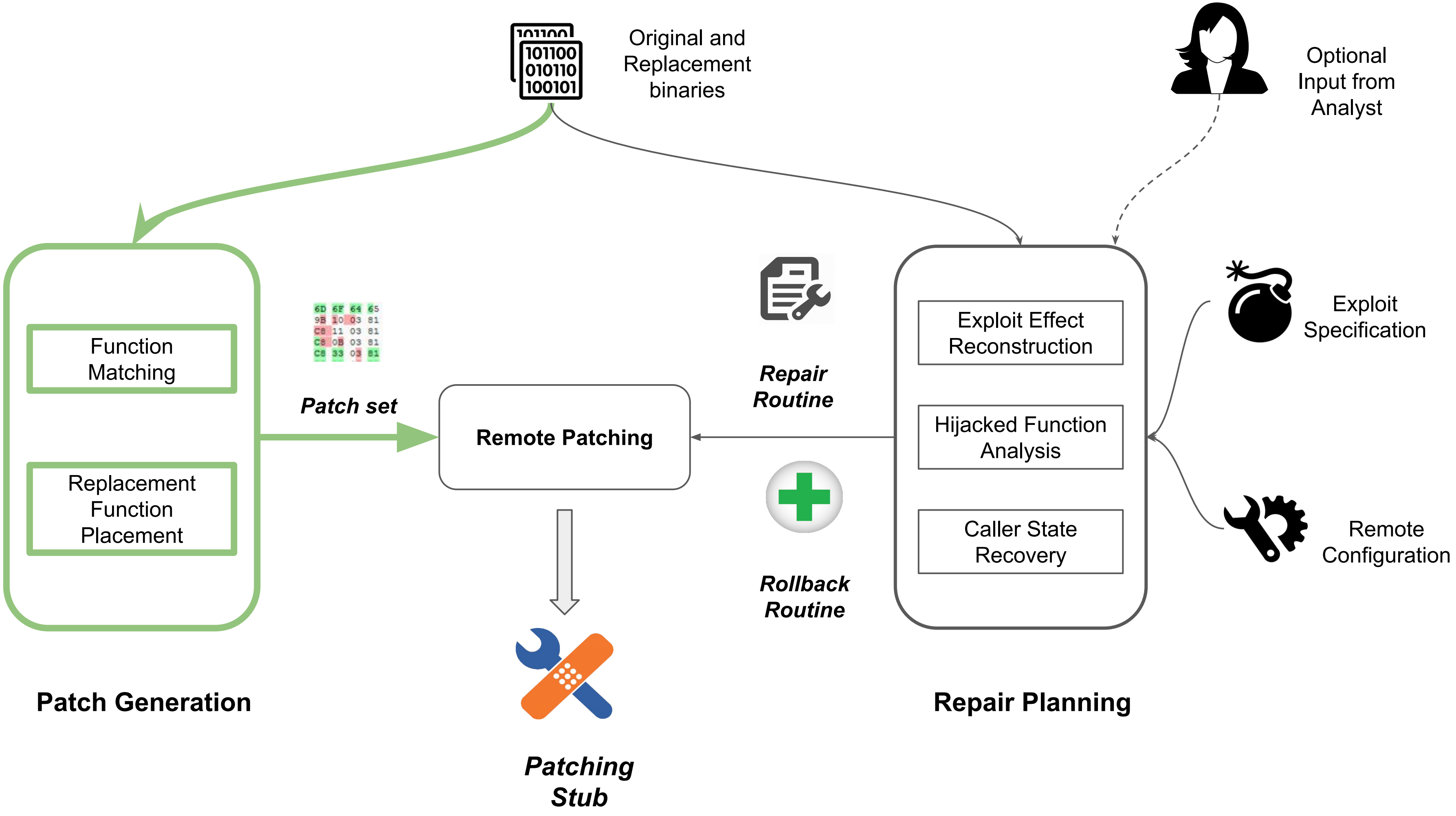
- Jump-out Stub:



```
;Old Function
oldloc:
jmp newloc
…
…
```

```
;Patched Function
newloc:
push ebp
mov ebp,esp
…
…
ret
```

# Piston: Patch Generation



Original and Replacement binaries

Optional Input from Analyst

Function Matching

Replacement Function Placement

Patch Generation

*Patch set*

Remote Patching

*Patching Stub*

*Repair Routine*

*Rollback Routine*

Exploit Effect Reconstruction

Hijacked Function Analysis

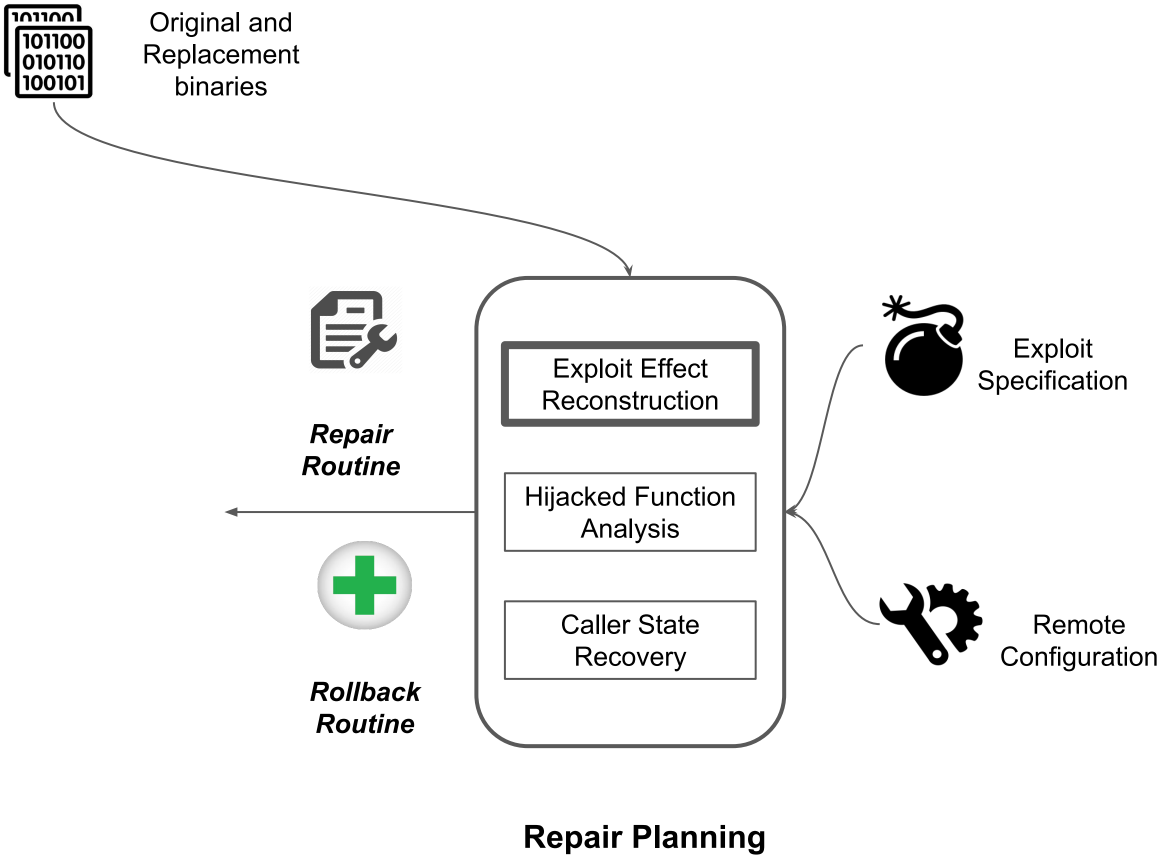
Caller State Recovery

Repair Planning

Exploit Specification

Remote Configuration

# Piston: Repair Planning



Original and Replacement binaries

Exploit Effect Reconstruction

Hijacked Function Analysis

Caller State Recovery

Exploit Specification

Remote Configuration

***Repair Routine***

***Rollback Routine***

**Repair Planning**

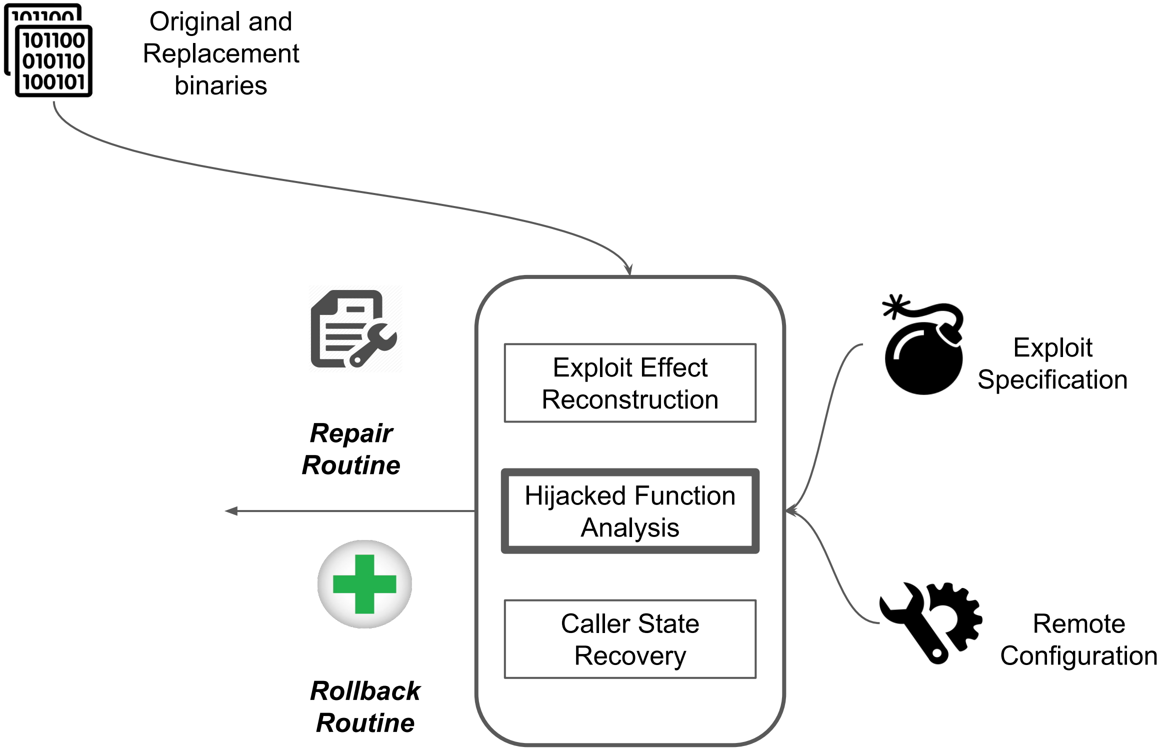
# Repair Planning: Exploit Effect Reconstruction

- 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



**Repair Planning**

# Repair Planning: Hijacked Function Analysis

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

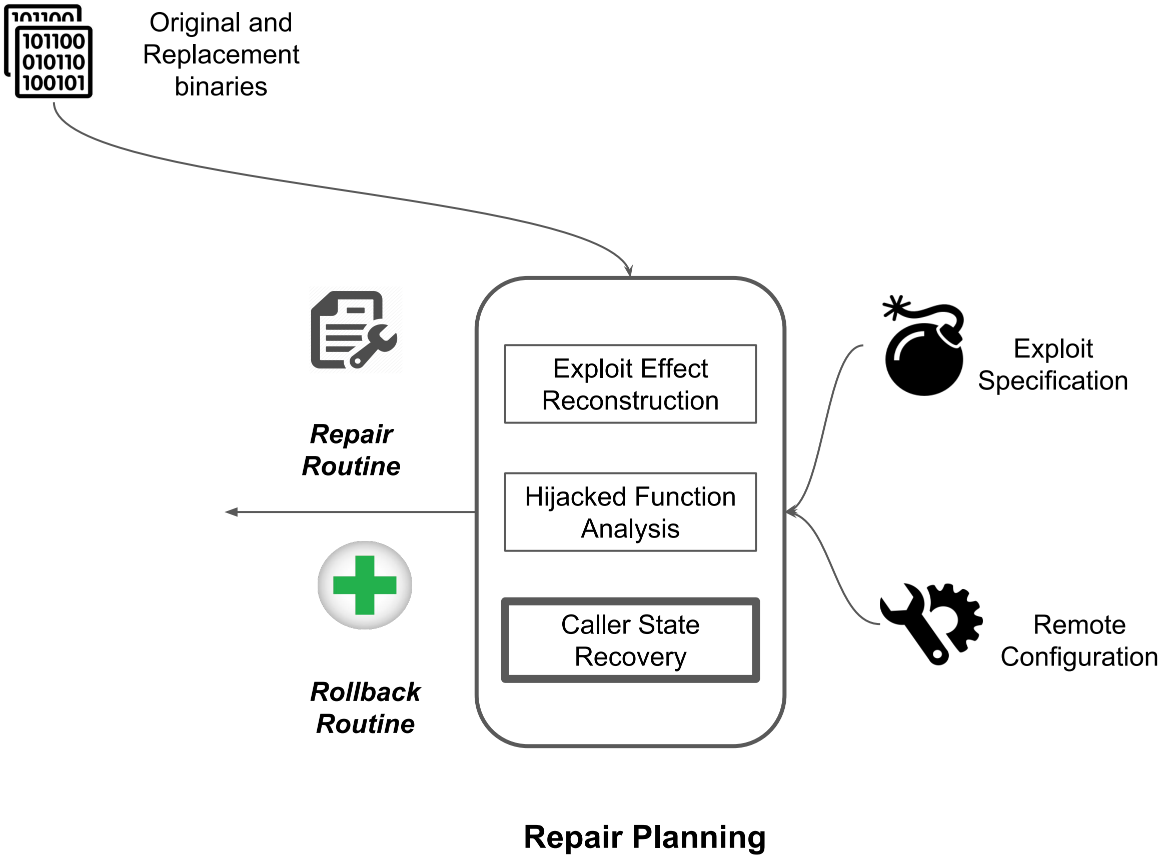
- Recover local and global state.

- 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



# Repair Planning: Caller State Recovery

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

# Repair Planning: Caller State Recovery

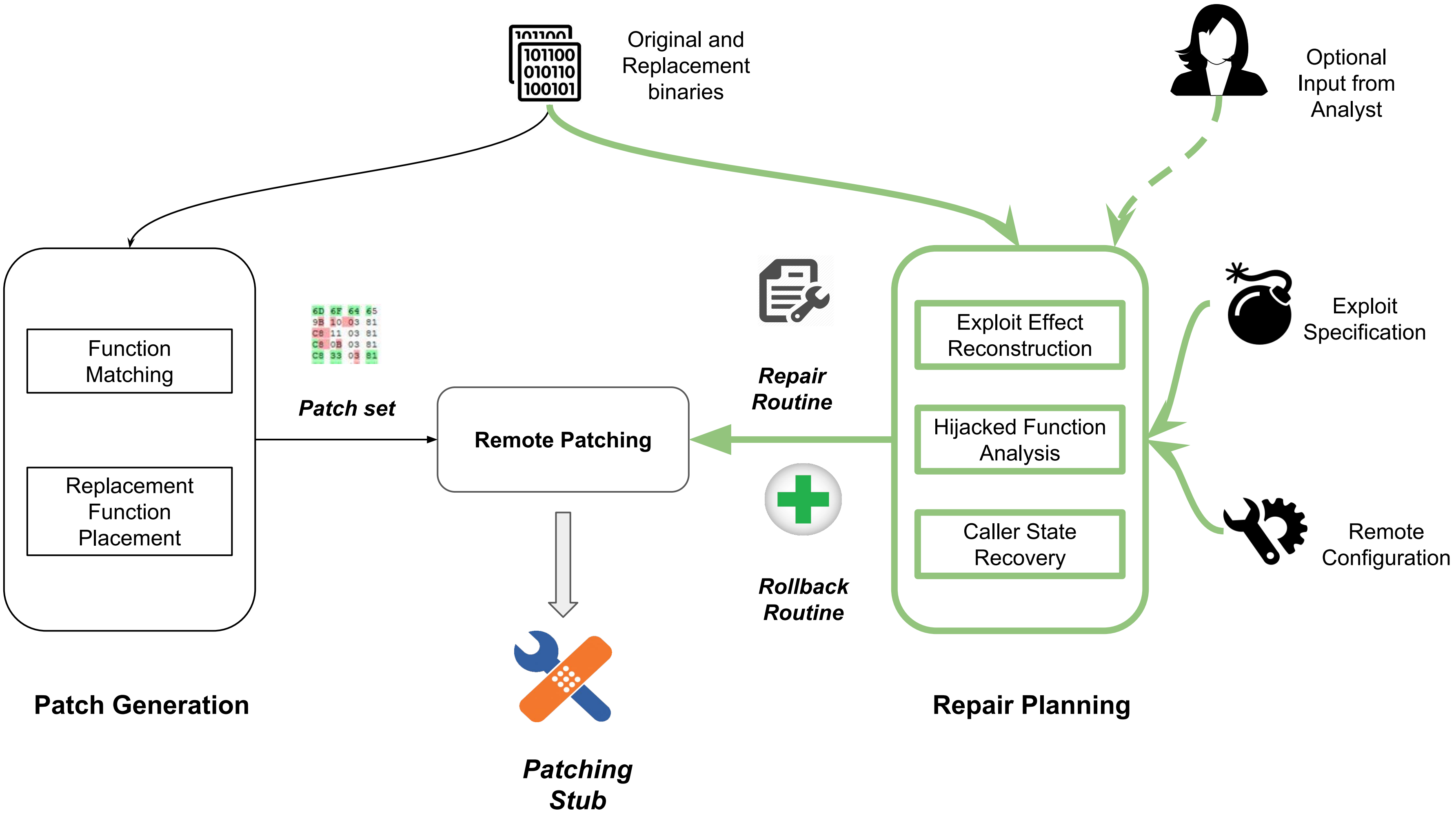
- Recover from redundant stack data:

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

If var_3C is corrupted it can be recovered as var_14 - var_8

- ***Repair Routine.***

# Piston: Repair Planning



Original and Replacement binaries

Optional Input from Analyst

Function Matching

Replacement Function Placement

Patch Generation

*Patch set*

**Remote Patching**

***Patching Stub***

***Repair Routine***

***Rollback Routine***

Exploit Effect Reconstruction

Hijacked Function Analysis

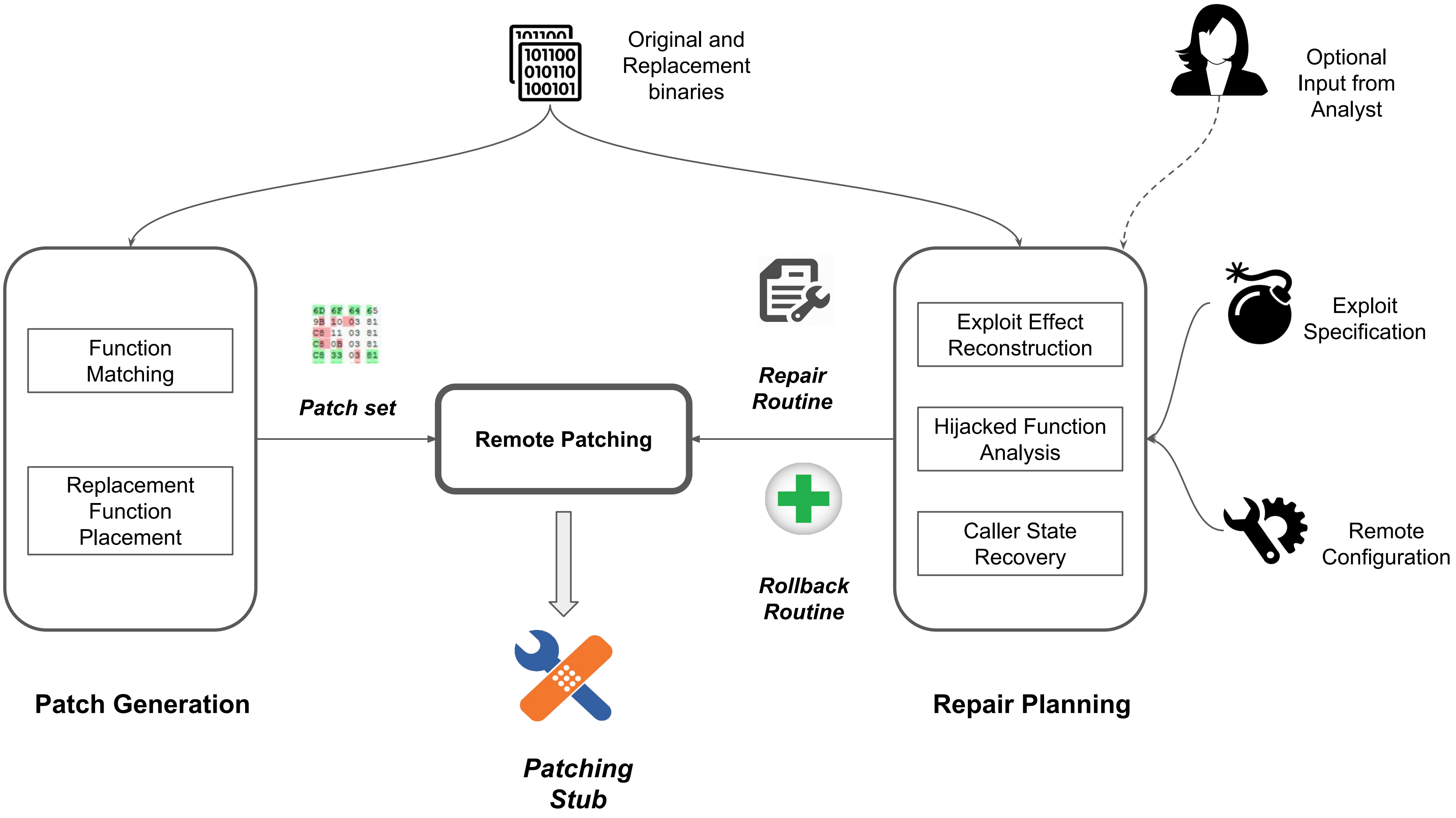
Caller State Recovery

**Repair Planning**

Exploit Specification

Remote Configuration

# Piston: Remote Patching



Original and Replacement binaries

Optional Input from Analyst

Function Matching

Replacement Function Placement

Patch Generation

*Patch set*

**Remote Patching**

*Patching Stub*

*Repair Routine*

*Rollback Routine*

Exploit Effect Reconstruction

Hijacked Function Analysis

Caller State Recovery

Repair Planning

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

| Binary Name | Function Interrupted | Automated Rollback? | Automated Repair? | Caller stack recovered (bytes) |
|---|---|---|---|---|
| CROMU_00017 | YES | YES | YES | 144 |
| CROMU_00020 | YES | YES | YES | 52 |
| CROMU_00037 | NO | N/A | YES | 4 |
| CROMU_00038 | YES | YES | NO 👩 | 4 |
| CROMU_00039 | YES | YES | YES | 303 |
| NGINX | YES | NO 👩 | YES | 28 |

# Limitations

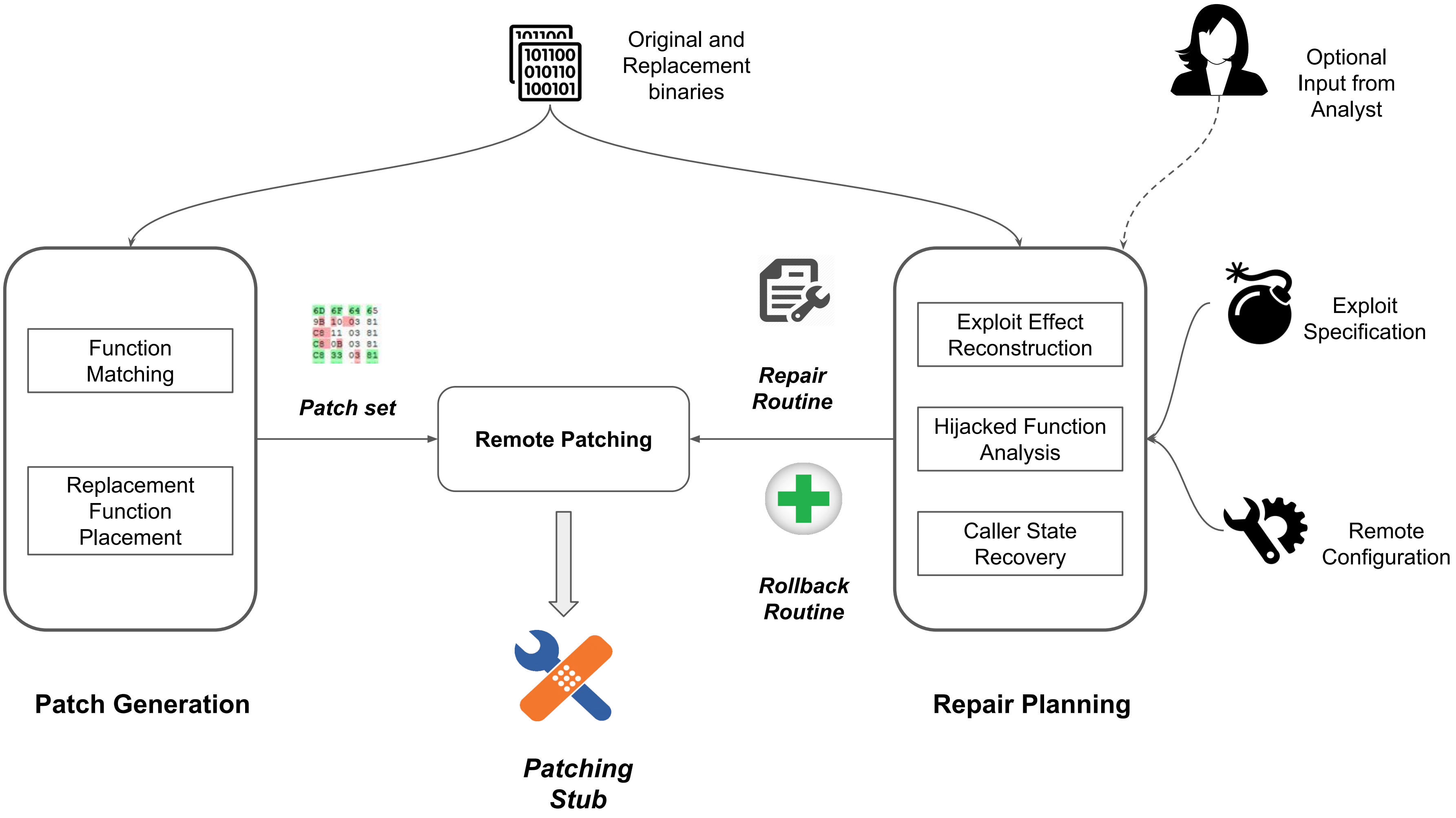
- Demonstrated for only stack-buffer overflows.
- Recovering from other type of exploits need analyst input.
- Data recovery needs redundancy.

# Conclusions

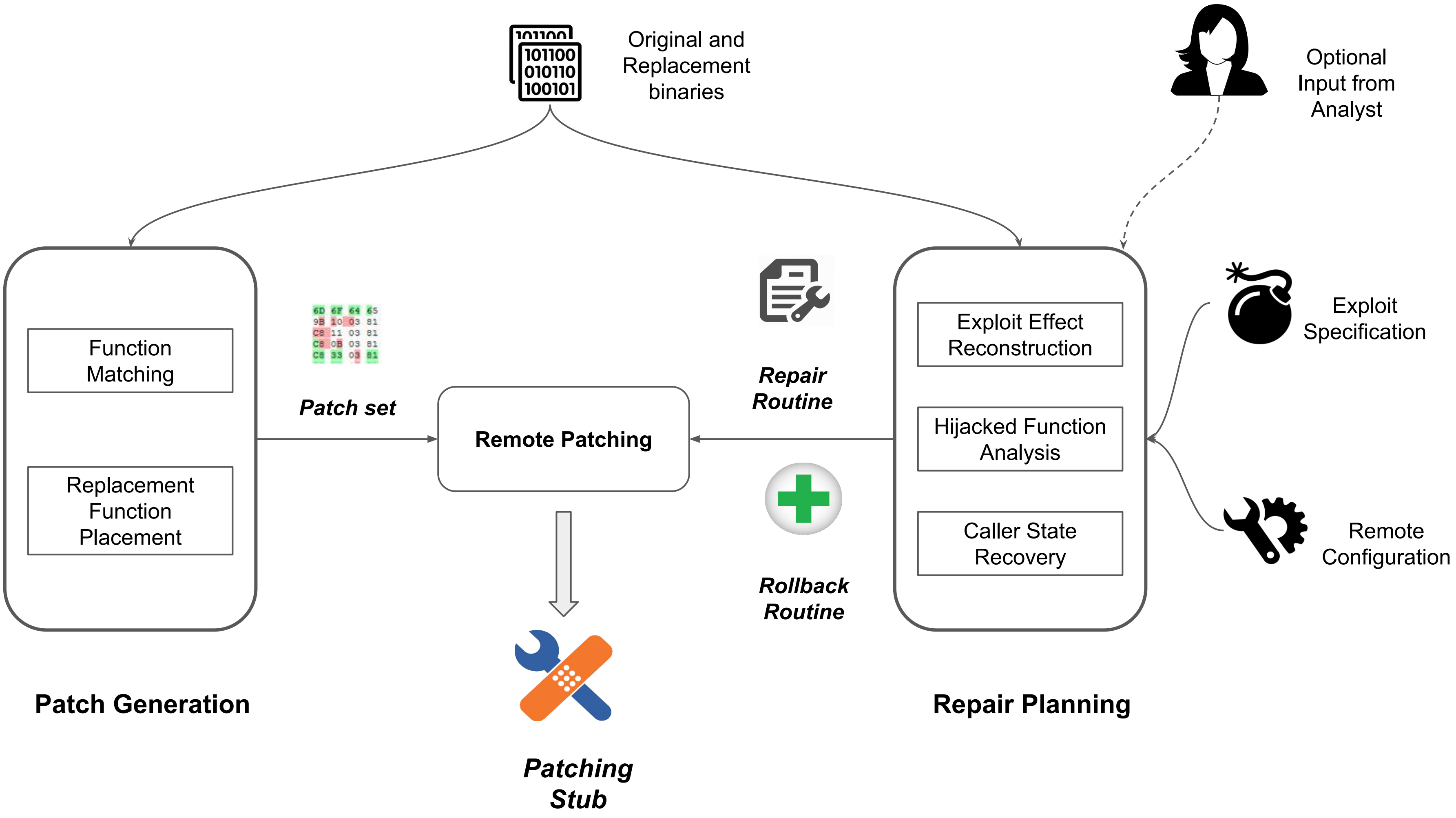
- Automated Patching Mechanism for Uncooperative Processes.
- Automated Rollback and Recovery.
- Empirical Evaluation.

# BACKUP

# Piston: Overview



# Piston: Overview



Original and Replacement binaries

Optional Input from Analyst

Function Matching

Replacement Function Placement

**Patch Generation**

***Patch set***

**Remote Patching**

***Patching Stub***

***Repair Routine***

***Rollback Routine***

Exploit Effect Reconstruction

Hijacked Function Analysis

Caller State Recovery

**Repair Planning**

Exploit Specification

Remote Configuration