Taint Checker : Tracking the Taint Propagation on Binary Programs

Sangjun Park
in class IS661

KAIST

Outline

Motivation

Problem

Idea and Challenges

Evaluation and observation

Conclusion

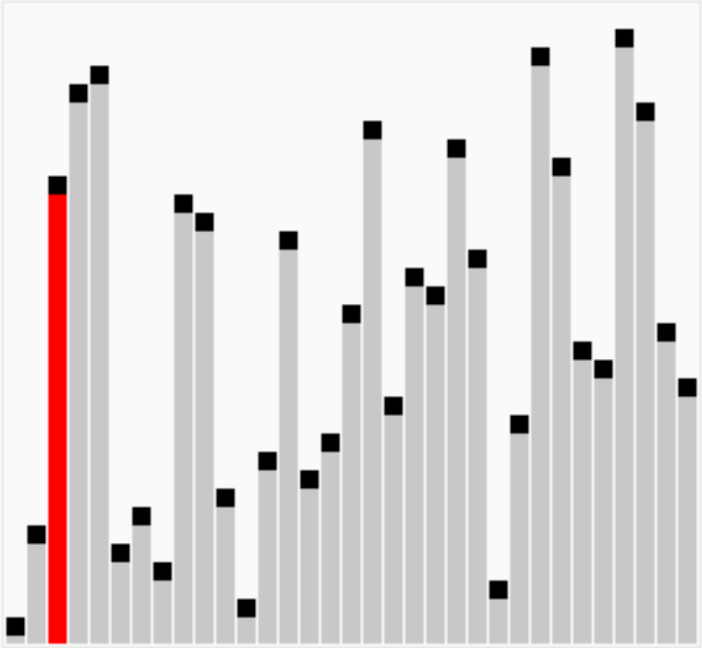
Importance of Visualization

- Observation is the beginning of all sciences

Importance of Visualization

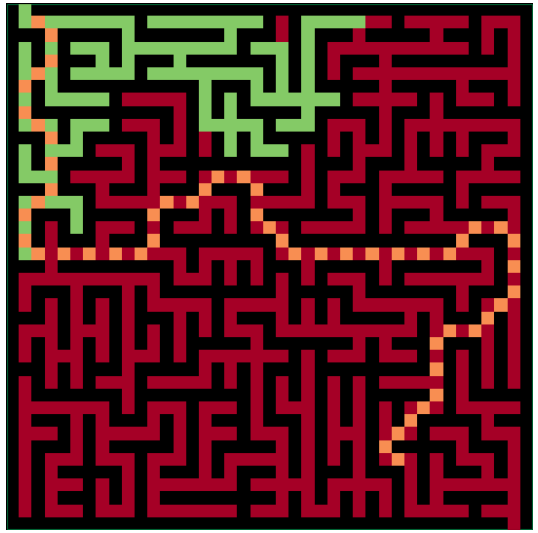
- Observation is the beginning of all sciences
- Only when we observe the behavior of large and complicated software systems, we can understand how they work and discover new knowledge

Observation



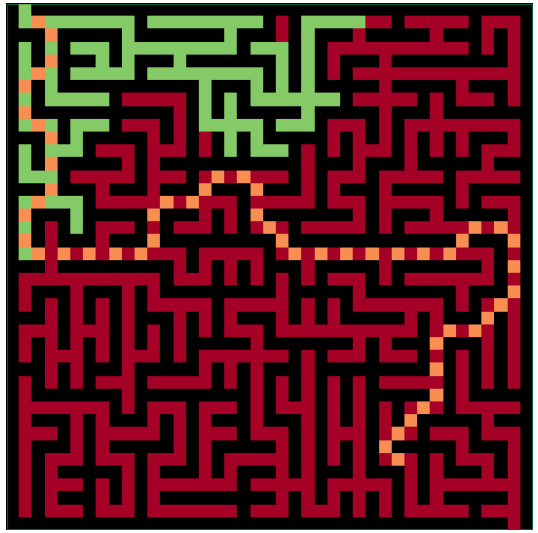
Bubble sort

Visualization Examples



Fuzzle: Haeun Lee et al., 2022

Visualization Examples

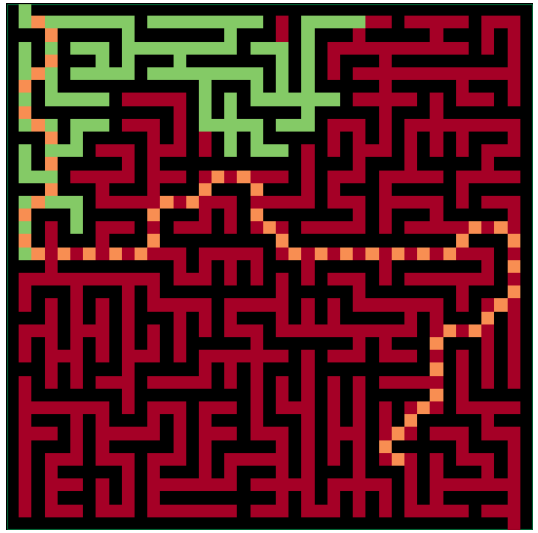


Fuzzle: Haeun Lee et al., 2022

```
637        1 :   if (fstat (STDOUT_FILENO, &stat_buf) < 0)
638        0 :     die (EXIT_FAILURE, errno, _("standard output"));
639          :
640        1 :   outsize = io_blksize (stat_buf);
641        1 :   out_dev = stat_buf.st_dev;
642        1 :   out_ino = stat_buf.st_ino;
643        1 :   out_isreg = S_ISREG (stat_buf.st_mode) != 0;
644          :
645        1 :   if (! (number || show_ends || squeeze_blank))
646        0 :     {
647          :       file_open_mode |= O_BINARY;
648        0 :       if (O_BINARY && ! isatty (STDOUT_FILENO))
649        0 :         xfreopen (NULL, "wb", stdout);
650          :     }
651        0 :
652          :   /* Check if any of the input files are the same as the output file.  */
653          :
654          :   /* Main loop.  */
655        0 :
656        1 :   infile = "-";
657        1 :   argind = optind;
658          :
659          :   do
660          :     {
661        1 :       if (argind < argc)
662        1 :         infile = argv[argind];
663          :
664        1 :       if (STREQ (infile, "-"))
```

LCOV: Linux Test Project, 2001

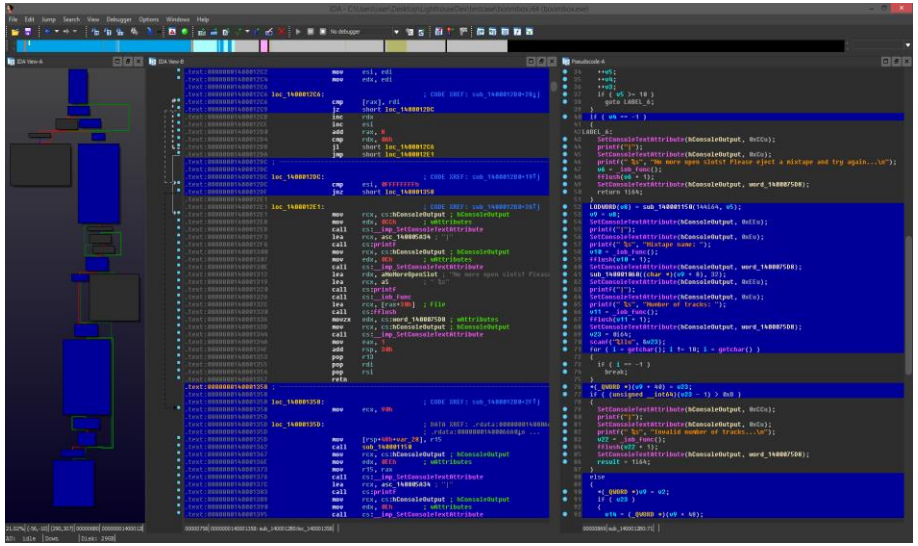
Visualization Examples



Fuzzle: Haeun Lee et al., 2022

```
        :
    1 :   if (fstat (STDOUT_FILENO, &stat_buf) < 0)
    0 :     die (EXIT_FAILURE, errno, _("standard output"));
        :
    1 :   outsize = io_blksize (stat_buf);
    1 :   out_dev = stat_buf.st_dev;
    1 :   out_ino = stat_buf.st_ino;
    1 :   out_isreg = S_ISREG (stat_buf.st_mode) != 0;
        :
    1 :   if (! (number || show_ends || squeeze_blank))
    0 :     {
        :       file_open_mode |= O_BINARY;
    0 :       if (O_BINARY && ! isatty (STDOUT_FILENO))
    0 :         xfreopen (NULL, "wb", stdout);
        :     }
    0 :
        :   /* Check if any of the input files are the same as the output file.  */
        :
        :   /* Main loop.  */
    0 :
    1 :   infile = "-";
    1 :   argind = optind;
        :
        :   do
        :     {
    1 :       if (argind < argc)
    1 :         infile = argv[argind];
        :
    1 :       if (STREQ (infile, "-"))
```

LCOV: Linux Test Project, 2001



LightHouse: Markus Gaasedelen, 2017

Lack of Static Analysis Visualization

- Visualizations of code coverage based on fuzzers or code execution are abundant, but less in static analysis

Lack of Static Analysis Visualization

- Visualizations of code coverage based on fuzzers or code execution are abundant, but less in static analysis

- Visualization of static analysis is also important for understanding static analysis

Static Analysis

- Symbolic Execution
  Qsym, Arbirter

Static Analysis

- Symbolic Execution
  Qsym, Arbirter

- Taint analysis
  SaTC, Karonte, Taint pipe, Code sonar

Static Analysis

- Symbolic Execution
  Qsym, Arbirter

- Taint analysis
  SaTC, Karonte, Taint pipe, Code sonar

- Concolic Execution
  Symsan

Static Analysis

- Symbolic Execution
  Qsym, Arbirter
- Taint analysis
  SaTC, Karonte, Taint pipe, Code sonar
- Concolic Execution
  Symsan

Taint Analysis

- Tracking the data flow from source to sink
  - ✓ source : origin of data
  - ✓ sink : dangerous function

```
void ping(char *source){
  char command[256];
  sprintf(command, "ping %s",source);
  system(command);
}
```

Taint Analysis

- Tracking the data flow from source to sink
  - ✓ source : origin of data
  - ✓ sink : dangerous function

```
void ping(char *source){
  char command[256];
  sprintf(command, "ping %s",source);
  system(command);
}
```

Taint Analysis

- Tracking the data flow from source to sink
  - ✓ source : origin of data
  - ✓ sink : dangerous function

```
void ping(char *source){
  char command[256];
  sprintf(command, "ping %s",source);
  system(command);
}
```

Taint Analysis

- Tracking the data flow from source to sink
  - ✓ source : origin of data
  - ✓ sink : dangerous function

```
void ping(char *source){
  char command[256];
  sprintf(command, "ping %s",source);
  system(command);
}
```

Taint Analysis

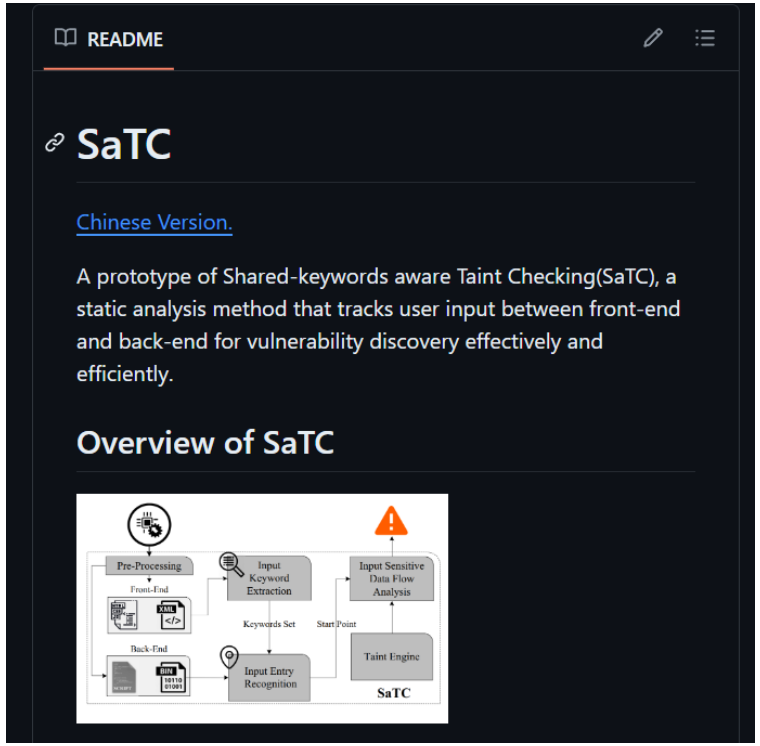
- Tracking the data flow from source to sink
  - ✓ source : origin of data
  - ✓ sink : dangerous function

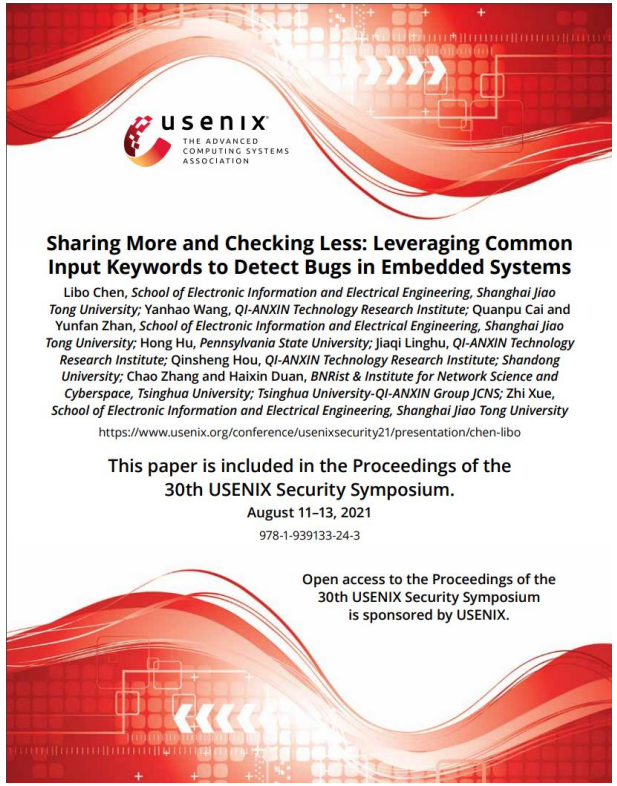
Taint Analysis

- Tracking the data flow from source to sink
  - ✓ source : origin of data
  - ✓ sink : dangerous function

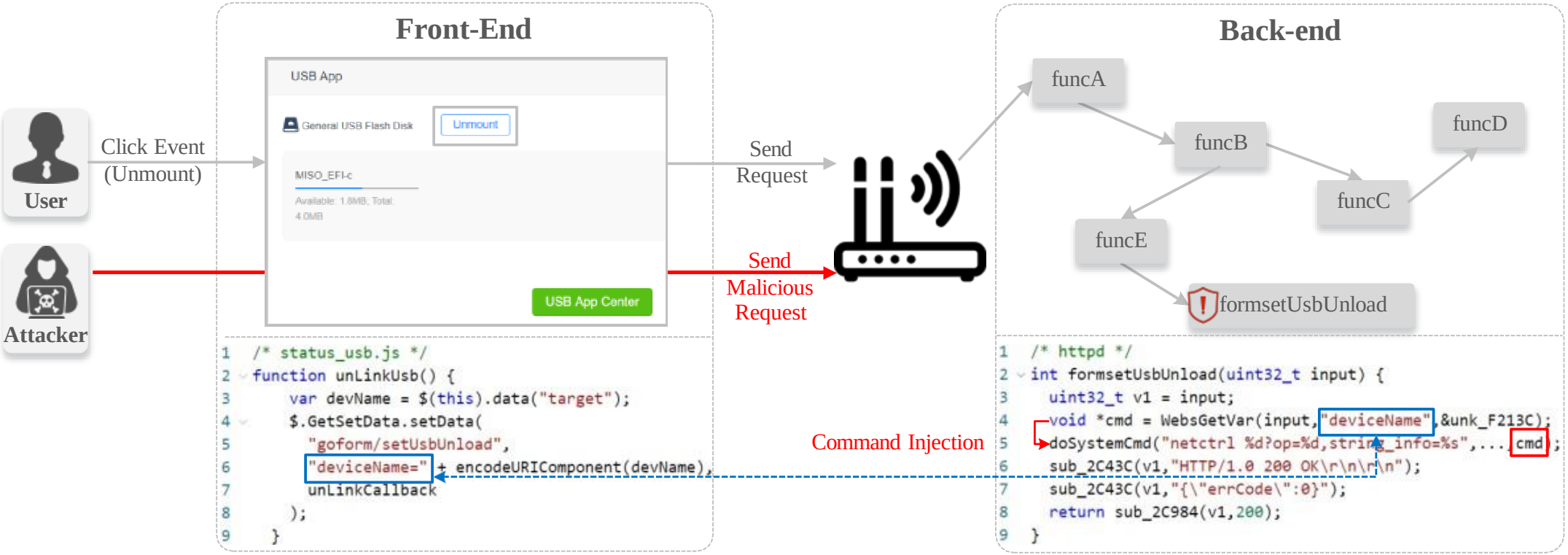
Applying Taint Analysis

- Bug finding in firmware through taint analysis
  - ➢ SaTC





Applying Taint Analysis



Malicious Request: http://IP:Port/goform/setUsbUnload?deviceName=***evalCMD***

Applying Taint Analysis

- Too much time consumption
- I thought there must have been some implementation overhead that caused this issue

Table 5: Compared with KARONTE on its dataset. For each vendor we report the device series, the number of firmware samples, the average analysis time (hour), the total number of alerts (#Alert) and the total number of true positives (#TP).

Vendor	Device Series	#Samples	KARONTE			SaTC		
			#Alerts	#TP	Time	#Alerts	#TP	Time
Netgear	R/XR/WNR	17	36	23	17:13 h	1,901	537	16:47 h
D-Link	DIR/DWR/DCS	9	24	15	14:09 h	32	22	1:57 h
TP-Link	TD/WA/WR/TX/KC	16	2	2	1:30 h	7	2	4:13 h
Tenda	AC/WH/FH	7	12	6	1:01 h	144	122	12:19 h
Total	-	49	74	46	33:57 h	2,084	683	35:16 h

Try to visualize taint propation and coverage then observe it

Visualization of Coverage

- Check which part of the code section taint engine has visited
- Display a heatmap showing which parts have been analyzed extensively.

```
    v4 = atoi((const char *)size);
    cfg->size = v4;
    v5 = atoi((const char *)pro_ver);
    cfg->pro_version = v5;
    v6 = atoi((const char *)timeout);
    cfg->timeout = v6;
    if ( cmd_get_ping_output(cfg, res_buf, 4096) )
    {
      printf("get result error! cmd=%s\n", (const char *)hostname);
      free(cfg);
    }
    else
    {
      free(cfg);
      if ( strstr((const char *)res_buf, "ttl") )
      {
        sscanf((const char *)res_buf, "%*[^=]=%*[^ ] %*[^=]=%[^ ] %*[^=]=%[^ ]", ttl, PingTime);
      }
      else
      {
        strcpy((char *)res_buf, "host is unreachable!");
        strcpy((char *)PingTime, "-1");
      }
      root = cJSON_CreateObject();
      String = cJSON_CreateString(hostname);
      cJSON_AddItemToObject(root, "ipAddr", String);
      v8 = cJSON_CreateString(ttl);
      cJSON_AddItemToObject(root, "timeToLive", v8);
      v9 = cJSON_CreateString(PingTime);
      cJSON_AddItemToObject(root, "pingtime", v9);
      out = cJSON_Print(root);
      cJSON_Delete(root);
      outputToWebs(wp, out);
      if ( out )
        free(out);
    }
  }
  else
  {
    printf(
      "Error->%s: %s(%d)--malloc failed!\n",
      "/home/work/workspace/EROS_Maintain/prod/httpd/X3/cgi/system_management.c",
      "formSetNetCheckTools",
      1031);
  }
}
000A1498 formSetNetCheckTools:36 (A9498) (Synchronized with IDA View-A, Hex View-1)
```

Less Visited

Too Much Visited

Out of Analyzed Scope

Visualization of Coverage

- Visualizing where taint propagation occurred in the code
- Can assess the advantages and disadvantages of taint propagation rule

```
if ( cfg )
{
  memset(cfg, 0, sizeof(CMD_PING_CFG_STRU));
  hostname = websGetVar(wp, "hostName", "192.168.10.1");
  count = websGetVar(wp, "packageNum", "3");
  size = websGetVar(wp, "packageSize", "56");
  pro_ver = websGetVar(wp, "pro_ver", "4");
  timeout = websGetVar(wp, "timeout", "1");
  v3 = strlen((const char *)hostname);
  strncpy((char *)cfg->hostname, (const char *)hostname, v3);
  cfg->count = 1;
  v4 = atoi((const char *)size);
  cfg->size = v4;
  v5 = atoi((const char *)pro_ver);
  cfg->pro_version = v5;
  v6 = atoi((const char *)timeout);
  if ( cmd_get_ping_output(cfg, res_buf, 4096) )
    printf("get result error! cmd=%s\n", (const char *)hostname);
    free(cfg);
  }
```

```
UGW_RETURN_CODE_ENUM __cdecl cmd_get_ping_output(const CMD_PING_CFG_STRU *cfg, unsigned __int8 *output, int o_size)
{
  unsigned __int8 new_cmd_buf[256]; // [sp+20h] [bp-10Ch] BYREF
  int read_len; // [sp+120h] [bp-Ch]
  FILE *fp; // [sp+124h] [bp-8h]

  fp = 0;
  memset(new_cmd_buf, 0, sizeof(new_cmd_buf));
  read_len = 0;
  if ( !cfg )
    return 1;
  snprintf(
    (char *)new_cmd_buf,
    0x100u,
    "ping %s -%d -c %d -s %d -W %d",
    (const char *)cfg->hostname,
    cfg->pro_version,
    cfg->count,
    cfg->size,
    cfg->timeout);
  fp = popen((const char *)new_cmd_buf, "r");
  if ( fp )
  {
    read_len = fread(output, 1u, o_size - 1, fp);
    pclose(fp);
    return 0;
  }
  else
  {
    printf("invalid cmd [%s]\n", (const char *)new_cmd_buf);
    return 1;
  }
}
```

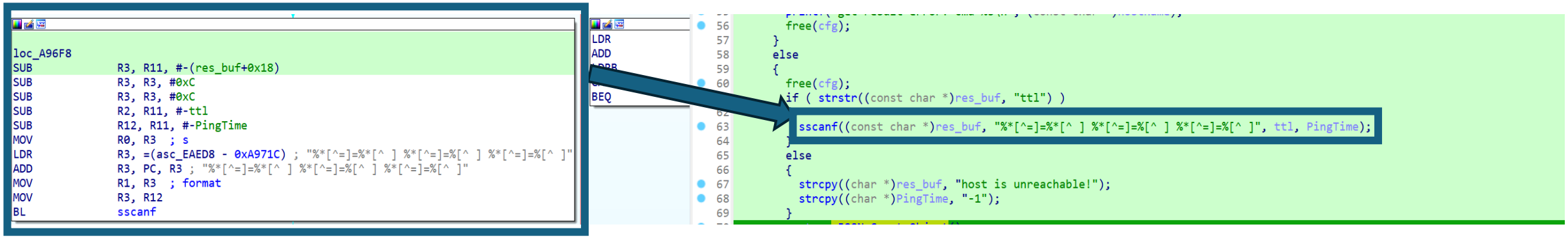
Demo Time

Challenges in Implementing Visualization

- C1: Matching the address with the corresponding pseudocode line
- C2: Identifying the argument and return value name in Decompiler
- C3: The differences in basic block boundary between angr and IDA
- C4: Analyzing the SaTC implement

C1: Matching the address with the corresponding pseudocode line

- Matching the address with the corresponding pseudocode line
- Using the “C item” that is an internal representation used by the decompiler to express each part of the analyzed code



C2: Identifying the argument and return value name in Decompiler

- IDA Decompiler API doesn’t support identify the function’s argument name
- Finding names traversing the Abstract Syntax Tree (AST)

```
hostname = websGetVar(wp, "hostName", "192.168.10.1");

return value name : hoastname
arg1 : wp
arg2 : "hostName"
arg3 : "192.168.10.1"
```

C3: The differences in Basic Block boundary between angr and IDA

- Taint Log cannot match the basic block correctly
- Logging the **CALL/JMP** instruction address instead of **MOV** instruction (example)

Paint incorrectly

```
v3 = strlen(hostname);
    LDR             R3, [R11,#cfg]
    ADD             R4, R3, #0x10
    LDR             R0, [R11,#hostname] ; s

strncpy(cfg->hostname, hostname, v3);
    MOV             R3, R0
    MOV             R0, R4  ; dest
    LDR             R1, [R11,#hostname] ; src
    MOV             R2, R3  ; n
    BL              strncpy
```

Angr Basic Block Boundary

```
v3 = strlen(hostname);
    LDR             R3, [R11,#cfg]
    ADD             R4, R3, #0x10
    LDR             R0, [R11,#hostname] ; s
    BL              strlen
    MOV             R3, R0

strncpy(cfg->hostname, hostname, v3);
    MOV             R0, R4  ; dest
    LDR             R1, [R11,#hostname] ; src
    MOV             R2, R3  ; n
    BL              strncpy
```

IDA Basic Block Boundary

C3: The differences in Basic Block boundary between angr and IDA

- Taint Log cannot match the basic block correctly
- Logging the **CALL/JMP** instruction address instead of **MOV** instruction (example)

Paint incorrectly

```
v3 = strlen(hostname);
    LDR             R3, [R11,#cfg]
    ADD             R4, R3, #0x10
    LDR             R0, [R11,#hostname] ; s

strncpy(cfg->hostname, hostname, v3);
    MOV             R3, R0
    MOV             R0, R4  ; dest
    LDR             R1, [R11,#hostname] ; src
    MOV             R2, R3  ; n
    BL              strncpy
```

Angr Basic Block Boundary

```
v3 = strlen(hostname);
    LDR             R3, [R11,#cfg]
    ADD             R4, R3, #0x10
    LDR             R0, [R11,#hostname] ; s
    BL              strlen
    MOV             R3, R0

strncpy(cfg->hostname, hostname, v3);
    MOV             R0, R4  ; dest
    LDR             R1, [R11,#hostname] ; src
    MOV             R2, R3  ; n
    BL              strncpy
```

IDA Basic Block Boundary

C3: The differences in Basic Block boundary between angr and IDA

- Taint Log cannot match the basic block correctly
- Logging the **CALL/JMP** instruction address instead of **MOV** instruction (example)

Paint incorrectly

```
v3 = strlen(hostname);
    LDR             R3, [R11,#cfg]
    ADD             R4, R3, #0x10
    LDR             R0, [R11,#hostname] ; s

strncpy(cfg->hostname, hostname, v3);
    MOV             R3, R0
    MOV             R0, R4  ; dest
    LDR             R1, [R11,#hostname] ; src
    MOV             R2, R3  ; n
    BL              strncpy
```

Angr Basic Block Boundary

```
v3 = strlen(hostname);
    LDR             R3, [R11,#cfg]
    ADD             R4, R3, #0x10
    LDR             R0, [R11,#hostname] ; s
    BL              strlen
    MOV             R3, R0

strncpy(cfg->hostname, hostname, v3);
    MOV             R0, R4  ; dest
    LDR             R1, [R11,#hostname] ; src
    MOV             R2, R3  ; n
    BL              strncpy
```

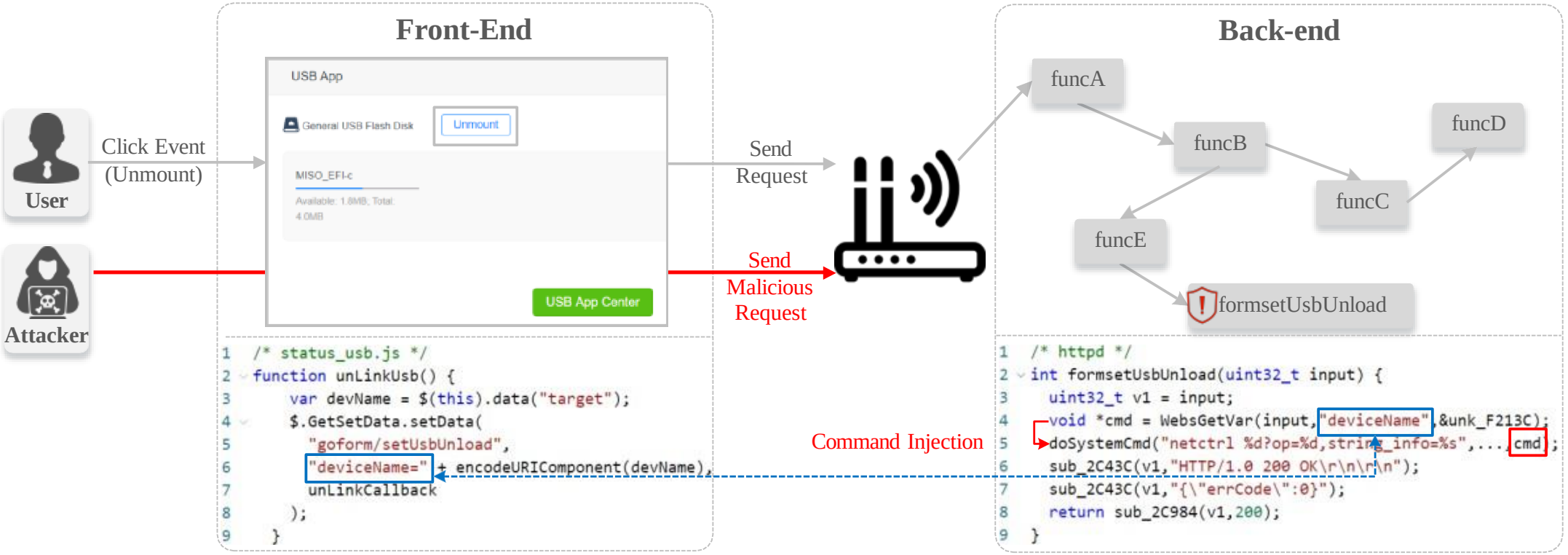
IDA Basic Block Boundary

C4: Analyzing the SaTC implement

- SaTC implemented based on the Angr
- I am familiar with angr, and it took some time, but it wasn’t bad
- As I told you, the interesting thing is Under Constrained Symbolic Execution that allows analysis starting from a desired point

Under Constrained Symbolic Execution

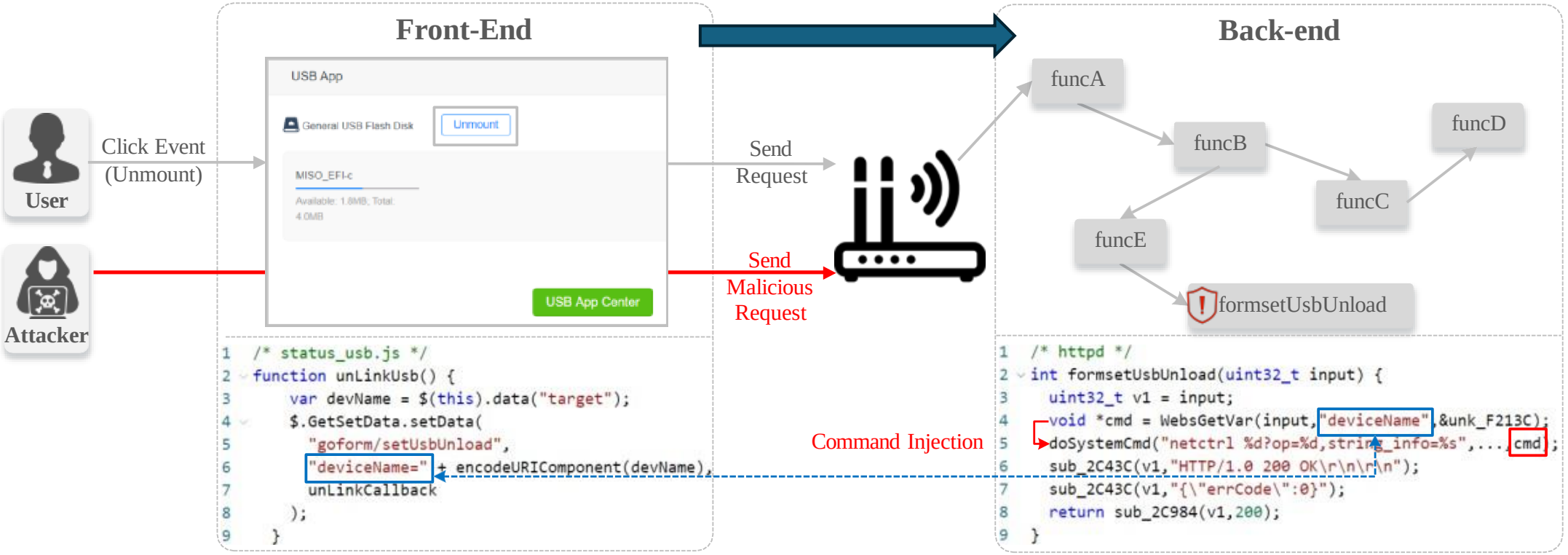
- UCSE(Under Constrained Symbolic Execution) allows analysis starting from a desired point



Malicious Request: http://IP:Port/goform/setUsbUnload?deviceName=***evalCMD***

Under Constrained Symbolic Execution

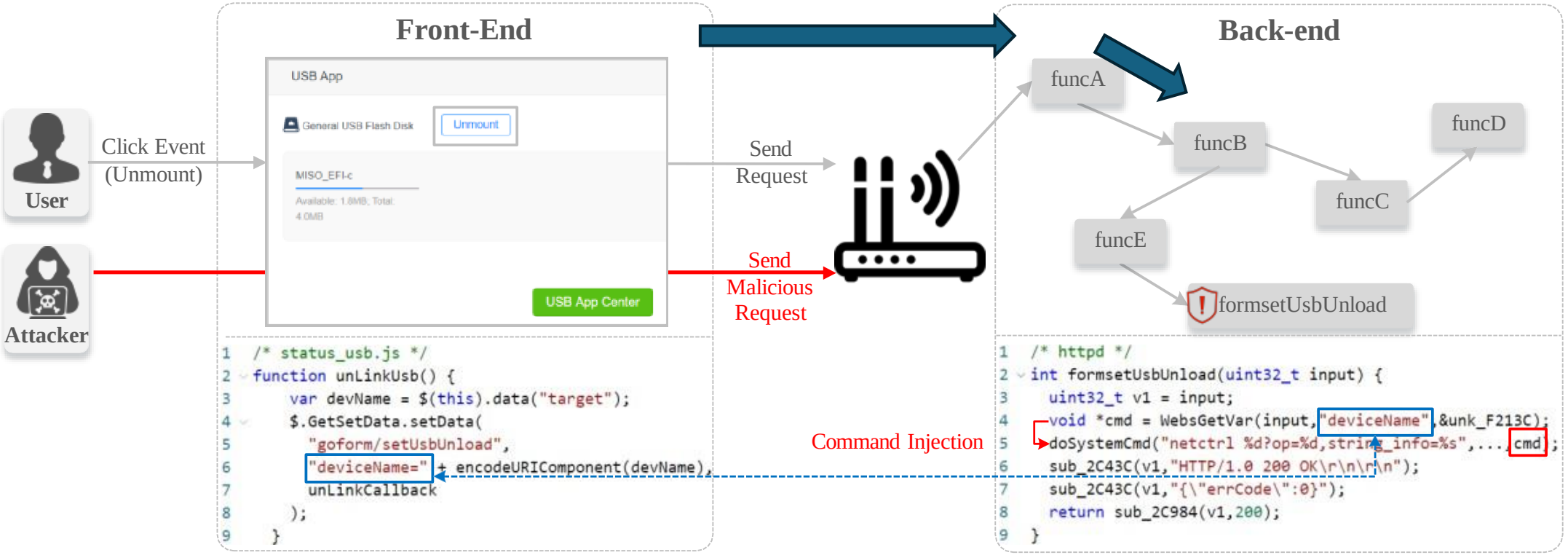
- UCSE(Under Constrained Symbolic Execution) allows analysis starting from a desired point



Malicious Request: http://IP:Port/goform/setUsbUnload?deviceName=***evalCMD***

Under Constrained Symbolic Execution

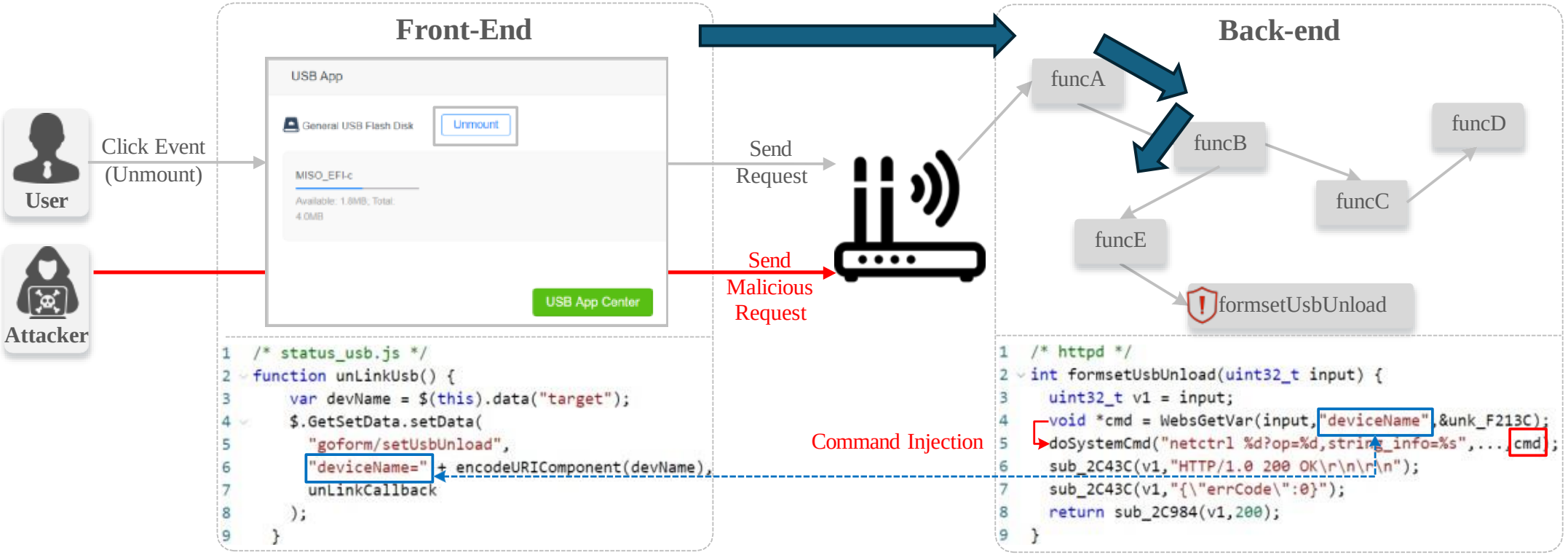
- UCSE(Under Constrained Symbolic Execution) allows analysis starting from a desired point



Malicious Request: http://IP:Port/goform/setUsbUnload?deviceName=***evalCMD***

Under Constrained Symbolic Execution

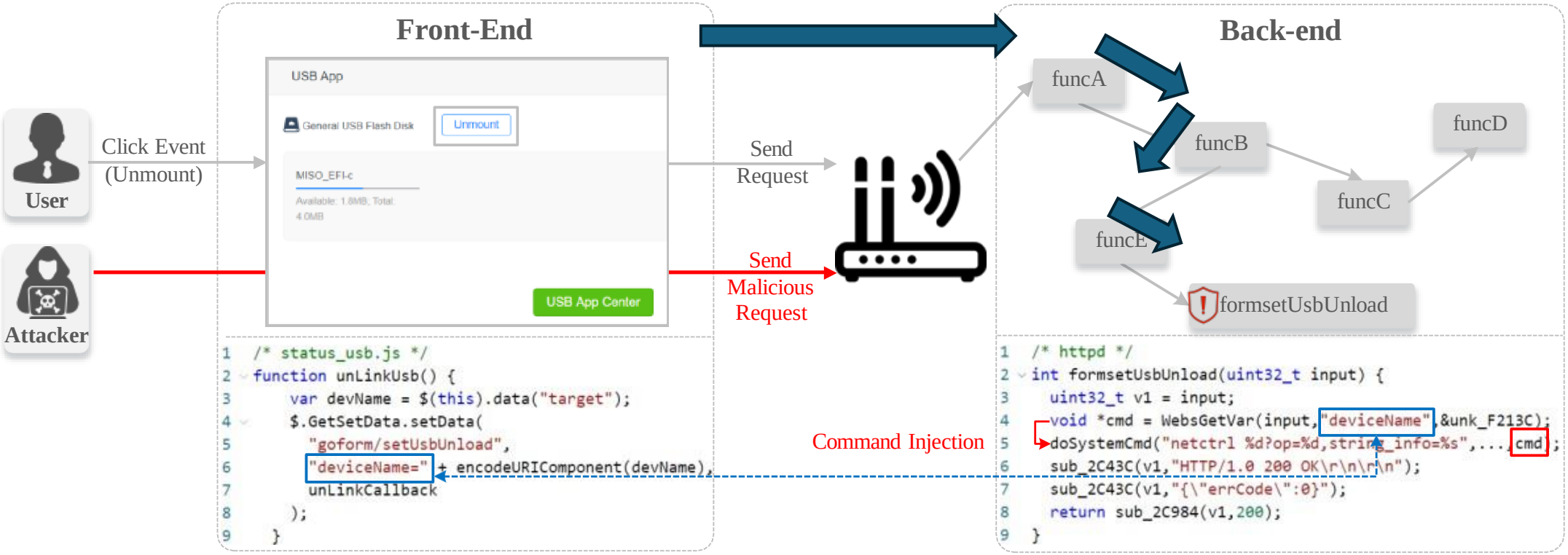
- UCSE(Under Constrained Symbolic Execution) allows analysis starting from a desired point



Malicious Request: http://IP:Port/goform/setUsbUnload?deviceName=***evalCMD***

Under Constrained Symbolic Execution

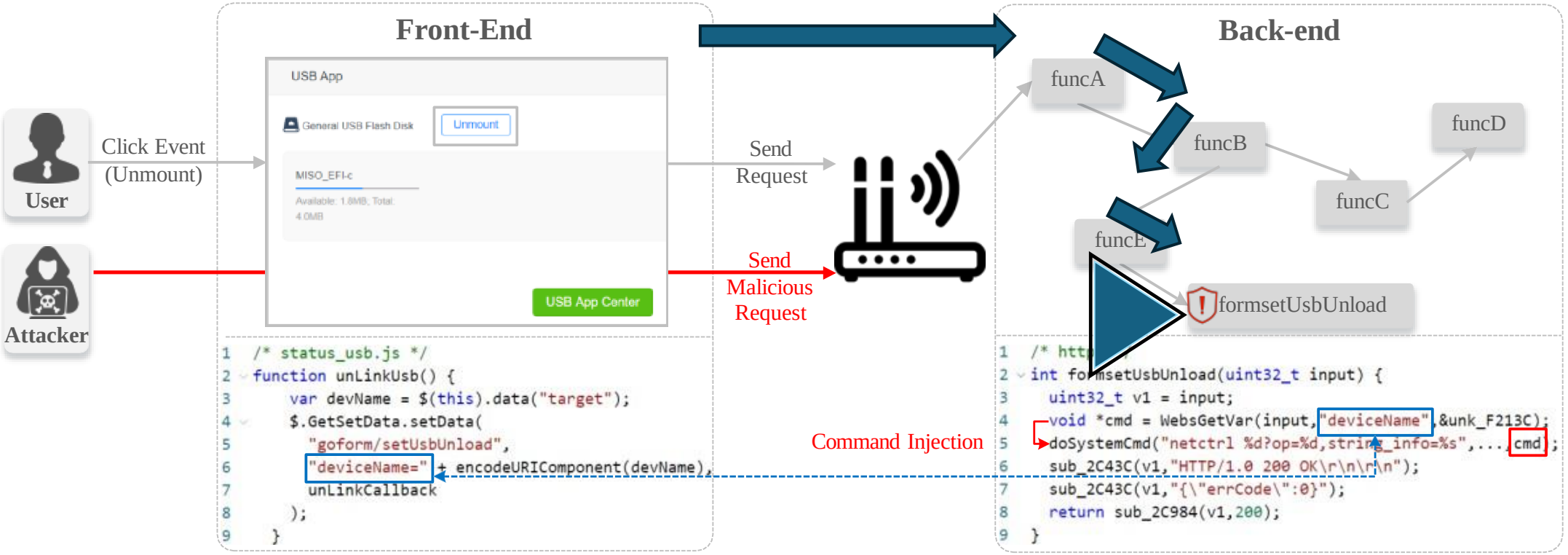
- UCSE(Under Constrained Symbolic Execution) allows analysis starting from a desired point



Malicious Request: http://IP:Port/goform/setUsbUnload?deviceName=***evalCMD***

Under Constrained Symbolic Execution

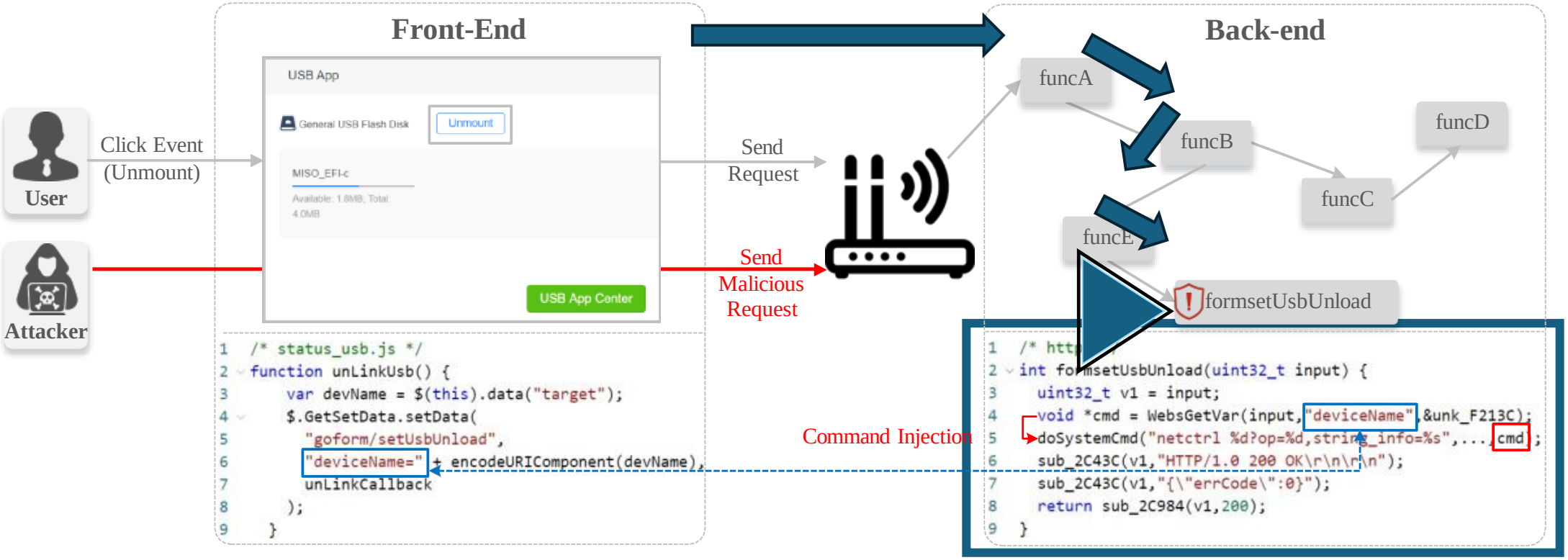
- UCSE(Under Constrained Symbolic Execution) allows analysis starting from a desired point



Malicious Request: http://IP:Port/goform/setUsbUnload?deviceName=***evalCMD***

Under Constrained Symbolic Execution

- UCSE(Under Constrained Symbolic Execution) allows analysis starting from a desired point



Malicious Request: http://IP:Port/goform/setUsbUnload?deviceName=***evalCMD***

Evaluation and Observation

- TaintChecker tested on SaTC dataset
- It was tested on 38 firmware filesystems from 5 different vendors.
- Among them, I introduce some interesting case

Evaluation and Observation

- SaTC cannot detect some CMDI

```
15  GetValue((int)"wl.guest.dhcps_ifname", (int)sMibValue);
16  snprintf((char *)sWifiGuestIfName, 0x10u, "%s", (const char *)sMibValue);
17  sMibValue[0] = 0;
18  GetValue((int)"wans.flag", (int)sMibValue);
19  iWanNum = atoi((const char *)sMibValue);
20  for ( iIndex = 0; iIndex /*signed*/< iWanNum; ++iIndex )
21  {
22    getWanIfName(iIndex + 1, sWanIfName);
23    if ( !ifaddrs_get_ifip(sWanIfName, sMibValue) )
24    {
25      deleteWifiGuestNatRule(iIndex + 1);
26      if ( enable == 1 )
27        addWifiGuestNatRule(iIndex + 1, sWanIfName, sWifiGuestIfName);// There is are cmdi using "sWifiGuestIfName".
28                                                  // Concrete Value is from "sMibValue"
29    }
30  }
31 }
```

Evaluation and Observation

- SaTC cannot detect some CMDI

Source of Command Injection Bugs

```
15  GetValue((int)"wl.guest.dhcps_ifname", (int)sMibValue);
16  snprintf((char *)sWifiGuestIfName, 0x10u, "%s", (const char *)sMibValue);
17  sMibValue[0] = 0;
18  GetValue((int)"wans.flag", (int)sMibValue);
19  iWanNum = atoi((const char *)sMibValue);
20  for ( iIndex = 0; iIndex /*signed*/< iWanNum; ++iIndex )
21  {
22    getWanIfName(iIndex + 1, sWanIfName);
23    if ( !ifaddrs_get_ifip(sWanIfName, sMibValue) )
24    {
25      deleteWifiGuestNatRule(iIndex + 1);
26      if ( enable == 1 )
27        addWifiGuestNatRule(iIndex + 1, sWanIfName, sWifiGuestIfName);// There is are cmdi using "sWifiGuestIfName".
28                                                  // Concrete Value is from "sMibValue"
29    }
30  }
31 }
```

Evaluation and Observation

- SaTC cannot detect some CMDI

Source of Command Injection Bugs

```
15  GetValue((int)"wl.guest.dhcps_ifname", (int)sMibValue);
16  snprintf((char *)sWifiGuestIfName, 0x10u, "%s", (const char *)sMibValue);
17  sMibValue[0] = 0;
18  GetValue((int)"wans.flag", (int)sMibValue);
19  iWanNum = atoi((const char *)sMibValue);
20  for ( iIndex = 0; iIndex /*signed*/< iWanNum; ++iIndex )
21  {
22    getWanIfName(iIndex + 1, sWanIfName);
23    if ( !ifaddrs_get_ifip(sWanIfName, sMibValue) )
24    {
25      deleteWifiGuestNatRule(iIndex + 1);
26      if ( enable == 1 )
27        addWifiGuestNatRule(iIndex + 1, sWanIfName, sWifiGuestIfName);// There is are cmdi using "sWifiGuestIfName".
28                                                    // Concrete Value is from "sMibValue"
29    }
30  }
31 }
```

Comman Injection Bugs in this function

Evaluation and Observation

- SaTC cannot detect some CMDI

Source of Command Injection Bugs

SaTC cannot locate this source correctly

Evaluation and Observation

- It performs analysis even on parts that are not tainted.
- Do not require dataflow tracking, wasting time on unnecessary sections

Pseudocode-A

```
30  networkTool = websGetVar(wp, "networkTool", "1");
31  NetType = atoi((const char *)networkTool);
32  if ( NetType == 1 )
33  {
34    cfg = 0;
35    cfg = (CMD_PING_CFG_STRU *)malloc(0x50u);
36    if ( cfg )
37    {
38      memset(cfg, 0, sizeof(CMD_PING_CFG_STRU));
39      hostname = websGetVar(wp, "hostName", "192.168.10.1");
40      count = websGetVar(wp, "packageNum", "3");
41      size = websGetVar(wp, "packageSize", "56");
42      pro_ver = websGetVar(wp, "pro_ver", "4");
43      timeout = websGetVar(wp, "timeout", "1");
44      v3 = strlen((const char *)hostname);
45      strncpy((char *)cfg->hostname, (const char *)hostname, v3);
46      cfg->count = 1;
47      v4 = atoi((const char *)size);
48      cfg->size = v4;
49      v5 = atoi((const char *)pro_ver);
50      cfg->pro_version = v5;
51      v6 = atoi((const char *)timeout);
52      cfg->timeout = v6;
53      if ( cmd_get_ping_output(cfg, res_buf, 4096) )
54      {
55        printf("get result error! cmd=%s\n", (const char *)hostname);
56        free(cfg);
57      }
58      else
59      {
60        free(cfg);
61        if ( strstr((const char *)res_buf, "ttl") )
62        {
63          sscanf((const char *)res_buf, "%*[^=]=%*[^ ] %*[^=]=%[^ ] %*[^=]=%[^ ]", ttl, PingTime);
64        }
65        else
66        {
67          strcpy((char *)res_buf, "host is unreachable!");
68          strcpy((char *)PingTime, "-1");
69        }
70        root = cJSON_CreateObject();
71        String = cJSON_CreateString(hostname);
72        cJSON_AddItemToObject(root, "ipAddr", String);
73        v8 = cJSON_CreateString(ttl);
74        cJSON_AddItemToObject(root, "timeToLive", v8);
75        v9 = cJSON_CreateString(PingTime);
76        cJSON_AddItemToObject(root, "pingtime", v9);
77        out = cJSON_Print(root);
78        cJSON_Delete(root);
79        outputToWebs(wp, out);
80        if ( out )
81          free(out);
82      }
83    }
84    else
85    {
86      printf(
87        "Error->%s: %s(%d)--malloc failed!\n",
88        "/home/work/workspace/EROS_Maintain/prod/httpd/X3/cgi/system_management.c",
89        "formSetNetCheckTools",
90        1031);
91    }
```

Evaluation and Observation

- It performs analysis even on parts that are not tainted.
- Do not require dataflow tracking, wasting
section

This is interesting field

Pseudocode-A

```
30  networkTool = websGetVar(wp, "networkTool", "1");
31  NetType = atoi((const char *)networkTool);
32  if ( NetType == 1 )
33  {
34    cfg = 0;
35    cfg = (CMD_PING_CFG_STRU *)malloc(0x50u);
36    if ( cfg )
37    {
38      memset(cfg, 0, sizeof(CMD_PING_CFG_STRU));
39      hostname = websGetVar(wp, "hostName", "192.168.10.1");
40      count = websGetVar(wp, "packageNum", "3");
41      size = websGetVar(wp, "packageSize", "56");
42      pro_ver = websGetVar(wp, "pro_ver", "4");
43      timeout = websGetVar(wp, "timeout", "1");
44      v3 = strlen((const char *)hostname);
45      strncpy((char *)cfg->hostname, (const char *)hostname, v3);
46      cfg->count = 1;
47      v4 = atoi((const char *)size);
48      cfg->size = v4;
49      v5 = atoi((const char *)pro_ver);
50      cfg->pro_version = v5;
51      v6 = atoi((const char *)timeout);
52      cfg->timeout = v6;
53      if ( cmd_get_ping_output(cfg, res_buf, 4096) )
54      {
55        printf("get result error! cmd=%s\n", (const char *)hostname);
56        free(cfg);
57      }
58      else
59      {
60        free(cfg);
61        if ( strstr((const char *)res_buf, "ttl") )
62        {
63          sscanf((const char *)res_buf, "%*[^=]=%*[^ ] %*[^=]=%[^ ] %*[^=]=%[^ ]", ttl, PingTime);
64        }
65        else
66        {
67          strcpy((char *)res_buf, "host is unreachable!");
68          strcpy((char *)PingTime, "-1");
69        }
70        root = cJSON_CreateObject();
71        String = cJSON_CreateString(hostname);
72        cJSON_AddItemToObject(root, "ipAddr", String);
73        v8 = cJSON_CreateString(ttl);
74        cJSON_AddItemToObject(root, "timeToLive", v8);
75        v9 = cJSON_CreateString(PingTime);
76        cJSON_AddItemToObject(root, "pingtime", v9);
77        out = cJSON_Print(root);
78        cJSON_Delete(root);
79        outputToWebs(wp, out);
80        if ( out )
81          free(out);
82      }
83    }
84    else
85    {
86      printf(
87        "Error->%s: %s(%d)--malloc failed!\n",
88        "/home/work/workspace/EROS_Maintain/prod/httpd/X3/cgi/system_management.c",
89        "formSetNetCheckTools",
90        1031);
91    }
```

Evaluation and Observation

- It performs analysis even on parts that are not tainted.
- Do not require dataflow tracking, wasting
section

This is interesting field

This is not interesting field, but spending much time

Pseudocode-A

```
30  networkTool = websGetVar(wp, "networkTool", "1");
31  NetType = atoi((const char *)networkTool);
32  if ( NetType == 1 )
33  {
34    cfg = 0;
35    cfg = (CMD_PING_CFG_STRU *)malloc(0x50u);
36    if ( cfg )
37    {
38      memset(cfg, 0, sizeof(CMD_PING_CFG_STRU));
39      hostname = websGetVar(wp, "hostName", "192.168.10.1");
40      count = websGetVar(wp, "packageNum", "3");
41      size = websGetVar(wp, "packageSize", "56");
42      pro_ver = websGetVar(wp, "pro_ver", "4");
43      timeout = websGetVar(wp, "timeout", "1");
44      v3 = strlen((const char *)hostname);
45      strncpy((char *)cfg->hostname, (const char *)hostname, v3);
46      cfg->count = 1;
47      v4 = atoi((const char *)size);
48      cfg->size = v4;
49      v5 = atoi((const char *)pro_ver);
50      cfg->pro_version = v5;
51      v6 = atoi((const char *)timeout);
52      cfg->timeout = v6;
53      if ( cmd_get_ping_output(cfg, res_buf, 4096) )
54      {
55        printf("get result error! cmd=%s\n", (const char *)hostname);
56        free(cfg);
57      }
58      else
59      {
60        free(cfg);
61        if ( strstr((const char *)res_buf, "ttl") )
62        {
63          sscanf((const char *)res_buf, "%*[^=]=%*[^ ] %*[^=]=%[^ ] %*[^=]=%[^ ]", ttl, PingTime);
64        }
65        else
66        {
67          strcpy((char *)res_buf, "host is unreachable!");
68          strcpy((char *)PingTime, "-1");
69        }
70        root = cJSON_CreateObject();
71        String = cJSON_CreateString(hostname);
72        cJSON_AddItemToObject(root, "ipAddr", String);
73        v8 = cJSON_CreateString(ttl);
74        cJSON_AddItemToObject(root, "timeToLive", v8);
75        v9 = cJSON_CreateString(PingTime);
76        cJSON_AddItemToObject(root, "pingtime", v9);
77        out = cJSON_Print(root);
78        cJSON_Delete(root);
79        outputToWebs(wp, out);
80        if ( out )
81          free(out);
82      }
83    }
84    else
85    {
86      printf(
87        "Error->%s: %s(%d)--malloc failed!\n",
88        "/home/work/workspace/EROS_Maintain/prod/httpd/X3/cgi/system_management.c",
89        "formSetNetCheckTools",
90        1031);
91    }
```

Evaluation and Observation

- It performs analysis even on parts that are not tainted.
- Do not require dataflow tracking, wasting time on unnecessary sections

```
  v1 = "0";
  nptr = v1;
  v12 = (char *)sub_2BA8C(a1, "list", &unk_F3538);
  v11 = (char *)sub_2BA8C(a1, "vlanId", &unk_F3538);
  v10 = (char *)sub_2BA8C(a1, "iptvType", &unk_F3538);
  GetValue("adv.iptv.stballvlans", s);
  GetValue("adv.iptv.stbpvid", s2);
  GetValue("iptv.city.vlan", v7);
  GetValue("iptv.stb.enable", v6);
  if ( strcmp(s1, v6) || strcmp(v12, s) || strcmp(v11, s2) || strcmp(v10, v7) )
    v15 = 1;
  if ( atoi(nptr) == 1 || atoi(s1) == 1 )
  {
    SetValue("iptv.enable", "1");
    doSystemCmd("nvram set iptv.enable=1");
  }
  else
  {
    SetValue("iptv.enable", "0");
    doSystemCmd("nvram set iptv.enable=0");
  }
  SetValue("iptv.city.vlan", v10);
  if ( atoi(nptr) == 1 )
  {
    sub_AFEDC();
    sprintf(v5, "op=%d", 1);
  }
  else
  {
    sub_AFF18();
    sprintf(v5, "op=%d", 2);
  }
  if ( atoi(s1) == 1 )
    sub_AFF54(v12, v11, v10);
  else
    sub_B0140(v12, v11, v10);
  doSystemCmd("nvram commit");
  v2 = printf("[he debug]:%s,%d--nvram commit ok!\n", "formSetIptv", 193);
  if ( CommitCfm(v2) )
    send_msg_to_netctrl(26, v5);
  else
    v16 = 1;
  sub_2C40C(
    a1,
    "HTTP/1.1 200 OK\nContent-type: text/plain; charset=utf-8\nPragma: no-cache\nCache-Control: no-cache\n\n");
  sub_2C40C(a1, "{\"errCode\":%d}", v16);
  result = sub_2C954(a1, 200);
  if ( v15 )
    return tpi_systool_handle(0);
  return result;
}
```

Evaluation and Observation

- It performs analysis even on parts that are not tainted.
- Do not require dataflow tracking, wasting
section

Cannot Visit this Block

```
  v1 = "0";
  nptr = v1;
  v12 = (char *)sub_2BA8C(a1, "list", &unk_F3538);
  v11 = (char *)sub_2BA8C(a1, "vlanId", &unk_F3538);
  v10 = (char *)sub_2BA8C(a1, "iptvType", &unk_F3538);
  GetValue("adv.iptv.stballvlans", s);
  GetValue("adv.iptv.stbpvid", s2);
  GetValue("iptv.city.vlan", v7);
  GetValue("iptv.stb.enable", v6);
  if ( strcmp(s1, v6) || strcmp(v12, s) || strcmp(v11, s2) || strcmp(v10, v7) )
    v15 = 1;
  if ( atoi(nptr) == 1 || atoi(s1) == 1 )
  {
    SetValue("iptv.enable", "1");
    doSystemCmd("nvram set iptv.enable=1");
  }
  else
  {
    SetValue("iptv.enable", "0");
    doSystemCmd("nvram set iptv.enable=0");
  }
  SetValue("iptv.city.vlan", v10);
  if ( atoi(nptr) == 1 )
  {
    sub_AFEDC();
    sprintf(v5, "op=%d", 1);
  }
  else
  {
    sub_AFF18();
    sprintf(v5, "op=%d", 2);
  }
  if ( atoi(s1) == 1 )
    sub_AFF54(v12, v11, v10);
  else
    sub_B0140(v12, v11, v10);
  doSystemCmd("nvram commit");
  v2 = printf("[he debug]:%s,%d--nvram commit ok!\n", "formSetIptv", 193);
  if ( CommitCfm(v2) )
    send_msg_to_netctrl(26, v5);
  else
    v16 = 1;
  sub_2C40C(
    a1,
    "HTTP/1.1 200 OK\nContent-type: text/plain; charset=utf-8\nPragma: no-cache\nCache-Control: no-cache\n\n");
  sub_2C40C(a1, "{\"errCode\":%d}", v16);
  result = sub_2C954(a1, 200);
  if ( v15 )
    return tpi_systool_handle(0);
  return result;
}
```

Evaluation and Observation

- Taint Checker offer visualization interface
- You can easily apply it for other taint analysis tools

Taint analysis Tools
Ex) SaTC

→

Taint Checker
Interface

→

Conclusion

- I propose TaintChecker, a novel approach to visualize taint analysis on Decompiler
- Using novel approach, I found the inefficiency and limitation of SaTC
- TaintChecker can offer visualization interface to you. Then you can understand the taint analysis more deeply

Prototype code
https://github.com/5angjun/TaintChecker

Thank you

Questions?

Email: sangjuns@kaist.ac.kr