ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

🥭**Operation Mango**: Scalable Discovery of Taint-Style Vulnerabilities in Binary Firmware Services

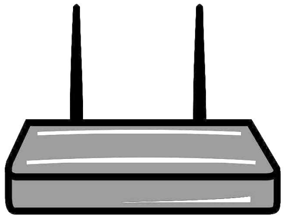
Wil Gibbs, Arvind S Raj, Jayakrishna Menon Vadayath, Hui Jun Tay, Justin Miller, Akshay Ajayan, Zion Leonahenahe Basque, Audrey Dutcher, Fangzhou Dong, Xavier Maso, Giovanni Vigna, Christopher Kruegel, Adam Doupé, Yan Shoshitaishvili, Ruoyu Wang

Arizona State University

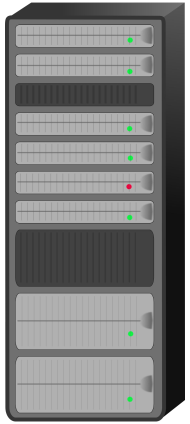
UCSB UNIVERSITY OF CALIFORNIA SANTA BARBARA

sefcom security engineering for future computing

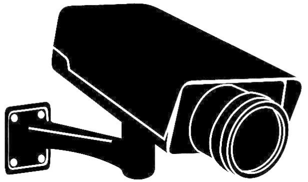
Firmware in Embedded Devices



Router

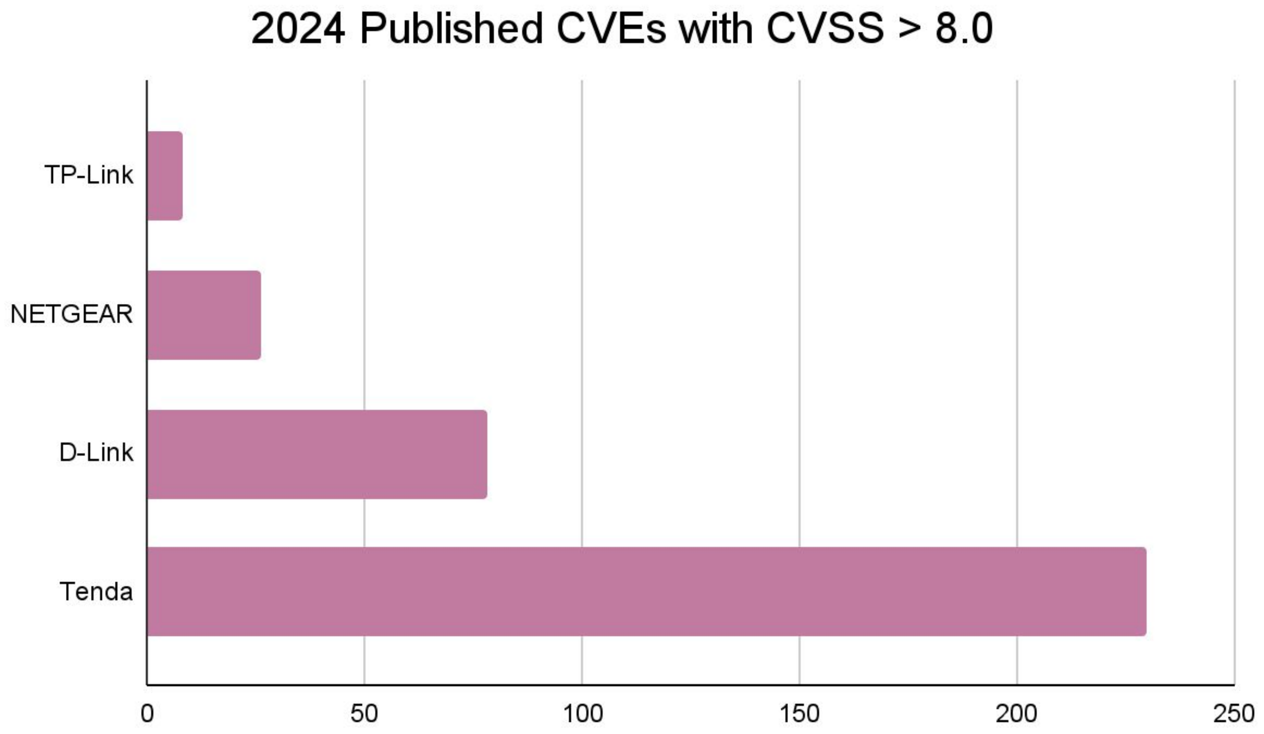


NAS

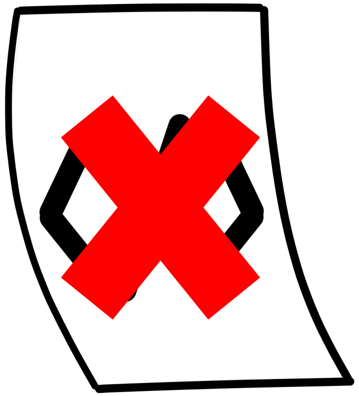


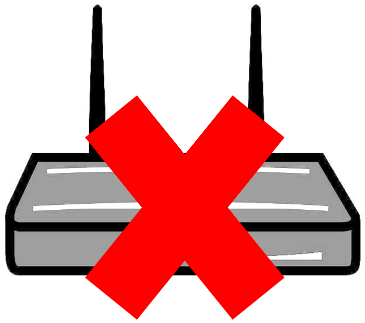
Camera

Recent IoT CVEs



Binary Analysis is Necessary

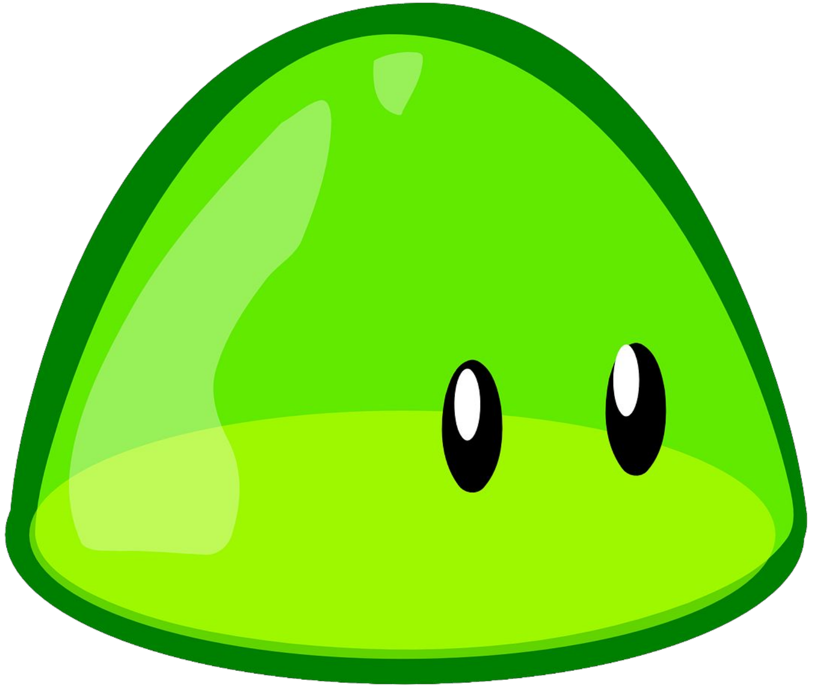




sefcom
security engineering for future computing

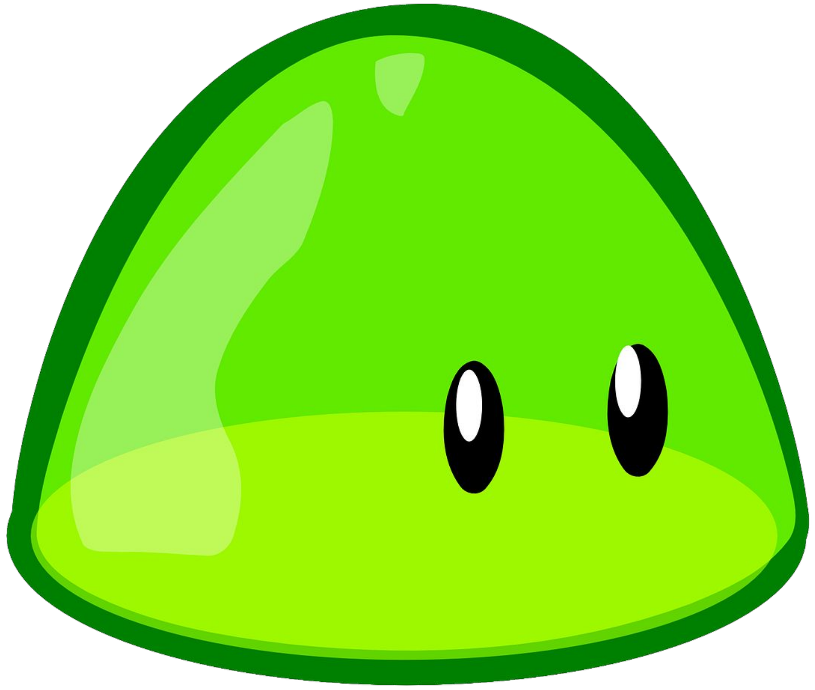
Types of firmware

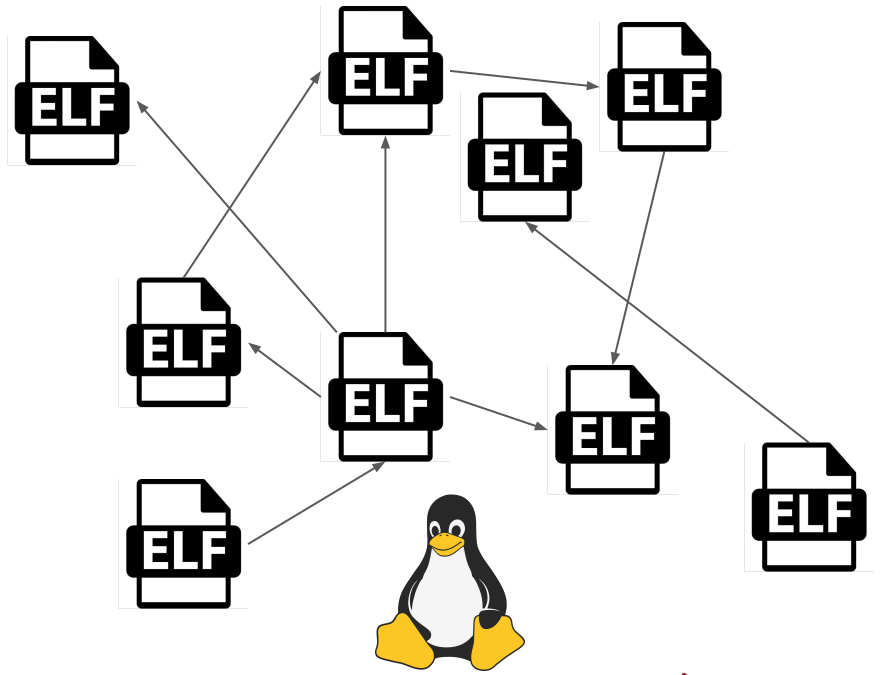
Blob-Based Firmware



security engineering for future computing

Linux-Based Firmware





sefcom
security engineering for future computing

Multi-Binary Dataflow

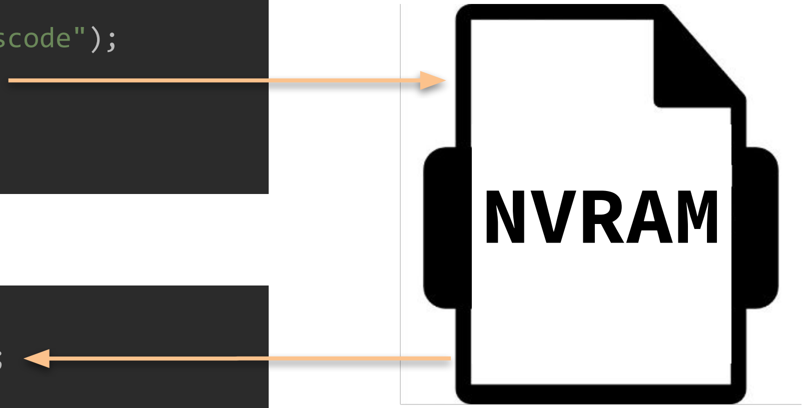
NVRAM: Non-Volatile Random Access Memory

httpd

```
void change_passcode(request) {
    passcode = get_http_param(request, "iserver_passcode");
    nvram_set("iserver_remote_passcode", passcode);
    system("dlnad");
}
```

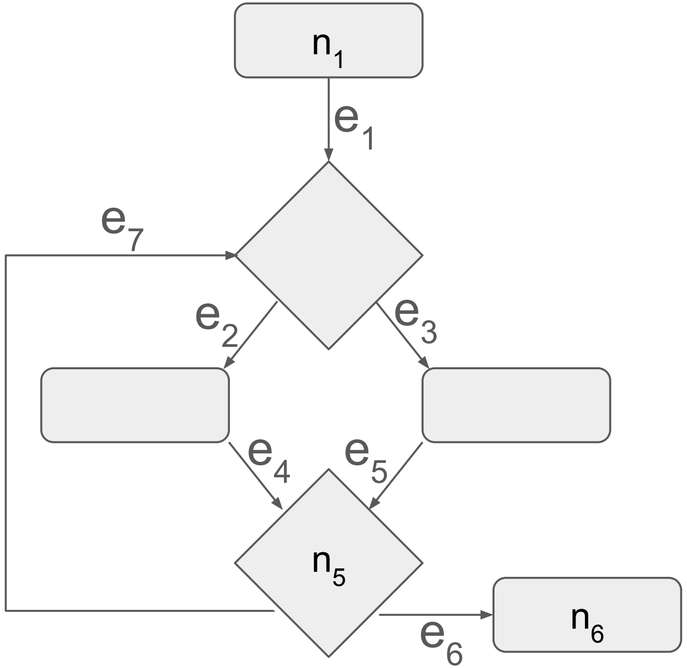
dlnad

```
void main() {
    passcode = nvram_get("iserver_remote_passcode");
    sprintf(v11, "set_password %s", passcode);
    system(v11);
}
```



NETGEAR R6400 Router

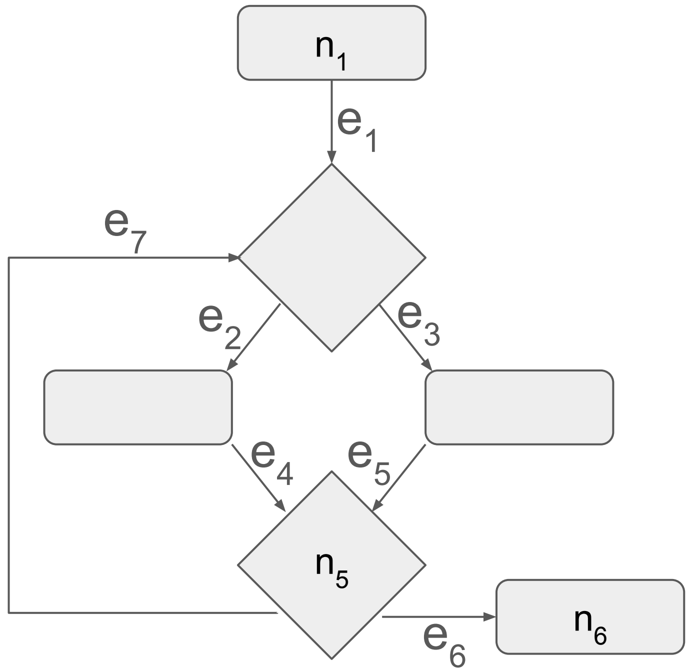
Symbolic Execution

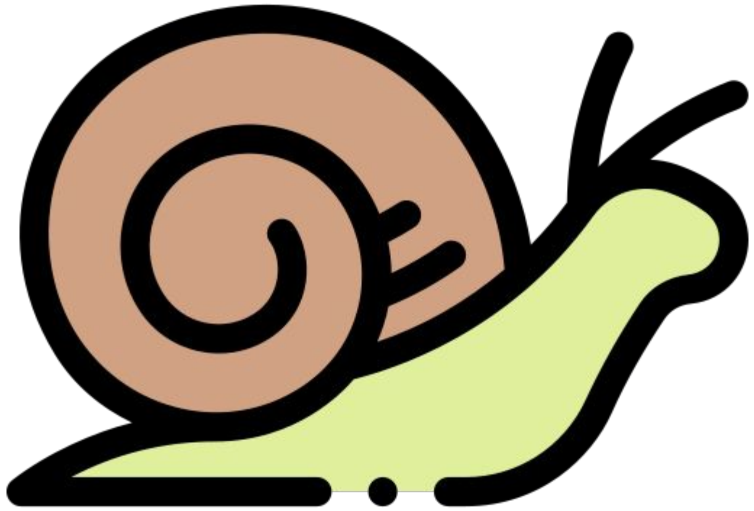


$$n_5 = e_1 \wedge ((e_2 \wedge e_4) \vee (e_3 \wedge e_5))$$

$$n_6 = e_1 \wedge (\sum n_5 \wedge (e_7 \vee e_6))$$

Symbolic Execution Does Not Scale

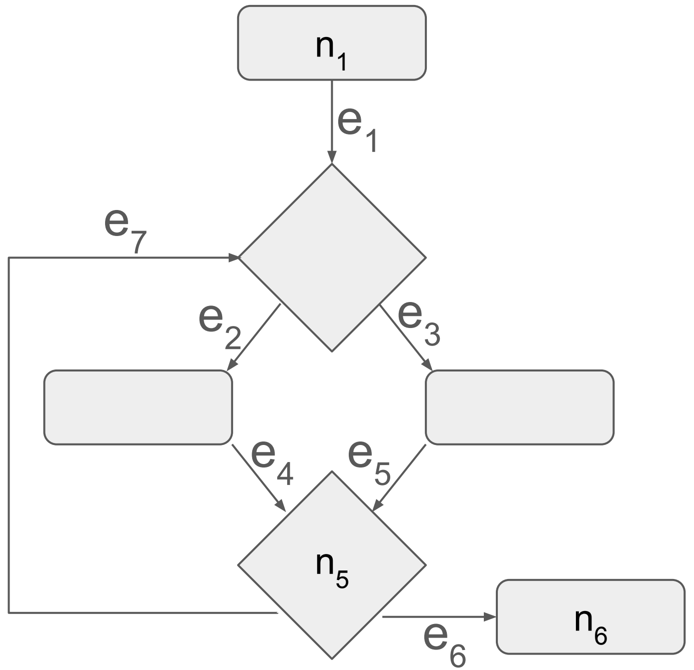




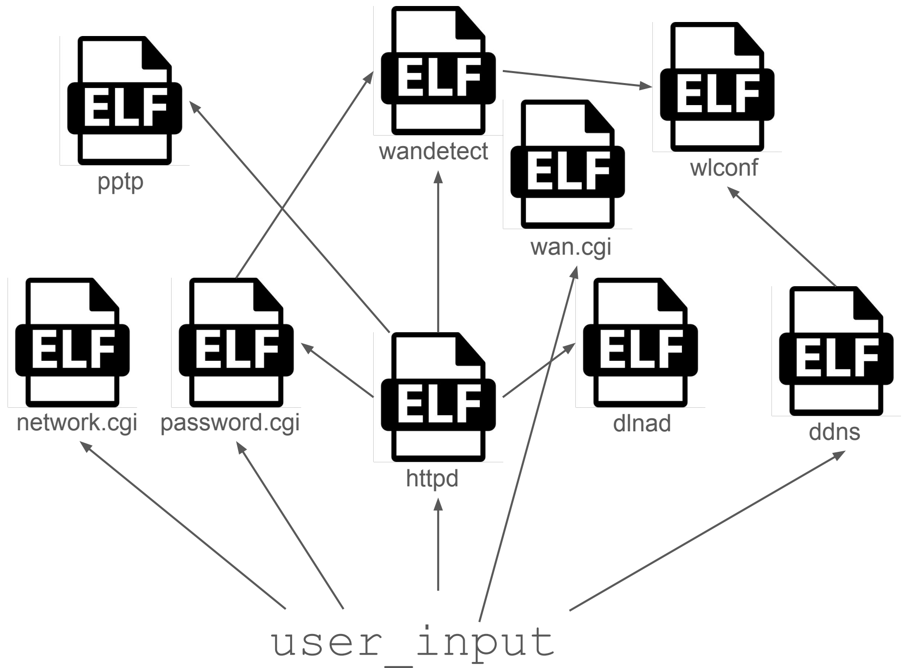
$$n_5 = e_1 \wedge ((e_2 \wedge e_4) \vee (e_3 \wedge e_5))$$

$$n_6 = e_1 \wedge (\sum n_5 \wedge (e_7 \vee e_6))$$

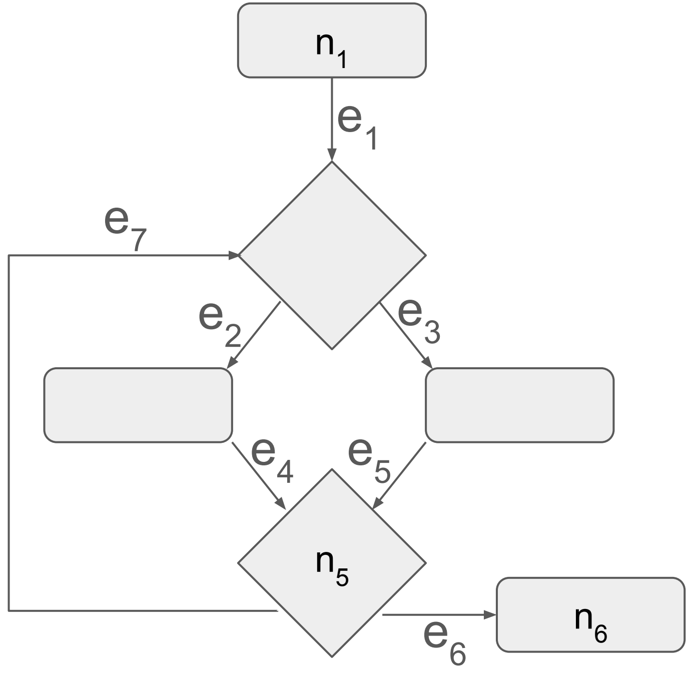
Symbolic Execution Does Not Scale



$$n_5 = e_1 \land ((e_2 \land e_4) \lor (e_3 \land e_5))$$
$$n_6 = e_1 \land (\sum n_5 \land (e_7 \lor e_6))$$

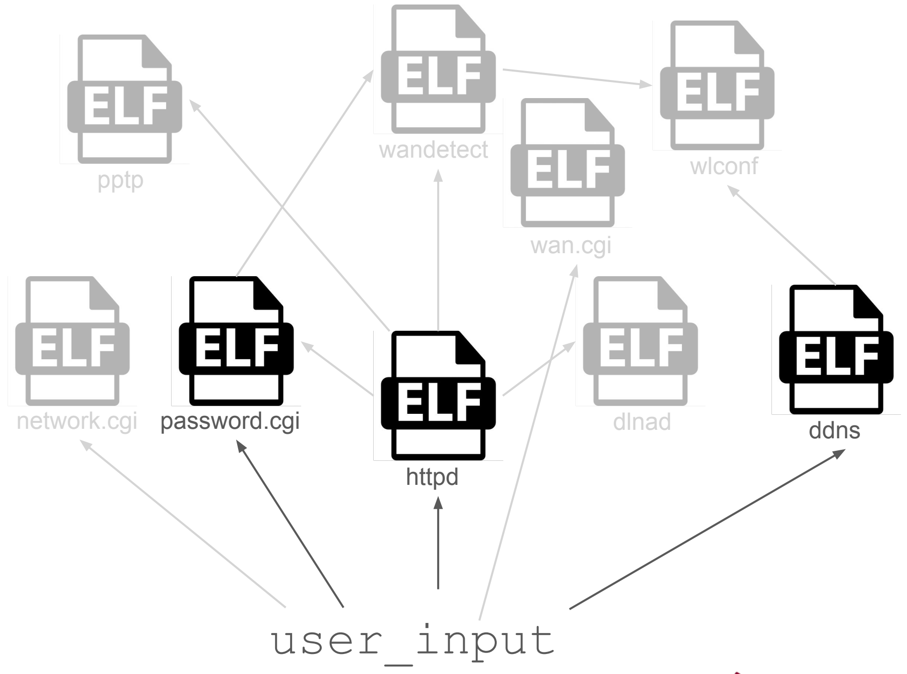


Border Binaries



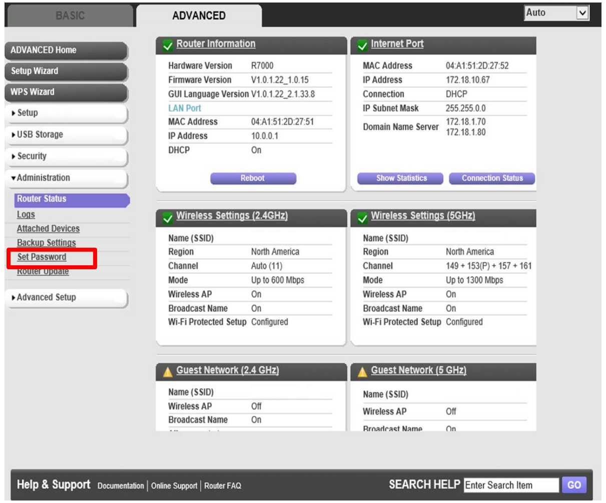
$$n_5 = e_1 \wedge ((e_2 \wedge e_4) \vee (e_3 \wedge e_5))$$

$$n_6 = e_1 \wedge (\sum n_5 \wedge (e_7 \vee e_6))$$



Firmware Dataflow

Frontend



Router Information

Hardware Version	R7000
Firmware Version	V1.0.1.22_1.0.15
GUI Language Version	V1.0.1.22_2.1.33.8
LAN Port	
MAC Address	04:A1:51:2D:27:51
IP Address	10.0.0.1
DHCP	On

Internet Port

MAC Address	04:A1:51:2D:27:52
IP Address	172.18.10.67
Connection	DHCP
IP Subnet Mask	255.255.0.0
Domain Name Server	172.18.1.70 172.18.1.80

Wireless Settings (2.4GHz)

Name (SSID)	
Region	North America
Channel	Auto (11)
Mode	Up to 600 Mbps
Wireless AP	On
Broadcast Name	On
Wi-Fi Protected Setup	Configured

Wireless Settings (5GHz)

Name (SSID)	
Region	North America
Channel	149 + 153(P) + 157 + 161
Mode	Up to 1300 Mbps
Wireless AP	On
Broadcast Name	On
Wi-Fi Protected Setup	Configured

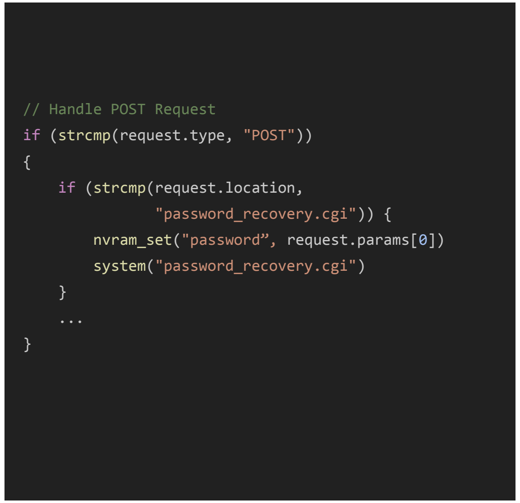
Guest Network (2.4 GHz)

Name (SSID)	
Wireless AP	Off
Broadcast Name	On

Guest Network (5 GHz)

Name (SSID)	
Wireless AP	Off
Broadcast Name	On

Web Server

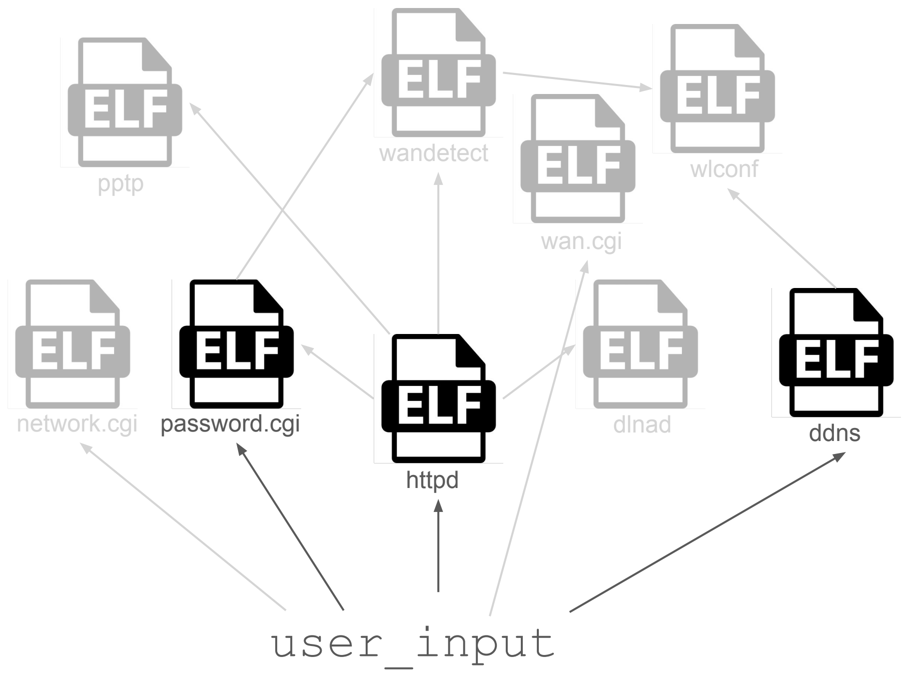


```
// Handle POST Request
if (strcmp(request.type, "POST"))
{
    if (strcmp(request.location,
                "password_recovery.cgi")) {
        nvram_set("password", request.params[0])
        system("password_recovery.cgi")
    }
    ...
}
```

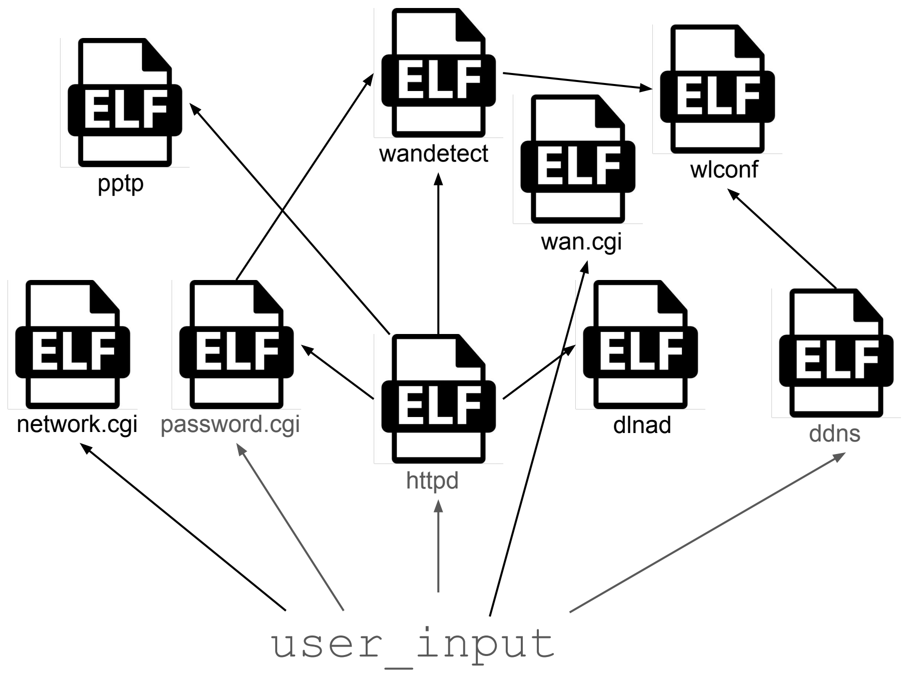
Backend Utilities

password_recovery.cgi

Operation Mango



Operation Mango



Mango — Analysis

Improved Static Analysis for Linux-Based Firmware

Mango—Analysis

Improved Static Analysis for Linux-Based Firmware

- MangoDFA—Value Dependency Analysis

Mango—MangoDFA

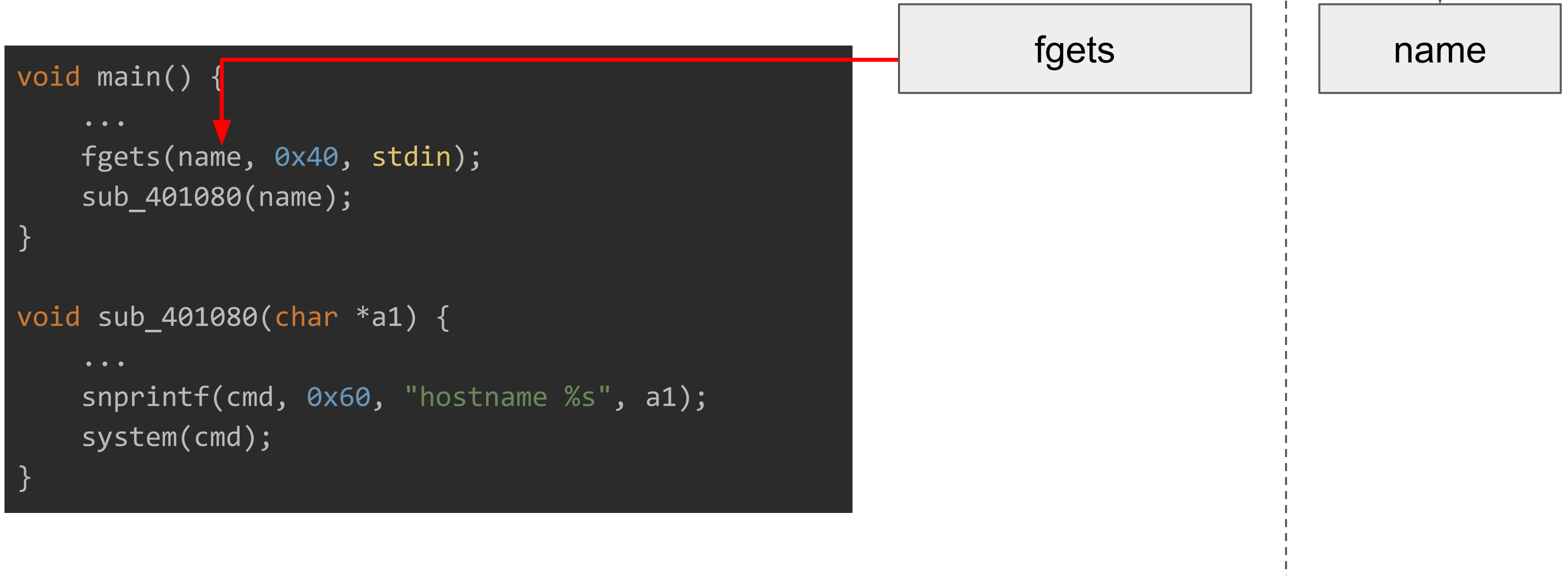
Value Dependency Analysis

```
void main() {
    ...
    fgets(name, 0x40, stdin);
    sub_401080(name);
}

void sub_401080(char *a1) {
    ...
    snprintf(cmd, 0x60, "hostname %s", a1);
    system(cmd);
}
```

sefcom
security engineering for future computing

Mango—MangoDFA

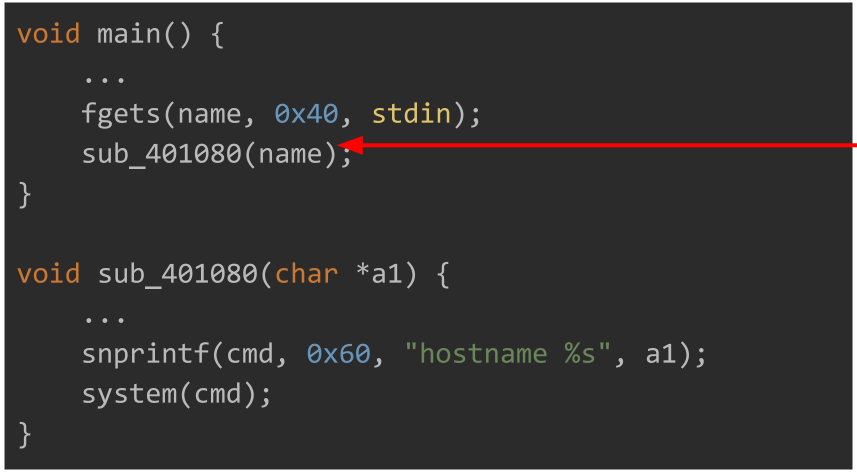
Value Dependency Analysis



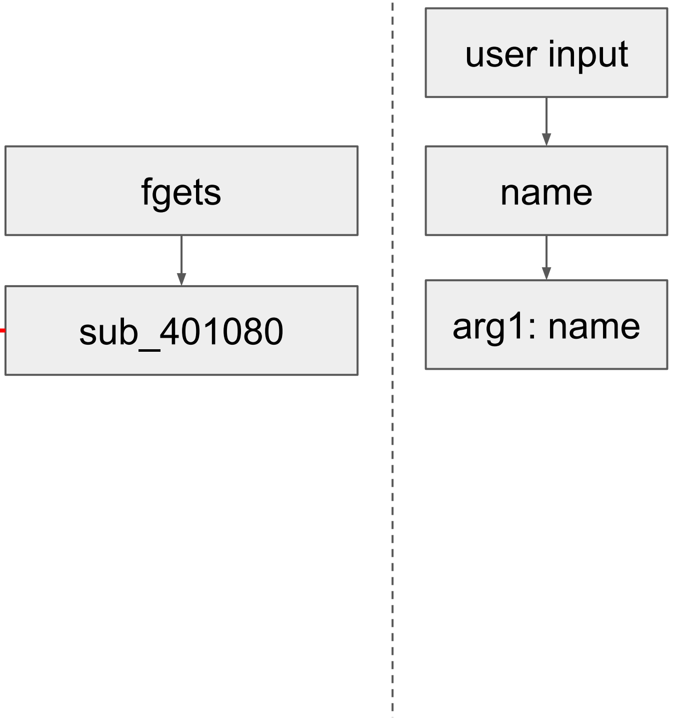
```
void main() {
    ...
    fgets(name, 0x40, stdin);
    sub_401080(name);
}

void sub_401080(char *a1) {
    ...
    snprintf(cmd, 0x60, "hostname %s", a1);
    system(cmd);
}
```

Mango—MangoDFA

Value Dependency Analysis



```
void main() {
    ...
    fgets(name, 0x40, stdin);
    sub_401080(name);
}

void sub_401080(char *a1) {
    ...
    snprintf(cmd, 0x60, "hostname %s", a1);
    system(cmd);
}
```

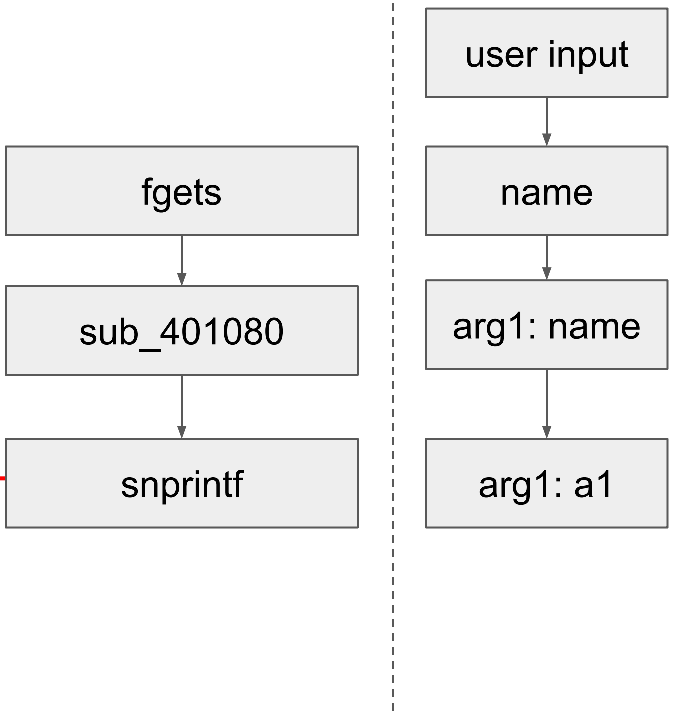


$$\begin{tikzcd}
 & \text{user input} \arrow[d] \\
\text{fgets} \arrow[d] & \text{name} \arrow[d] \\
\text{sub\_401080} & \text{arg1: name}
\end{tikzcd}$$

Mango—MangoDFA

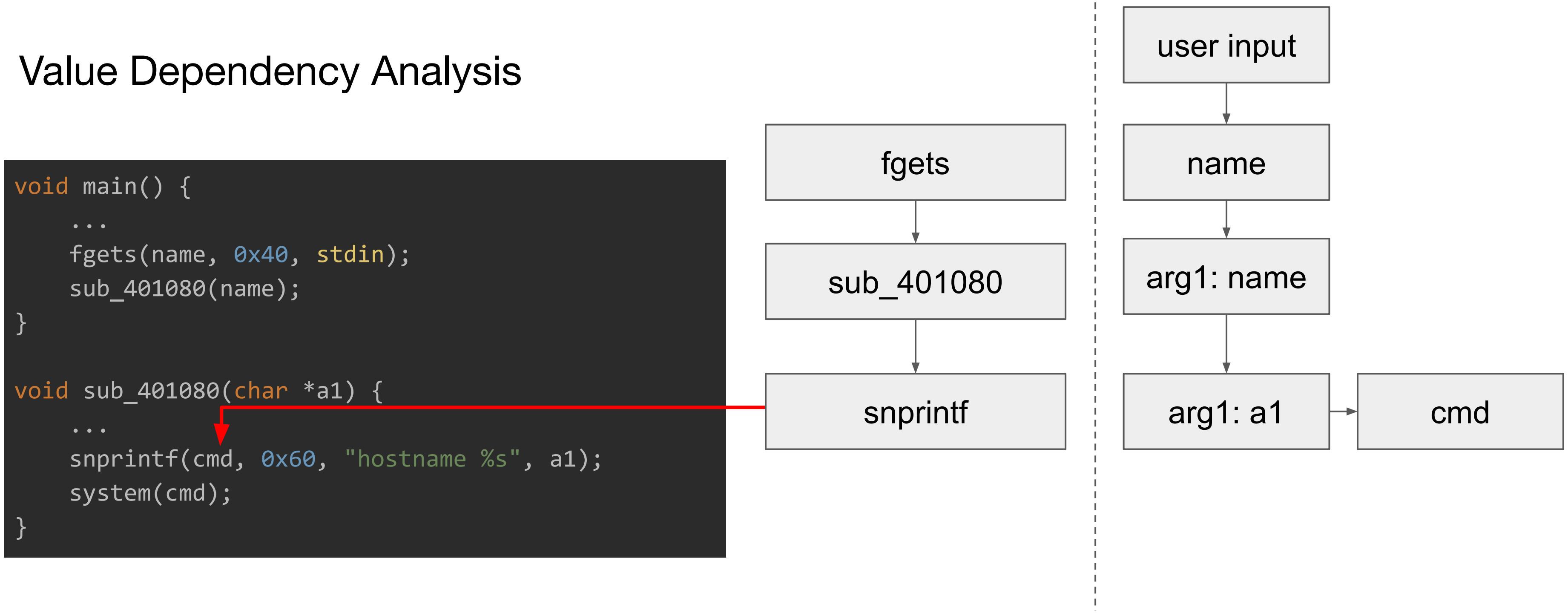
Value Dependency Analysis

```
void main() {
    ...
    fgets(name, 0x40, stdin);
    sub_401080(name);
}

void sub_401080(char *a1) {
    ...
    snprintf(cmd, 0x60, "hostname %s", a1);
    system(cmd);
}
```



Mango—MangoDFA

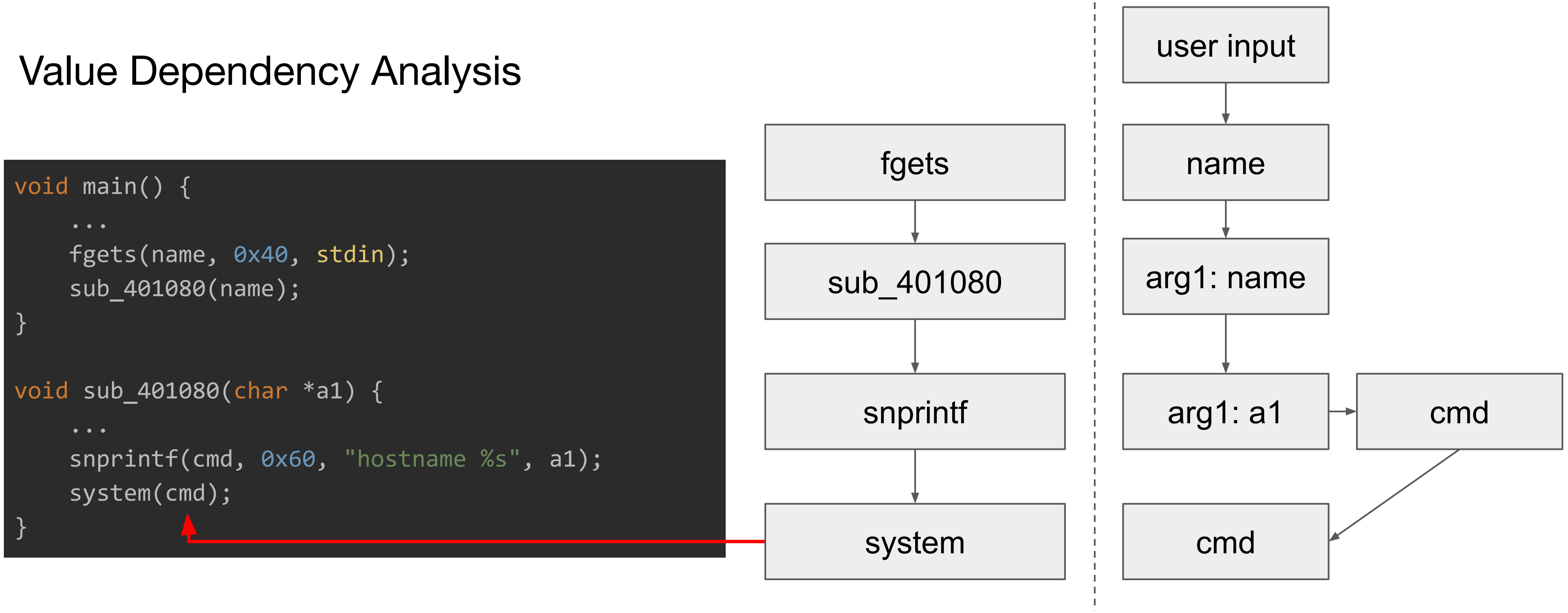
Value Dependency Analysis



```
void main() {
    ...
    fgets(name, 0x40, stdin);
    sub_401080(name);
}

void sub_401080(char *a1) {
    ...
    snprintf(cmd, 0x60, "hostname %s", a1);
    system(cmd);
}
```

Mango — MangoDFA

Value Dependency Analysis



```
void main() {
    ...
    fgets(name, 0x40, stdin);
    sub_401080(name);
}

void sub_401080(char *a1) {
    ...
    snprintf(cmd, 0x60, "hostname %s", a1);
    system(cmd);
}
```

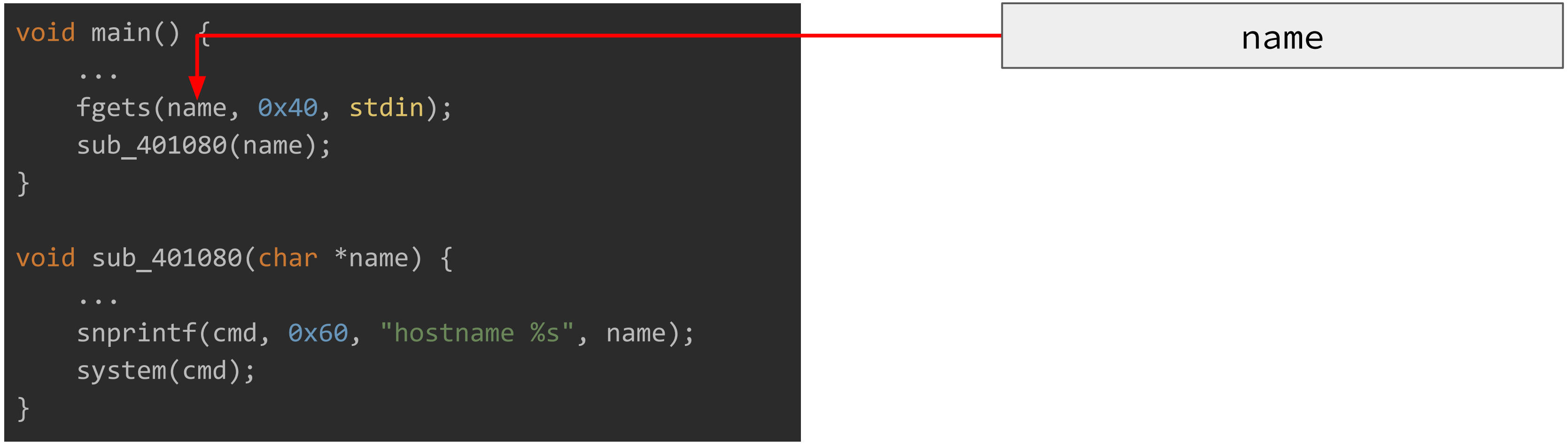
Mango—Analysis

Improved Static Analysis for Linux-Based Firmware

- MangoDFA—Value Dependency Analysis
- Rich Expressions

Mango—Rich Expressions

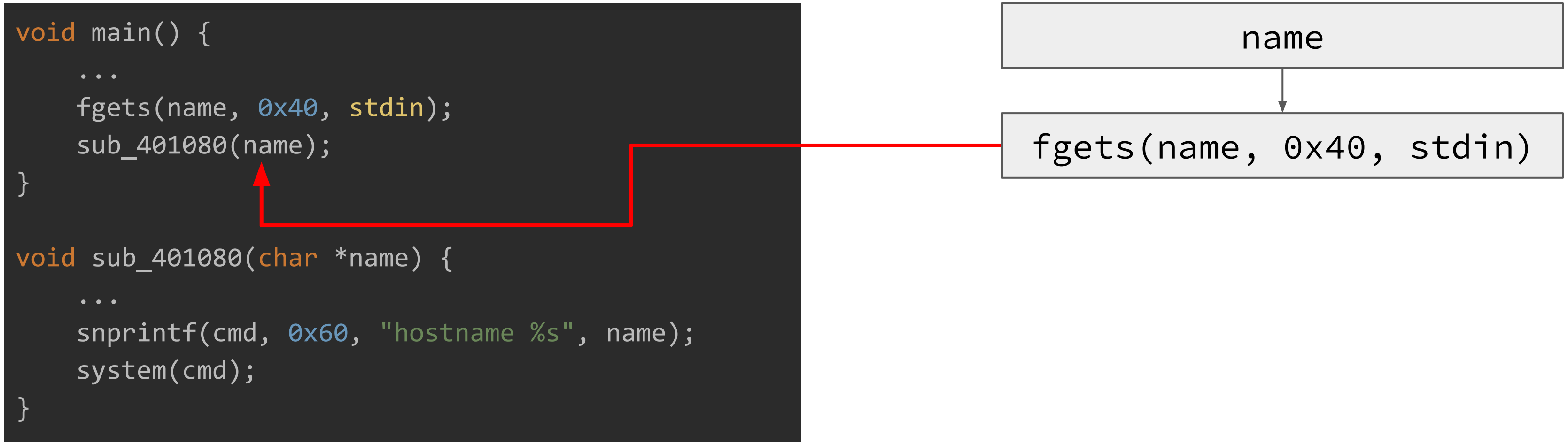
Value Dependency Tracking with Rich Expressions



```
void main() {
    ...
    fgets(name, 0x40, stdin);
    sub_401080(name);
}

void sub_401080(char *name) {
    ...
    snprintf(cmd, 0x60, "hostname %s", name);
    system(cmd);
}
```

Mango — MangoDFA

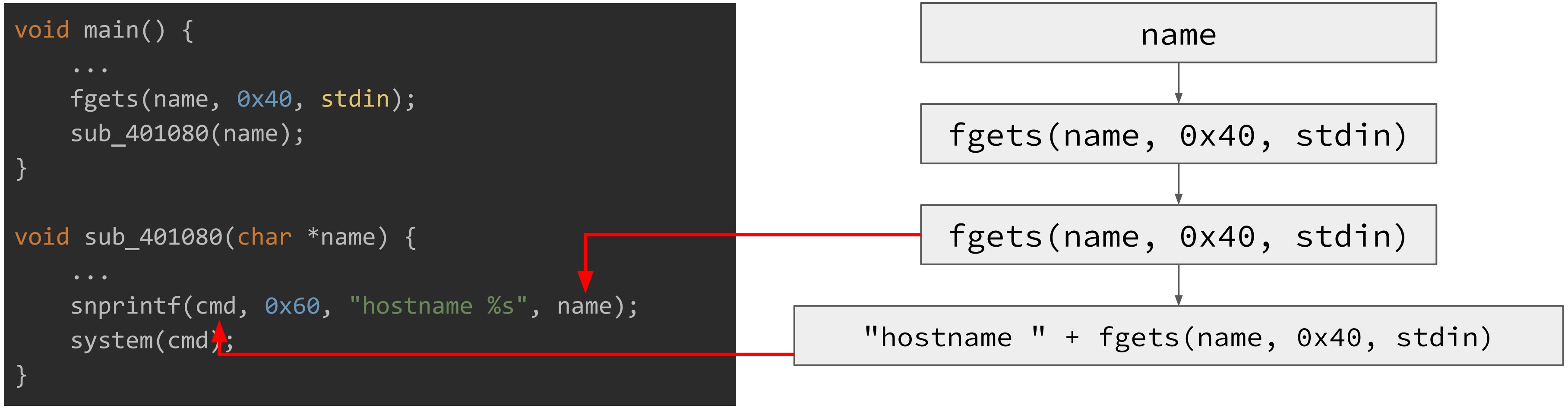
Value Dependency Tracking with Rich Expressions



```
void main() {
    ...
    fgets(name, 0x40, stdin);
    sub_401080(name);
}

void sub_401080(char *name) {
    ...
    snprintf(cmd, 0x60, "hostname %s", name);
    system(cmd);
}
```

Mango — MangoDFA

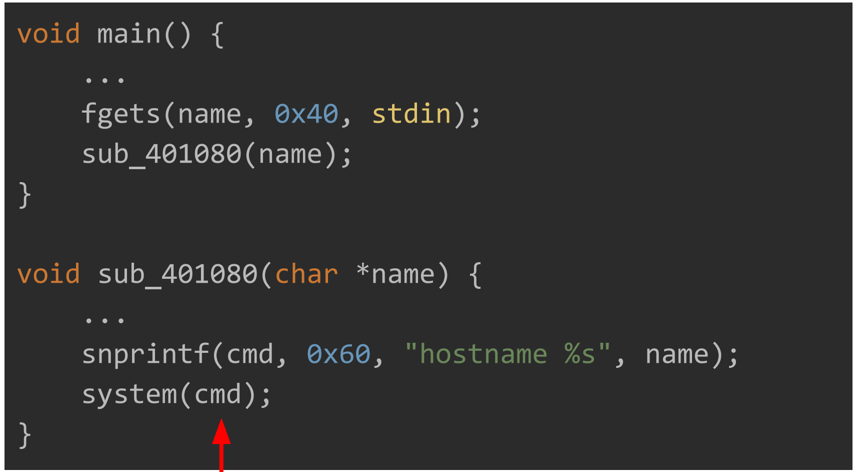
Value Dependency Tracking with Rich Expressions



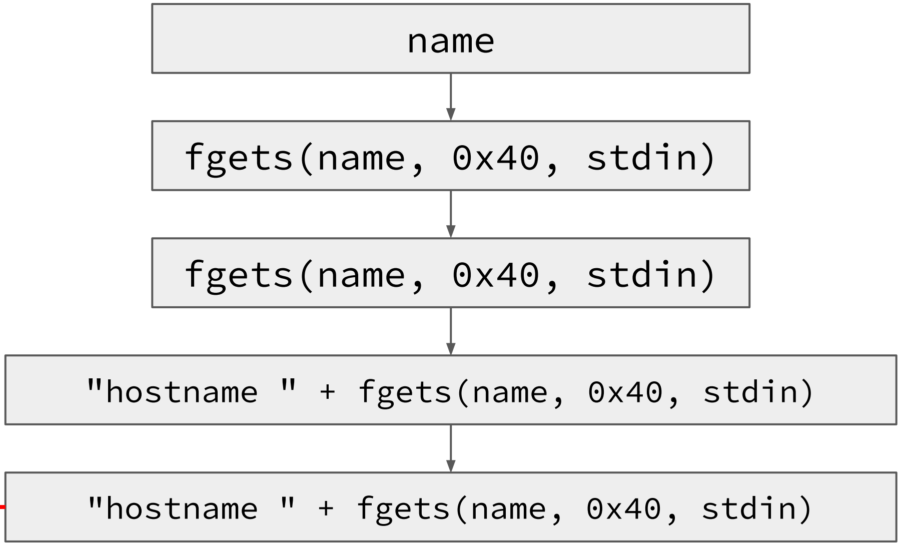
```
void main() {
    ...
    fgets(name, 0x40, stdin);
    sub_401080(name);
}

void sub_401080(char *name) {
    ...
    snprintf(cmd, 0x60, "hostname %s", name);
    system(cmd);
}
```

Mango—MangoDFA

Value Dependency Tracking with Rich Expressions



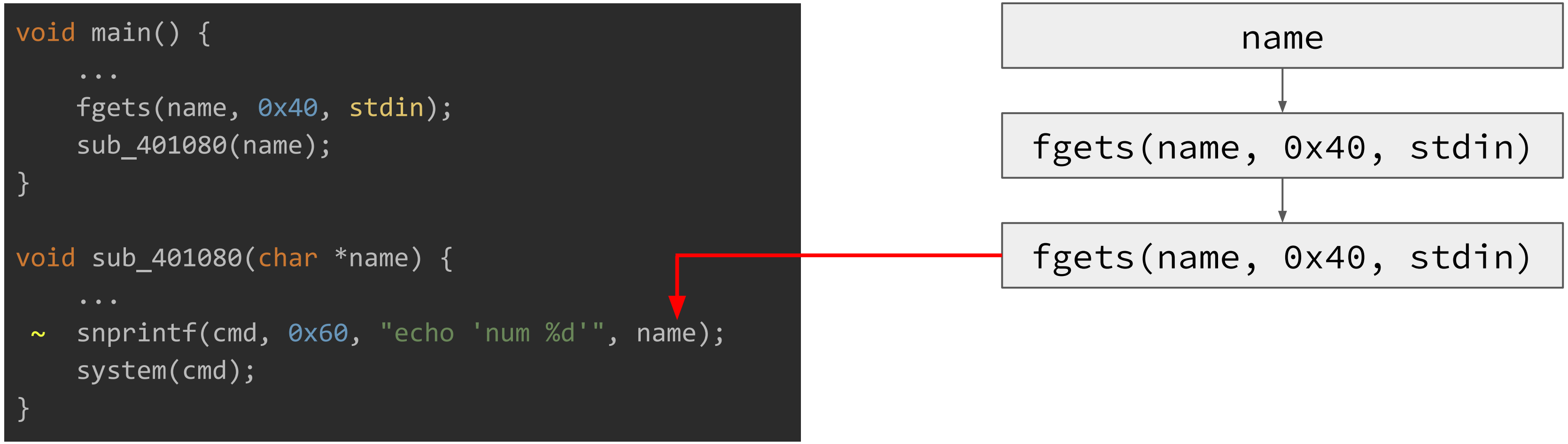
```c
void main() {
    ...
    fgets(name, 0x40, stdin);
    sub_401080(name);
}

void sub_401080(char *name) {
    ...
    snprintf(cmd, 0x60, "hostname %s", name);
    system(cmd);
}
```



name
fgets(name, 0x40, stdin)
fgets(name, 0x40, stdin)
"hostname " + fgets(name, 0x40, stdin)
"hostname " + fgets(name, 0x40, stdin)

Mango—MangoDFA

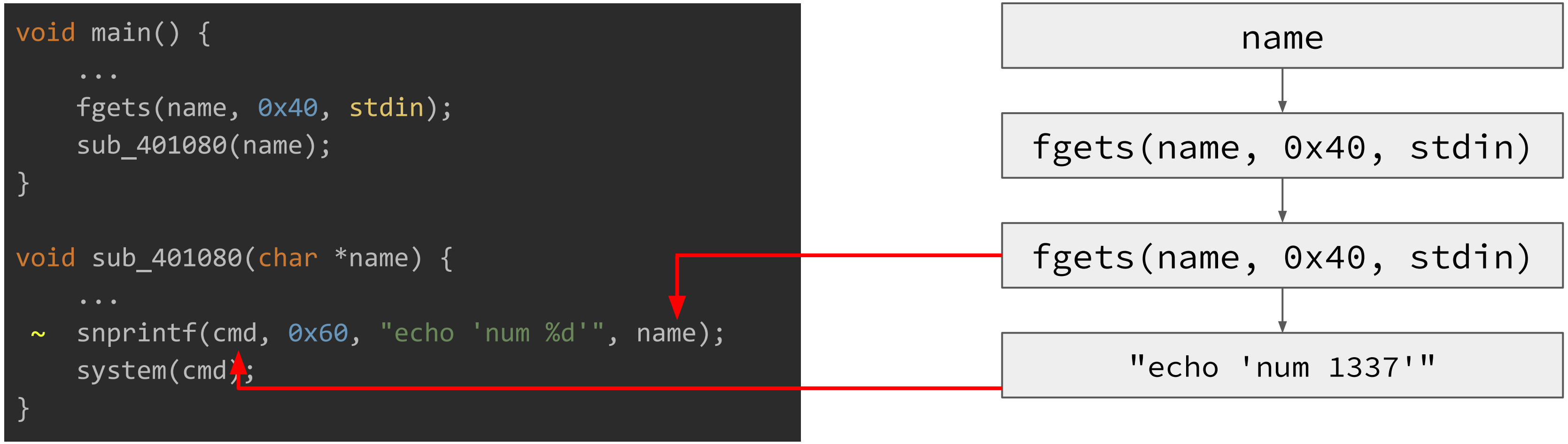
Rich Expressions with Value Transformations



```
void main() {
    ...
    fgets(name, 0x40, stdin);
    sub_401080(name);
}

void sub_401080(char *name) {
    ...
 ~  snprintf(cmd, 0x60, "echo 'num %d'", name);
    system(cmd);
}
```

Mango—MangoDFA

Rich Expressions with Value Transformations



```
void main() {
    ...
    fgets(name, 0x40, stdin);
    sub_401080(name);
}

void sub_401080(char *name) {
    ...
 ~  snprintf(cmd, 0x60, "echo 'num %d'", name);
    system(cmd);
}
```

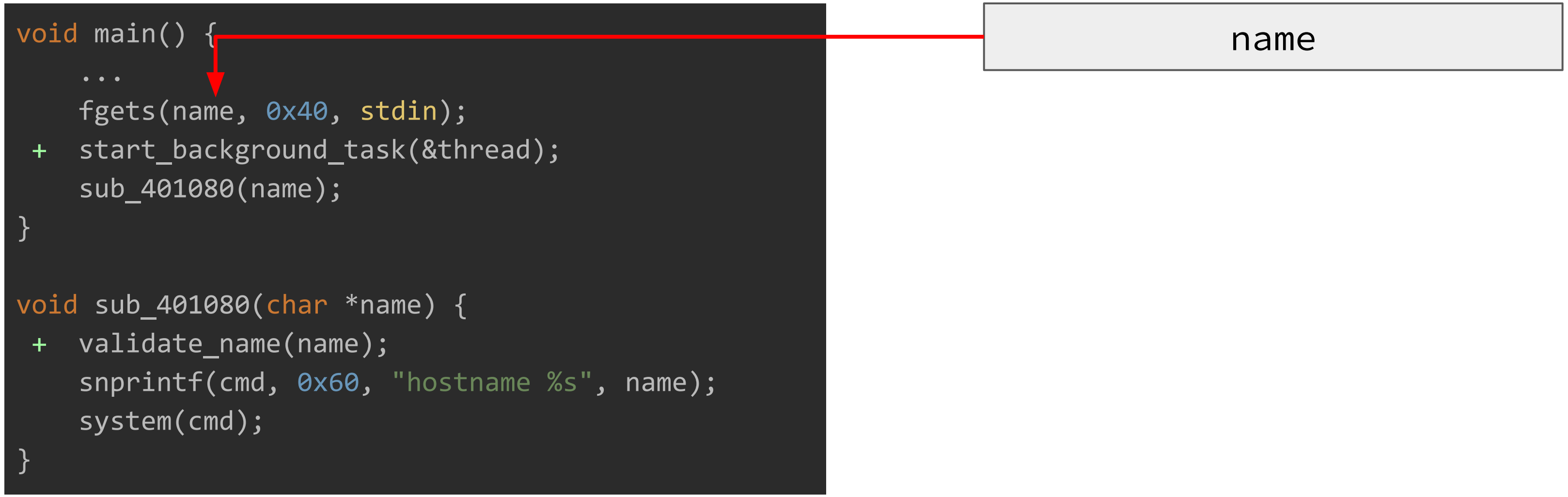
Mango—Analysis

Improved Static Analysis for Linux-Based Firmware

- MangoDFA—Value Dependency Analysis
- Rich Expressions
- Assumed Nonimpact

Mango—Assumed Nonimpact

Investigating Functions Based on Impact



```
void main() {
    ...
    fgets(name, 0x40, stdin);
 +  start_background_task(&thread);
    sub_401080(name);
}

void sub_401080(char *name) {
 +  validate_name(name);
    snprintf(cmd, 0x60, "hostname %s", name);
    system(cmd);
}
```

Mango — Assumed Nonimpact

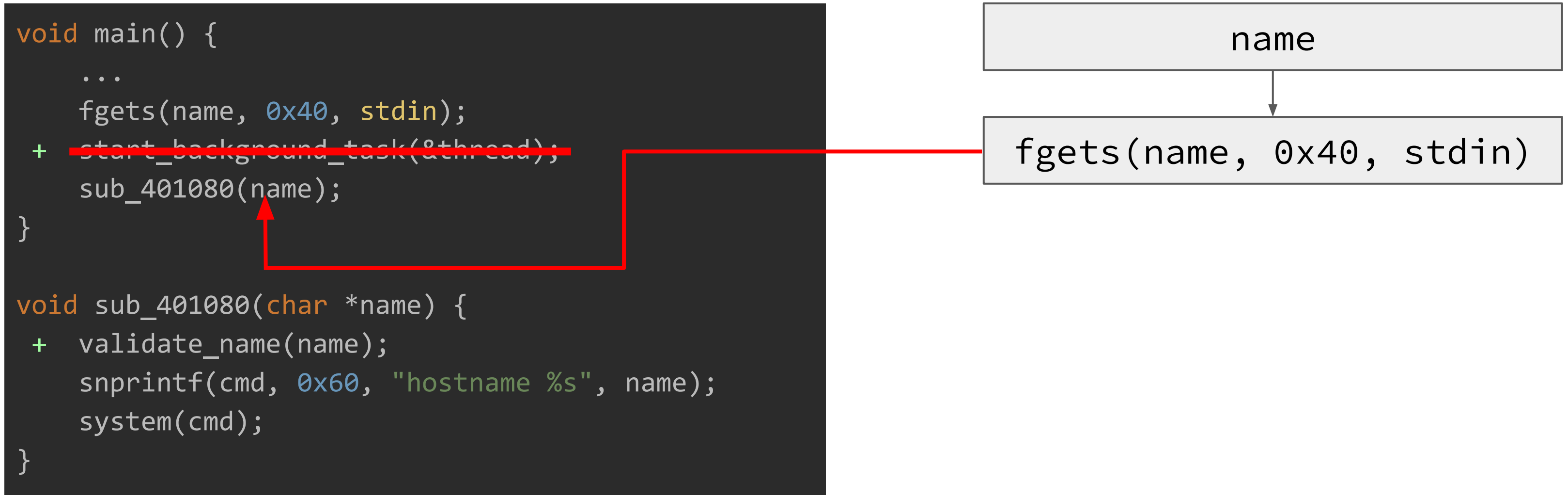
Investigating Functions Based on Impact

```
void main() {
    ...
    fgets(name, 0x40, stdin);
 +  ~~start_background_task(&thread);~~
    sub_401080(name);
}

void sub_401080(char *name) {
 +  validate_name(name);
    snprintf(cmd, 0x60, "hostname %s", name);
    system(cmd);
}
```

name

Mango—Assumed Nonimpact

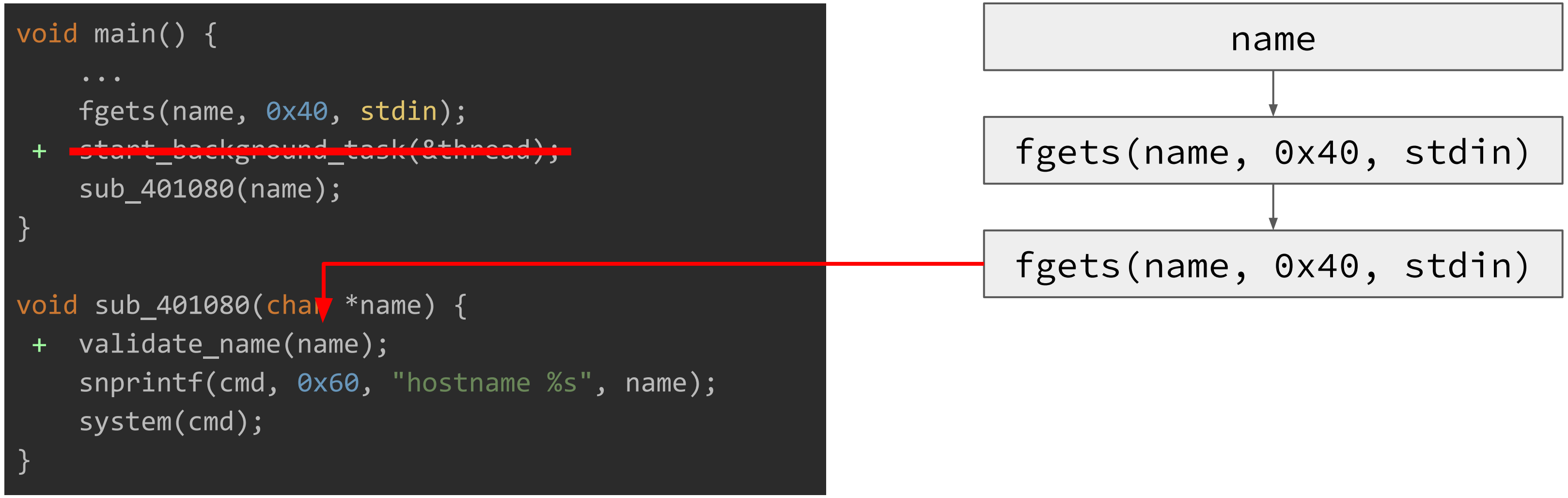
Investigating Functions Based on Impact



```
void main() {
    ...
    fgets(name, 0x40, stdin);
 +  ~~start_background_task(&thread);~~
    sub_401080(name);
}

void sub_401080(char *name) {
 +  validate_name(name);
    snprintf(cmd, 0x60, "hostname %s", name);
    system(cmd);
}
```

Mango—Assumed Nonimpact

Investigating Functions Based on Impact



```
void main() {
    ...
    fgets(name, 0x40, stdin);
 +  ~~start_background_task(&thread);~~
    sub_401080(name);
}

void sub_401080(char *name) {
 +  validate_name(name);
    snprintf(cmd, 0x60, "hostname %s", name);
    system(cmd);
}
```

sefcom security engineering for future computing

Mango—Analysis

Improved Static Analysis for Linux-Based Firmware

- MangoDFA—Value Dependency Analysis
- Rich Expressions
- Assumed Nonimpact
- Sink-to-Source Coarse-Grained Analysis

Mango—Sink-to-Source Coarse-Grained Analysis

Tracing Backwards Through a Call Trace

```
void main() {
    ...
    fgets(name, 0x40, stdin);
 +  start_background_task(&thread);
    sub_401080(name);
}

void sub_401080(char *name) {
 +  validate_name(name);
    snprintf(cmd, 0x60, "hostname %s", name);
    system(cmd);
}
```

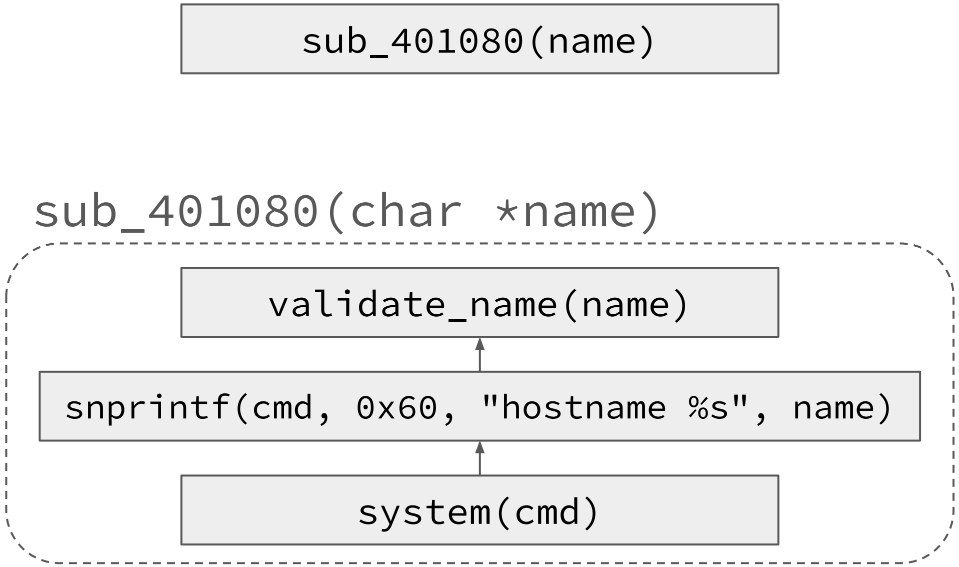
```
system(cmd)
```

sefcom security engineering for future computing

Mango—Sink-to-Source Coarse-Grained Analysis

Tracing Backwards Through a Call Trace

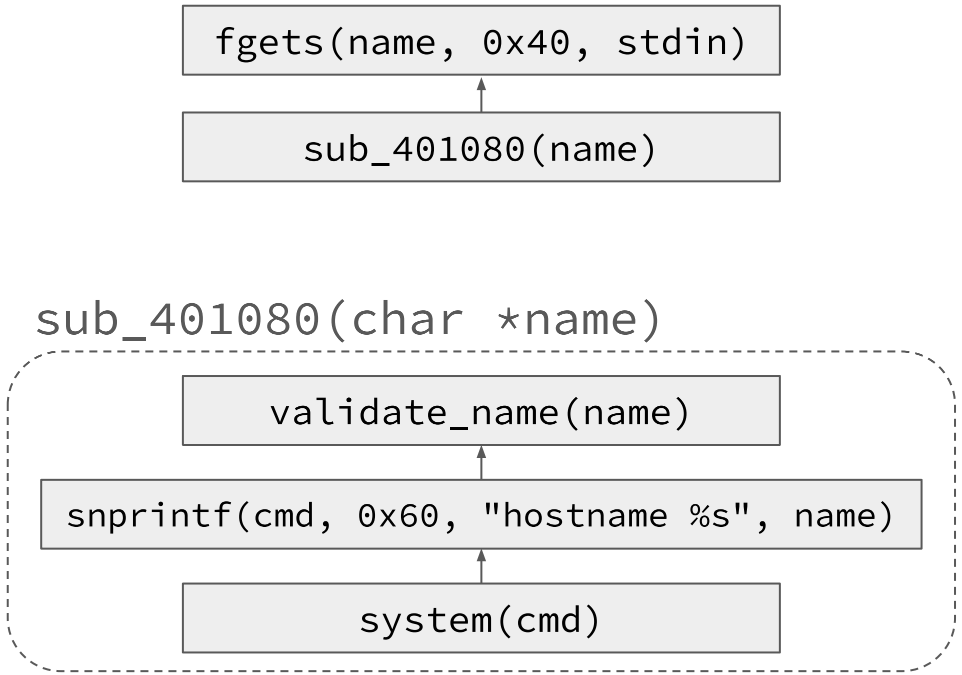
```
void main() {
    ...
    fgets(name, 0x40, stdin);
 +  start_background_task(&thread);
    sub_401080(name);
}

void sub_401080(char *name) {
 +  validate_name(name);
    snprintf(cmd, 0x60, "hostname %s", name);
    system(cmd);
}
```



Mango—Sink-to-Source Coarse-Grained Analysis

Tracing Backwards Through a Call Trace

```
void main() {
    ...
    fgets(name, 0x40, stdin);
 +  start_background_task(&thread);
    sub_401080(name);
}

void sub_401080(char *name) {
 +  validate_name(name);
    snprintf(cmd, 0x60, "hostname %s", name);
    system(cmd);
}
```

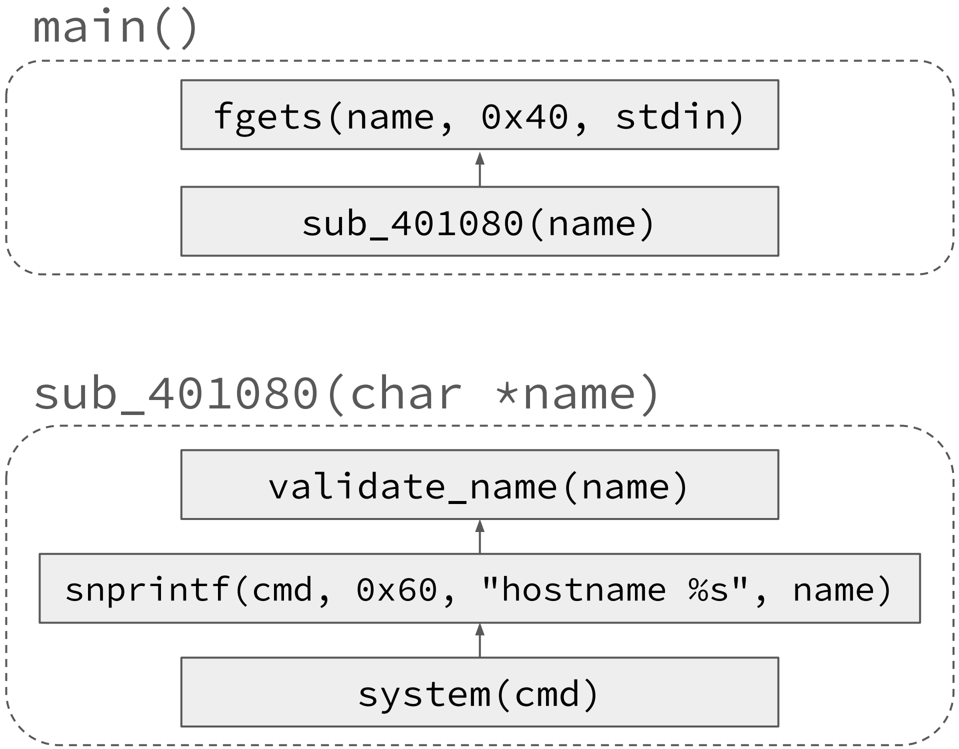


Mango—Sink-to-Source Coarse-Grained Analysis

Analysis Result:

`main -> sub_401080`

```
void main() {
    ...
    fgets(name, 0x40, stdin);
 +  start_background_task(&thread);
    sub_401080(name);
}

void sub_401080(char *name) {
 +  validate_name(name);
    snprintf(cmd, 0x60, "hostname %s", name);
    system(cmd);
}
```

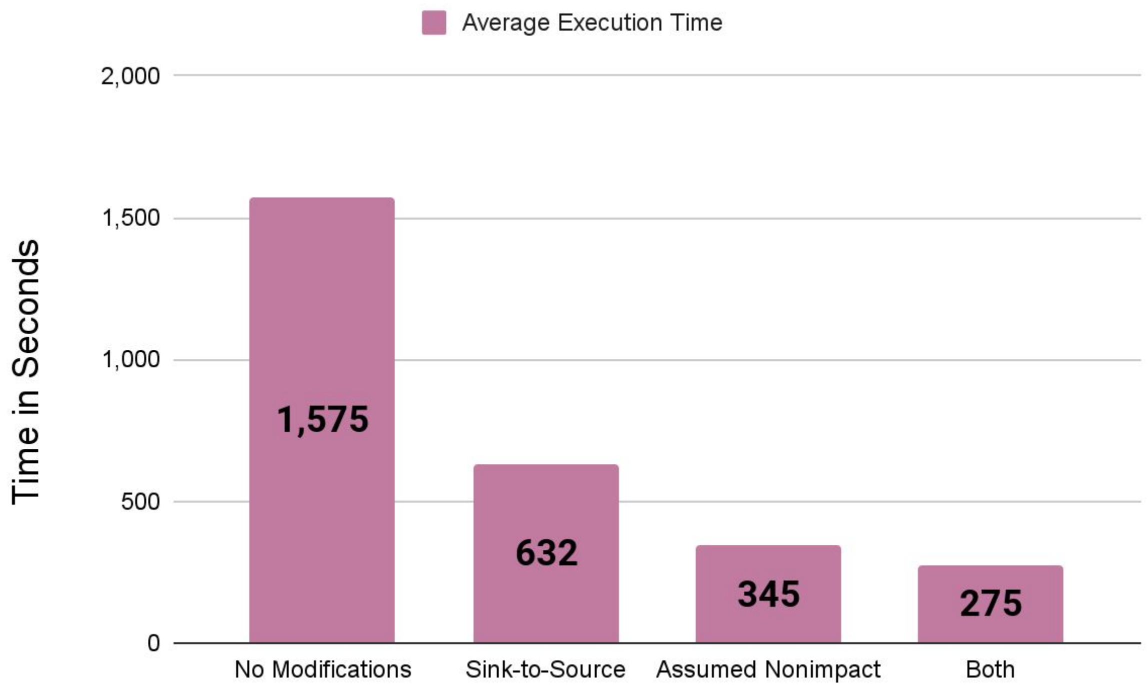


Mango—Analysis

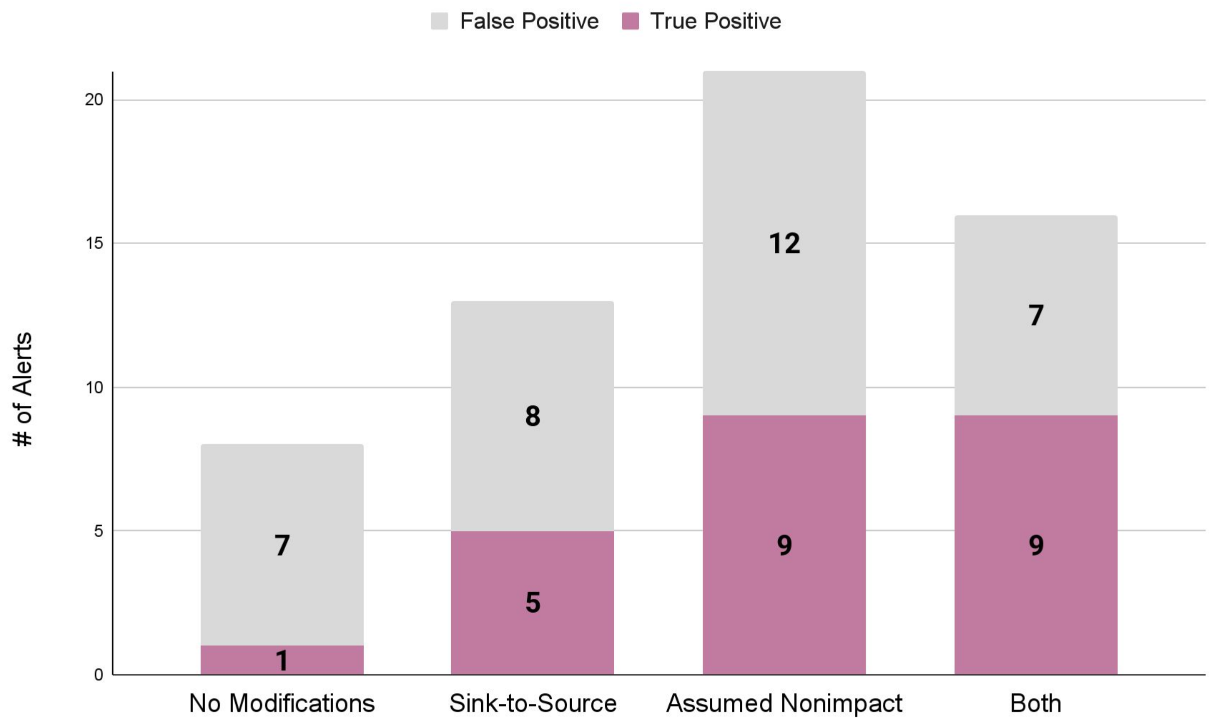
Improved Static Analysis for Linux-Based Firmware

- MangoDFA—Value Dependency Analysis
- Rich Expressions
- Assumed Nonimpact
- Sink-to-Source Coarse-Grained Analysis

Mango—Time Ablation



Mango—Alert Ablation



Mango — Efficacy

49 Firmware KARONTE Dataset

Mango — Efficacy

49 Firmware KARONTE Dataset

- 2,310 Alerts

Mango — Efficacy

49 Firmware KARONTE Dataset

- 2,310 Alerts
- 100 command injections
  - 57 True Positives
  - 57% TP Rate

Mango — Efficacy

49 Firmware KARONTE Dataset

- 2,310 Alerts
- 100 command injections
  - 57 True Positives
  - 57% TP Rate
- 230 buffer overflows
  - 109 True Positives
  - 47% TP Rate

Mango — Efficacy

49 Firmware KARONTE Dataset

- 2,310 Alerts
- 100 command injections
    - 57 True Positives
    - 57% TP Rate
- 230 buffer overflows
    - 109 True Positives
    - 47% TP Rate
- Total: 52% TP Rate

Operation Mango vs SaTC

	Analyzed Binaries	Alerts	Alerted Binaries	Runtime
SaTC				
Operation Mango				

sefcom
security engineering for future computing

Operation Mango vs SaTC

	Analyzed Binaries	Alerts	Alerted Binaries	Runtime
SaTC	131			
Operation Mango	3,599			

Operation Mango vs SaTC

	Analyzed Binaries	Alerts	Alerted Binaries	Runtime
SaTC	131	144		
Operation Mango	3,599	2,310		

sefcom
security engineering for future computing

Operation Mango vs SaTC

	Analyzed Binaries	Alerts	Alerted Binaries	Runtime
SaTC	131	144	52	
Operation Mango	3,599	2,310	174	

sefcom
security engineering for future computing

Operation Mango vs SaTC

	Analyzed Binaries	Alerts	Alerted Binaries	Runtime
SaTC	131	144	52	860:58h
Operation Mango	3,599	2,310	174	946:22h

sefcom
security engineering for future computing

Mango—Summary

- Found bugs in 3x more binaries at 22x faster per binary

Mango—Summary

- Found bugs in 3x more binaries at 22x faster per binary
- Manually verified 166 exploitable vulnerabilities

Mango—Summary

- Found bugs in 3x more binaries at 22x faster per binary
- Manually verified 166 exploitable vulnerabilities
- Operation Mango is open-sourced and available

Mango—Summary

- Found bugs in 3x more binaries at 22x faster per binary
- Manually verified 166 exploitable vulnerabilities
- Operation Mango is open-sourced and available
- Operation Mango is integrated in angr and will be maintained

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

Thank You!

Operation Mango
https://github.com/sefcom/operation-mango-public

Wil Gibbs | wilgibbs.com | wfgibbs@asu.edu | @cl4sm

Arizona State University

UCSB
UNIVERSITY OF CALIFORNIA
SANTA BARBARA

sefcom
security engineering for future computing