

John Toterhi

john@finitestate.io

@cetfor



Hunting for Backdoors in IoT Firmware at Unprecedented Scale

HITBSecConf Dubai
November 27, 2018

FINITE  STATE

Journey to Backdoor Discovery via Firmware Analysis

- 1. The Scale of Data & Duplication**
- 2. Backdoor Manifestations**
- 3. The Power of Correlation**
- 4. Source Code Analyzers**
- 5. Binary Analyzers**



1. The Scale of Data & Duplication



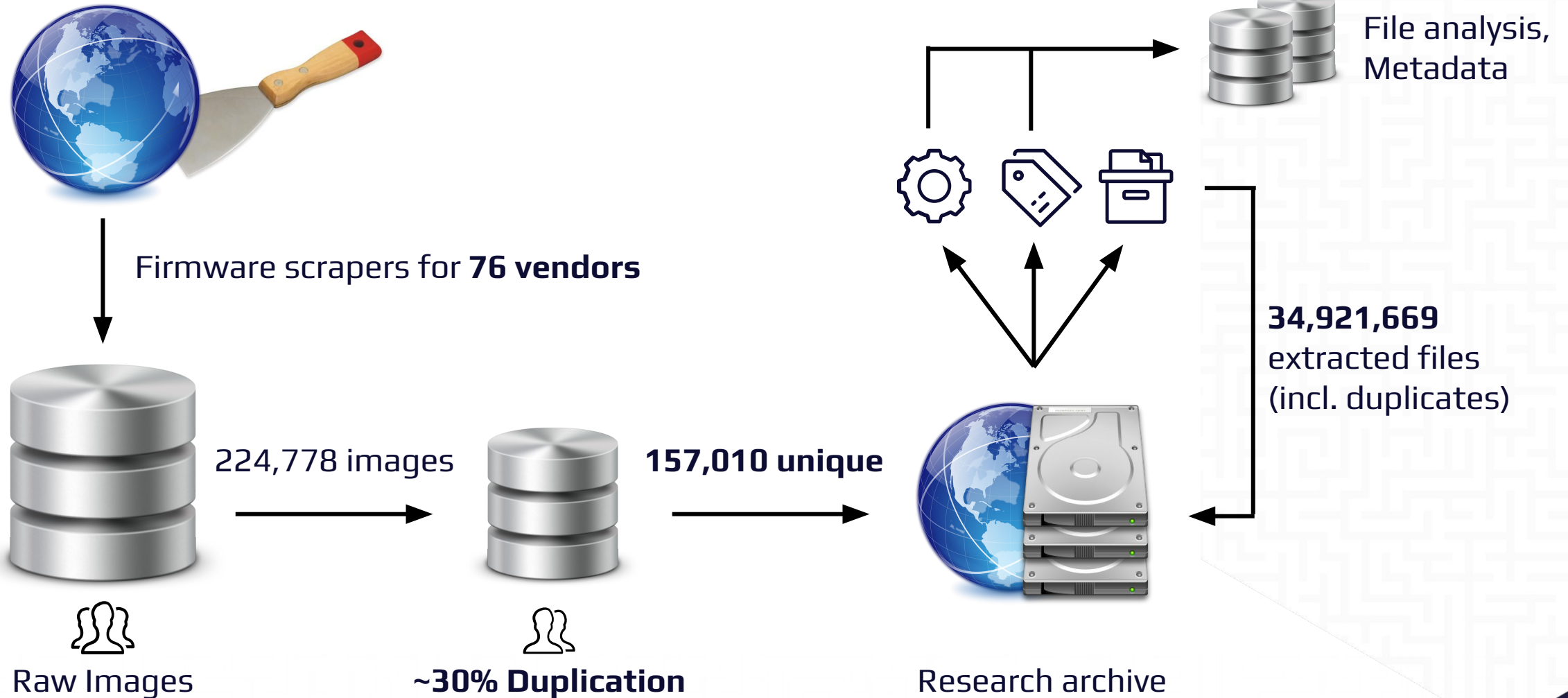
The Scale of D&D: Backdoors to Date

This effort lead to:

- Discovery of **4 verified** IoT backdoors (**75** unique devices)
- Discovery of **11 unverified** IoT backdoors (**107** unique devices)
- Backdoors exist in approximately **0.9 - 2.1% of analyzed IoT devices**



The Scale of D&D: Firmware Collection



The Scale of D&D: File Duplication Metrics

157,010 unique
firmware images
across **76** vendors



34,921,669
successful file
extractions (incl.
duplicates)

Select files types with attack surface significance

Executables:	1,474,686	(159,432 unique, 89.2% duplication)
Shared libs:	1,325,862	(144,721 unique, 89.1% duplication)
Python:	1,281,338	(8,635 unique, 99.3% duplication)
Shell scripts:	518,203	(13,800 unique, 97.3% duplication)
JavaScript:	261,394	(34,334 unique, 86.9% duplication)
Java applets:	188,340	(51,983 unique, 72.4% duplication)
PHP:	54,268	(8,159 unique, 85.0% duplication)

91.7% duplication
in these security-
significant files



The Scale of D&D: File Duplication Metrics

The other ~30 million files

- Audio files
- Binary blobs
- **Certificates, Key files**
- Configuration files
- Images (jpg, png, etc)
- Text files (license info, etc.)
- **Kernel objects (.ko)**
- Random scripts (lua, perl, etc.)
- Symlinks
- Web content (html, asp, etc)



2. Backdoor Manifestations



IoT Backdoor Manifestations: Juniper ScreenOS

- Telnet & SSH backdoor credentials in Juniper NetScreen firewall (ScreenOS)
- Password was “<<< %s(un='%s') = %u”, similar to surrounding strings
- Usable without a valid username

```

• ROM: 0013DC50      LDR      R0, =aSctUUnSSipSDip ; ">>> %s(ct=%u, un='%s',
• ROM: 0013DC54      LDR      R1, =aAuth_admin_int ; "auth_admin_internal"
• ROM: 0013DC58      BL       sub_558F74
ROM: 0013DC5C
ROM: 0013DC5C      loc_13DC5C
• ROM: 0013DC5C      ADD      R0, R5, #0x44          ; CODE XREF: auth_admin_internal+2C↑j
• ROM: 0013DC60      LDR      R1, =aSUnSU ; "<<< %s(un='%s') = %u"
• ROM: 0013DC64      BL       strcmp
• ROM: 0013DC68      CMP     R0, #0

```

JUNIPER
NETWORKS



IoT Backdoor Manifestations: DBL Tek GoIP

- `login` binary contains a challenge/response for the “dbladm” user (telnet)
- The user can compute the password based only on the challenge

```
Start login
do exec: /sbin/login
Login: dbladm
challenge: N2054086922
Password: _
```

$\text{md5}(\text{challenge} + 20139 + (\text{challenge} \gg 3))[0:6]$



IoT Backdoor Manifestations: Belkin F9K1102

- `dev.htm` file contains a debugging webshell
- Requests to the backend via `/dev.cgi?c=<cmd>` gives root access

```
> ?  
cfg sys sh  
> 
```

dev.htm interface

`nvram` from root shell

```
wl1_dispatch_mode=1  
wl0_rxchain_pwrsave_enable=1  
wl0_hw_rxchain=3  
http_passwd=[REDACTED]  
wl_wpa_psk=[REDACTED]  
wl_guest_passwd=[REDACTED]  
wl_rpcq_rxthresh=512  
dhcp_reserveMAC4=48:D2:24 [REDACTED]
```



IoT Backdoor Manifestations: WD My Cloud

- `network_mgr.cgi` (ARM binary) manages user sessions
- POST req. with **cmd=cgi_get_ipv6, flag=1** creates session tied to user IP
- Subsequent requests with Cookie data **username=admin** bypasses auth.

Triggering payload

```
POST /cgi-bin/network_mgr.cgi HTTP/1.1
Host: wdmycloud.local
Content-Type: application/x-www-form-urlencoded
Cookie: username=admin
Content-Length: 23

cmd=cgi_get_ipv6&flag=1
```

Western Digital®



IoT Backdoor Manifestations: Dahua IP Camera

- Backdoored **telnetd** in **`busybox`** (ARM) embedded linux “Swiss Army knife”
- Username: admin, Password begins with **7ujMko0**

```
162 @ 00011db0 r0_75 = strlen(0x6a59a) {"7ujMko0"}
163 @ 00011db8 int32_t r6_2 = r0_75
164 @ 00011dc0 void* r1_21 = &var_9c
165 @ 00011dc8 int32_t r2_10 = r6_2
166 @ 00011dcc r0_76 = strncmp(0x6a59a, r1_21, r2_10) {"7ujMko0"}
167 @ 00011dd0 int32_t r5_5 = r0_76 - 0
168 @ 00011dd4 if (r0_76 != 0) then 27 @ 0x12184 else 183 @ 0x11df0
```

Binary Ninja MLIL view



3. The Power of Correlation



The Power of Correlation

- **Cryptographic hashing** enables *deduplication*
- **Fuzzy hashing** enables *correlation*
 - ***ssdeep*** was designed to correlate corrupted image and video files
- For binary and source correlation, we use ***MRSH-CF***
 - Based on many years of AMA evolution; *ssdeep*, *sdhash*, *mrsh-v2*

Basic Properties ⓘ

MD5	a57b0d81081ee158d02a1b3ad4d20bb1
SHA-1	102e4a3f05d2e8b9de8c3fee844e1cf43746478f
File Type	Win32 EXE
Magic	PE32+ executable for MS Windows (GUI) Mono/.Net assembly
SSDeep	768:fUu7WleamRGpyysniU7byLzy9J3OI/qTTyvJGTSg7vo3Mi+1blucWJx4W4KxYRBF:feXayC9JgSivHJY1BBaxsyU7ZfVbiAP
File Size	63.2 KB



The Power of Correlation

	<i>File 1</i>	<i>File 2</i>	<i>SSDeep</i>	<i>MRSH-CF</i>
1	busybox (v1.18.4, mipsel)	busybox (v1.18.4, mipsel)	100%	100%
2	login.sh (EnGenius)	login.sh (WatchGuard)	96%	79.8%
3	asus_lighttpd (arm, 4G-AC55U)	asus_lighttpd (arm, RT-AC1900U)	0%	10%
4	busybox (v1.18.4, mipsel)	busybox (v1.19.0, mipsel)	0%	7.5%
5	wireless.so (RouterOS, mips)	wireless.so (RouterOS, arm)	0%	2.6%
6	lighttpd (mips, Ubiquiti nbm365)	libusb.so.4 (mips, Ubiquiti es-8xp)	0%	0%



The Power of Correlation: Telnet Jailbreak

- **Telnet jailbreak**, publicly identified in 6 devices (5 EnGenius, 1 Araknis)
- Found in *42 other devices* from **EnGenius**

Brand	Model	Firmware version	File	Match (%)
<input type="text" value="Filter"/>	<input type="text" value="Filter"/>	<input type="text" value="Filter"/>	<input type="text" value="Filter"/>	<input type="text" value="Filter"/>
EnGenius	ENS1200		login.sh	100.0
EnGenius	EAP1200H		login.sh	100.0
EnGenius	EnStationAC		login.sh	100.0
EnGenius	EWS500AP		login.sh	100.0
EnGenius	EWS300AP		login.sh	100.0
EnGenius	EAP1750H		login.sh	100.0

The Power of Correlation: Telnet Jailbreak

- **Telnet jailbreak**, publicly identified in 6 devices (5 EnGenius, 1 Araknis)
- Found in *42 other devices* from **EnGenius, WatchGuard**

EnGenius	EAP1750H		login.sh	100.0
WatchGuard	XTM 33		login.sh	78.9
WatchGuard	XTM 330		login.sh	78.9
WatchGuard	XTM 330		login.sh	78.9
WatchGuard	Firebox T70		login.sh	78.9
WatchGuard	Firebox M200 and M300		login.sh	78.9
WatchGuard	XTM 33		login.sh	78.9

The Power of Correlation: Telnet Jailbreak

- **Telnet jailbreak**, publicly identified in 6 devices (5 EnGenius, 1 Araknis)
- Found in *42 other devices* from **EnGenius**, **WatchGuard** and **TRENDNet**

EnGenius	ENS202EXT		login.sh	78.9
EnGenius	EnStation2		login.sh	78.9
trendnet	TEW-753DAP		login.sh	78.9
EnGenius	ENS500EXT		cli.sh	63.2

The Power of Correlation: Webshell

```
> ?  
cfg sys sh  
> 
```

dev.htm interface



`nvram` from root shell

```
wl1_dispatch_mode=1  
wl0_rxchain_pwrsave_enable=1  
wl0_hw_rxchain=3  
http_passwd=[REDACTED]  
wl_wpa_psk=[REDACTED]  
wl_guest_passwd=[REDACTED]  
wl_rpcq_rxthresh=512  
dhcp_reserveMAC4=48:D2:24 [REDACTED]
```



The Power of Correlation: Webshell

- **Belkin Webshell**, publicly identified in 1 device from Belkin
- Found in *28 other devices* from **Belkin**

Brand	Model	Firmware version	File	Match (%)
<input type="text" value="Filter"/>	<input type="text" value="Filter"/>	<input type="text" value="Filter"/>	<input type="text" value="Filter"/>	<input type="text" value="Filter"/>
belkin	N750 DB Wi-Fi Dual-Band N+ Gigabit Router		dev.htm	100.0
belkin	AC 1800 DB Wi-Fi Dual-Band AC+ Gigabit Router		dev.htm	100.0
belkin	AC 1800 DB Wi-Fi Dual-Band AC+ Gigabit Router		dev.htm	100.0
belkin	N600 DB Wireless Dual-Band N+ Router		dev.htm	100.0



The Power of Correlation: Webshell

- **Belkin Webshell**, publicly identified in 1 device from Belkin
- Found in *28 other devices* from **Belkin, Ubiquiti**

belkin	N750 DB Wi-Fi Dual-Band N+ Router		dev.htm	100.0
Ubiquiti Networks	aircam		dev.htm	100.0
Ubiquiti Networks	aircam-dome		dev.htm	100.0
Ubiquiti Networks	aircam-mini		dev.htm	100.0
belkin	F9K1118		dev.htm	100.0



The Power of Correlation: Webshell

- **Belkin Webshell**, publicly identified in 1 device from Belkin
- Found in *28 other devices* from **Belkin, Ubiquiti, TP-Link**

belkin	N450 DB Wi-Fi Dual-Band N+ Router		dev.htm	100.0
tp-link	TL-WR740N		dev.htm	100.0
belkin	F9K1113		dev.htm	100.0
belkin	AC 1200 DB Wi-Fi Dual-Band AC+ Gigabit Router		dev.htm	100.0



The Power of Correlation: Webshell

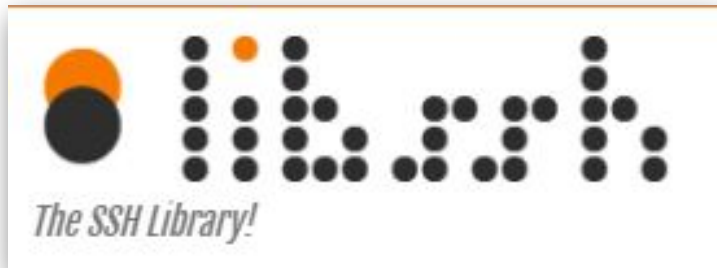
- **Belkin Webshell**, publicly identified in 1 device from Belkin
- Found in *28 other devices* from **Belkin, Ubiquiti, TP-Link** and **TRENDnet**

Ubiquiti Networks	nsm5		dev.htm	100.0
belkin	F9K1102		dev.htm	80.0
belkin	N600 DB Wireless Dual-Band N+ Router		dev.htm	80.0
trendnet	TU2-NU4		dev.htm	80.0



The Power of Correlation: LibSSH Auth Bypass

- CVE-2018-10933: **Authentication Bypass in libSSH**
- Client sends **SSH2_MSG_USERAUTH_SUCCESS** instead of **SSH2_MSG_USERAUTH_REQUEST**, an attacker could successfully authenticate without credentials



Technical Advisory: Authentication Bypass in libSSH

Vendor: libSSH

Vendor URL: <https://www.libssh.org/>

Versions affected: Versions of libSSH 0.6 and above, prior to 0.7.6 or 0.8.4.

Author: Peter Winter-Smith [peter.winter-smith\[at\]nccgroup.com](mailto:peter.winter-smith[at]nccgroup.com)

Advisory URL / CVE Identifier: CVE-2018-10933 - <https://www.libssh.org/security/advisories/CVE-2018-10933.txt>

Risk: Critical - Authentication Bypass

The Power of Correlation: LibSSH Auth Bypass

Vulnerable versions found in 5 devices from **Belkin**, **TP-Link**, **WatchGuard** *

The screenshot shows a search interface for the library 'libssh.so.4'. The search bar at the top contains 'libssh.so.4' and has a close button. To the right of the search bar are two dropdown menus labeled 'Category' and 'Brand'. Below the search bar, there are five product cards arranged in a grid. Each card displays a small icon of the device, the product name, and the number of firmwares found. At the bottom of the page, there is a pagination indicator showing '1' and a summary of results: '5 products • 6 firmwares'.

Product	Firmwares
Belkin F7D7501	2 firmwares
Tp-Link Deco	1 firmware
Tp-Link Deco+M5	1 firmware
Tp-Link TL-WA701N V2.0	1 firmware
Watchguard Technologies Firebox T50	3 firmwares



* Existence of the vulnerable library does not imply these systems are vulnerable in their configurations

The Power of Correlation: LibSSH Auth Bypass

SSH_PACKET_CALLBACK:

```
0 @ 00006c24 int32_t r3 = [arg1 + 0x4e4].d
1 @ 00006c2c bool cond:0 = r3 s> 3
2 @ 00006c30 int32_t r4 = arg1 // ssh_packet_userauth_success
3 @ 00006c34 if (cond:0) then 4 @ 0x6ce8 else 10 @ 0x6c3c
```

```
10 @ 00006c3c int32_t r0 = r4
11 @ 00006c48 ssh_log(r0, 3, data_30870, r3, var_10) {"Received SSH_USERAUTH_SUCCESS"}
12 @ 00006c58 int32_t r0_1 = r4
13 @ 00006c5c ssh_log(r0_1, 2, data_30890) {"Authentication successful"}
14 @ 00006c60 int32_t r3_1 = [r4 + 0x4a8].d
15 @ 00006c6c bool cond:1 = r3_1 == 0
16 @ 00006c70 [r4 + 0x46c].d = 2 // SSH_AUTH_STATE_SUCCESS
17 @ 00006c74 [r4 + 0x45c].d = 8 // SSH_SESSION_STATE_AUTHENTICATED
18 @ 00006c78 if (cond:1) then 19 @ 0x6c94 else 21 @ 0x6c7c
```



4. Source Code Analyzers

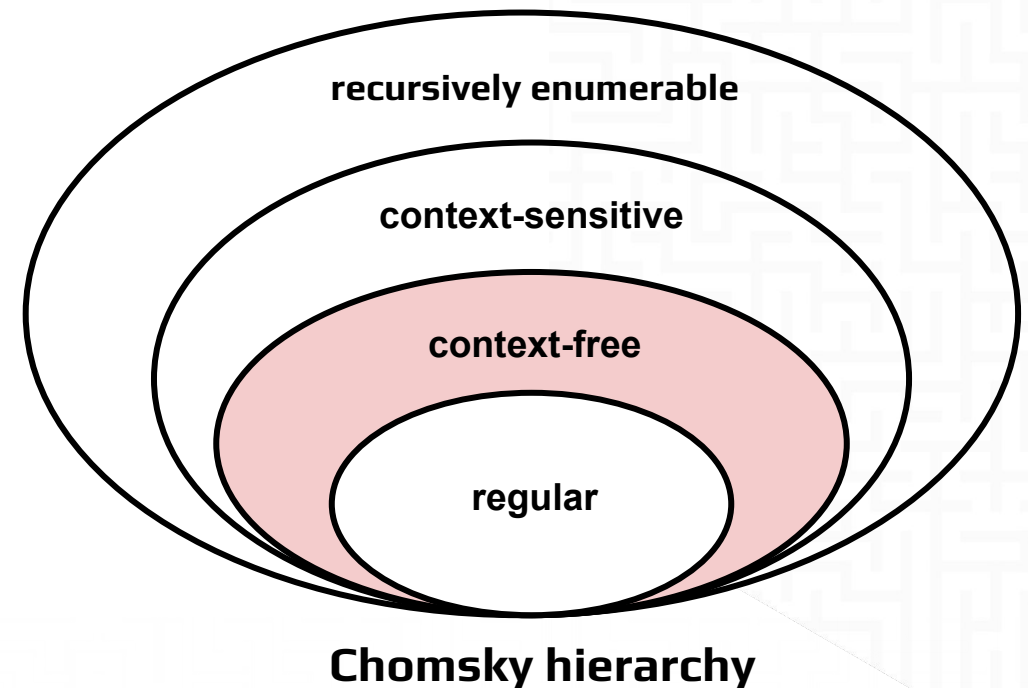


Source Code Analyzers: About Complexity

Source files account for a large portion of **attack surface** on many devices

- Pattern matching and regular expressions *cannot* find most issues
- There are more advanced methods for analyzing source code

Most languages are at least **context-sensitive**, but we aim to simplify them to grammars that are **context-free**




Source Code Analyzers: Why regex won't work

```


// What value does this function return?
public String moreOrLess(int x) {
    String y = null;
    if (x > 0) {
        y = "more";
    } else if (x < 0) {
        y = "less";
    }
    return y.toUpperCase();
}

```



Source Code Analyzers: Why regex won't work

```
// What value does this function return?  
public String moreOrLess(int x) {  
    String y = null;  
    if (x > 0) {  
        y = "more";  
    } else if (x < 0) {  
        y = "less";  
    }  
    return y.toUpperCase();  
}
```



$$y_4 = \Phi(y_1, y_2, y_3) = \Phi(\text{null}, \text{"more"}, \text{"less"})$$

Source Code Analyzers: Lexing

Lexing is the process of breaking an inputstream into discrete components (lexemes) and applying defining characteristics to them

```
const fruit = "apple";
```

```
type: VariableDeclaration  
start: 0  
end: 22  
kind: "const"
```

```
type: Identifier  
start: 6  
end: 11  
name: "fruit"
```

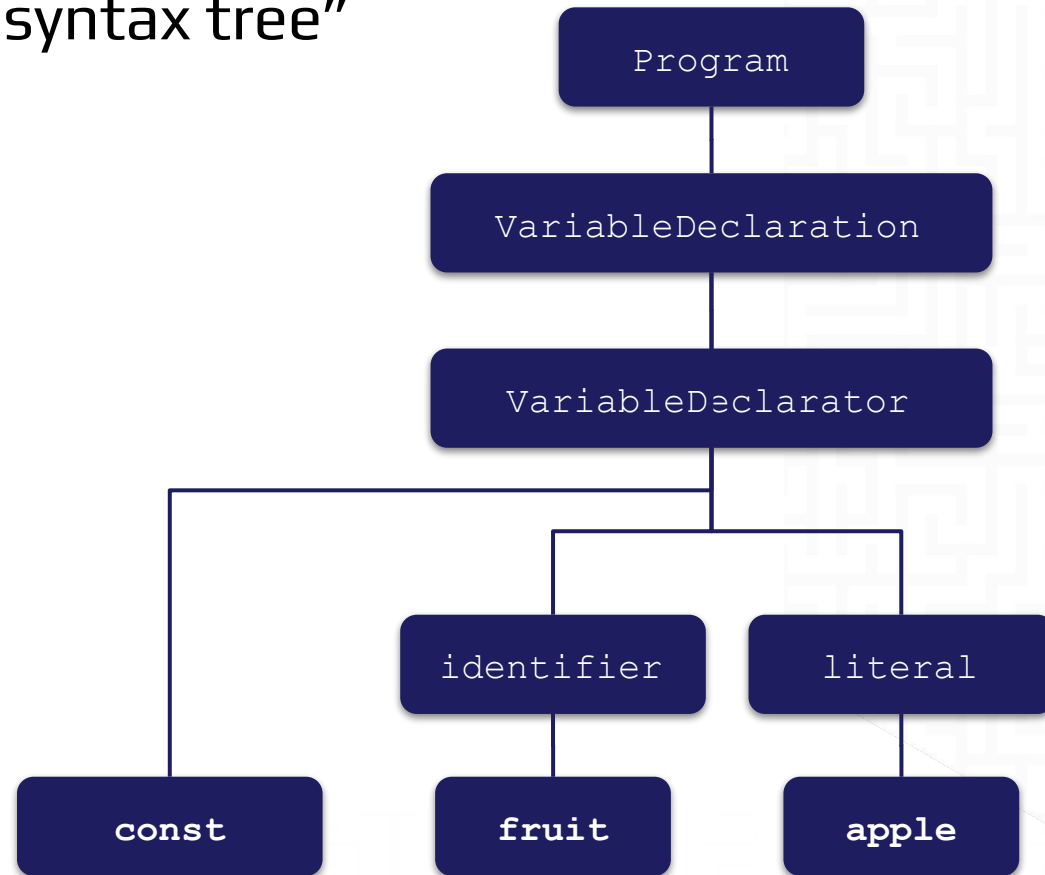
```
type: Literal  
start: 14  
end: 21  
value: "apple"  
raw: "\"apple\""
```



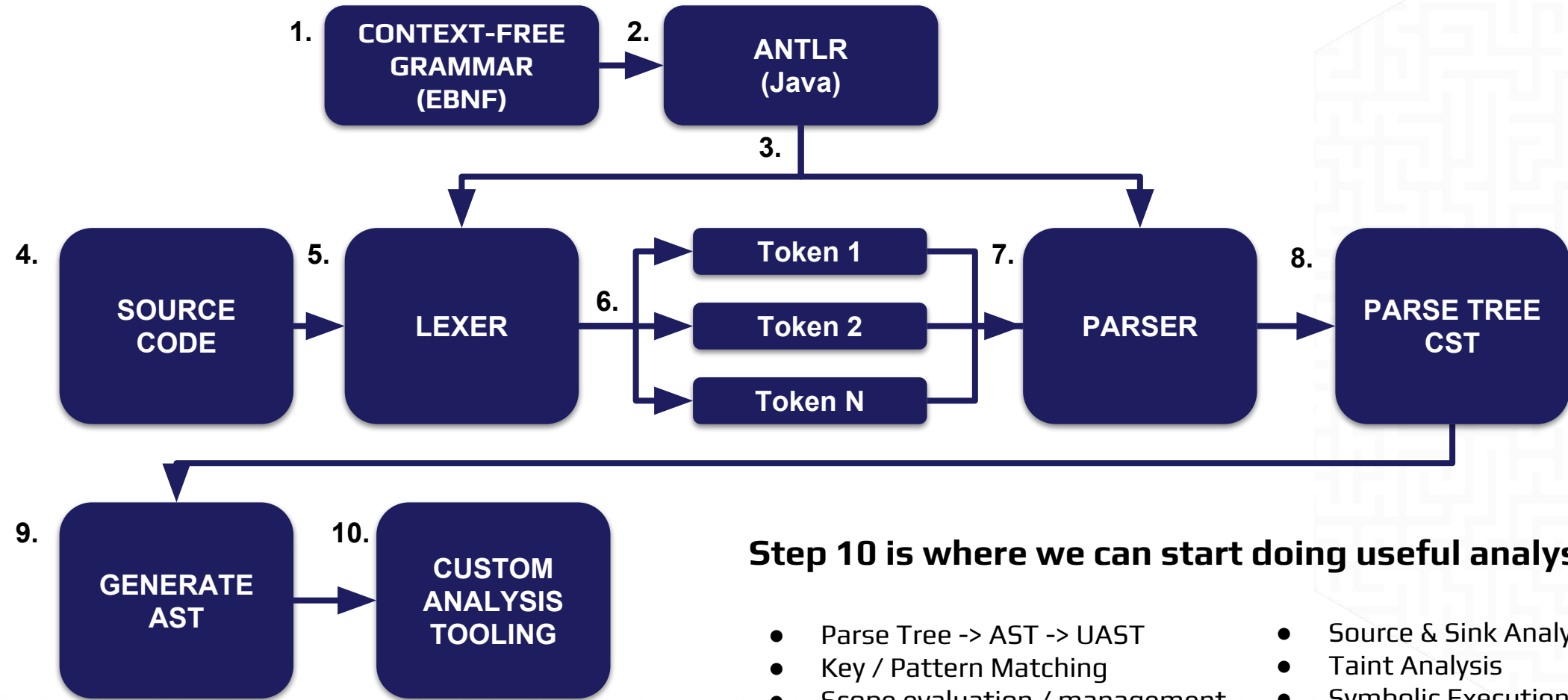
Source Code Analyzers: Parsing

Parsing is the process of applying structure to an input token stream in the form of a parse tree or “concrete syntax tree”

```
const fruit = "apple";
```



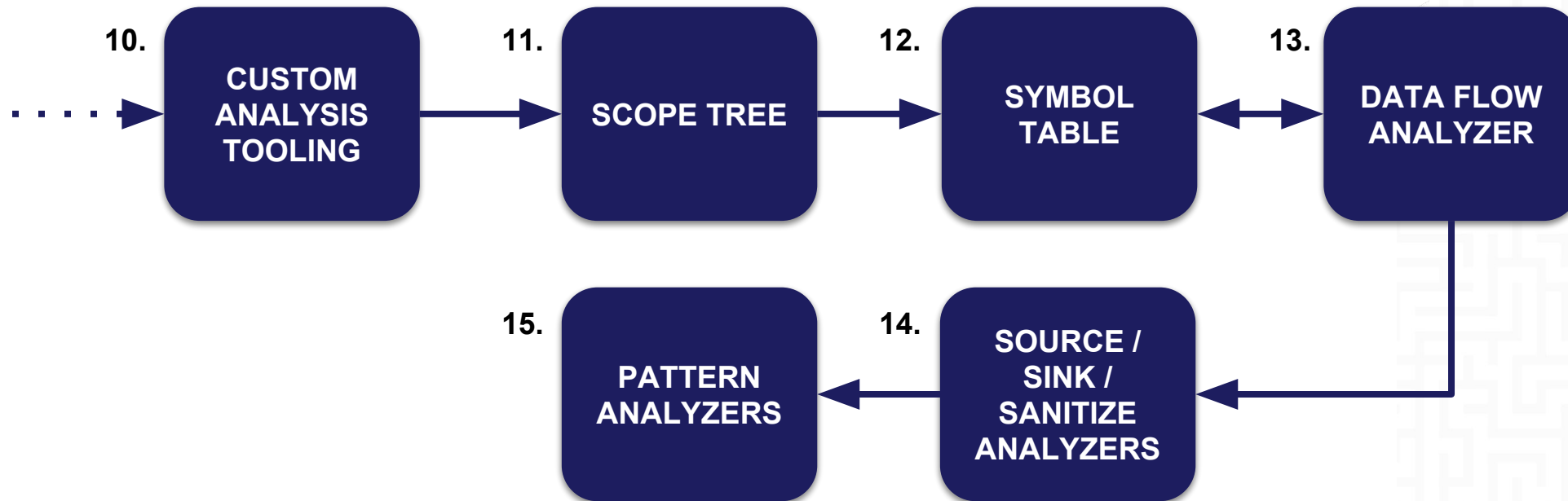
Source Code Analyzers: Before we can begin



Step 10 is where we can start doing useful analysis

- Parse Tree -> AST -> UAST
- Key / Pattern Matching
- Scope evaluation / management
- Global symbol table evaluation
- Source & Sink Analysis
- Taint Analysis
- Symbolic Execution
- Constraint Solving

Source Code Analyzers: Getting to work



Step 10 is where we can start doing useful analysis

- Parse Tree -> AST -> UAST
- Key / Pattern Matching
- Scope evaluation / management
- Global symbol table evaluation
- Source & Sink Analysis
- Taint Analysis
- Symbolic Execution
- Constraint Solving



Source Code Analyzers: Demo 1

Shell script analysis

EnGenius restricted shell (`login.sh`, 363 lines of code)

```
192     while [ true ]; do
193         input="$(read_line "> ")"
194         command="$(echo "$input" | sed -e "s/^[ \t]*\([^ \t]*\)[ \t]*.*$/\1/g")"
```

```
298         elif [ "$input" = "1d68d24ea0d9bb6e████████████████████" ]; then
299             exec /bin/ash --login
```

```
totes:fs-vr-bash john$ time node app.js -s ~/Desktop/hitb/demos/login.sh
bash found a possible 'DangerousExec' in file '/Users/john/Desktop/hitb/demos/login.sh' on line 189: exec /bin/ash --login
bash found a possible 'DangerousExec' in file '/Users/john/Desktop/hitb/demos/login.sh' on line 299: exec /bin/ash --login
bash found a possible 'DangerousExec' in file '/Users/john/Desktop/hitb/demos/login.sh' on line 326: exec /bin/ash --login
Finished analyzing 1 files. Found 3 issues.
```

```
real    0m0.600s
user    0m0.594s
sys     0m0.067s
```

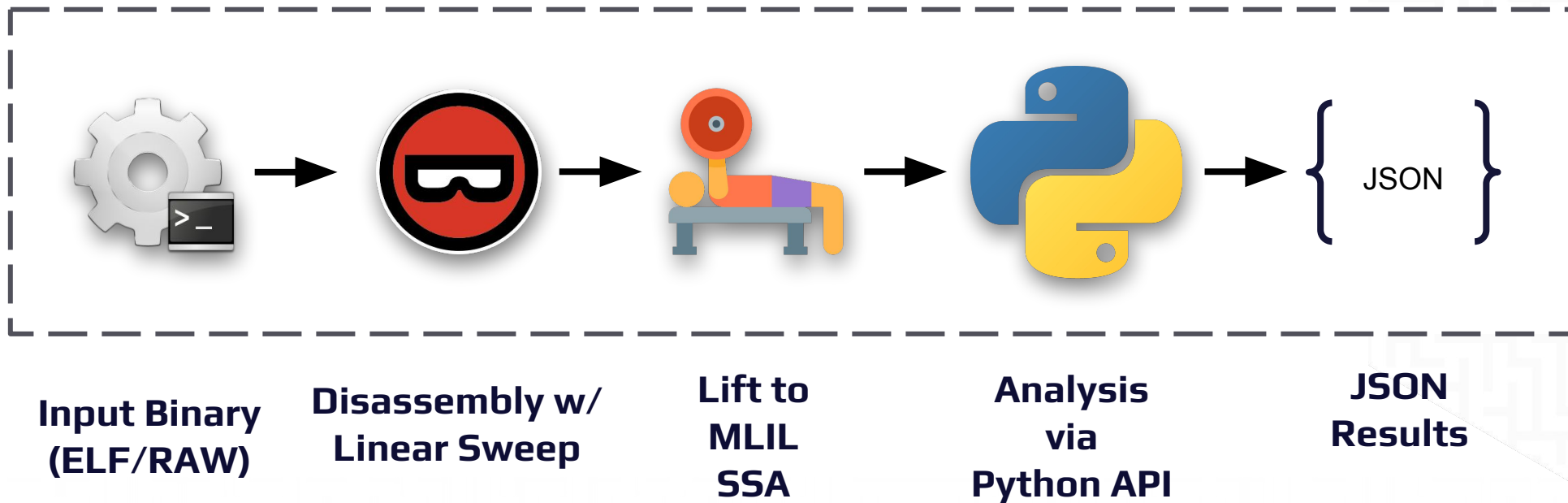


5. Binary Analyzers



Binary Analyzers

- **Binary files** account for most of the heavy lifting in IoT devices
- Architecture considerations in analysis: ARM, MIPS, PowerPC, x86, etc.
 - Multi-architecture handled through an **intermediate language**



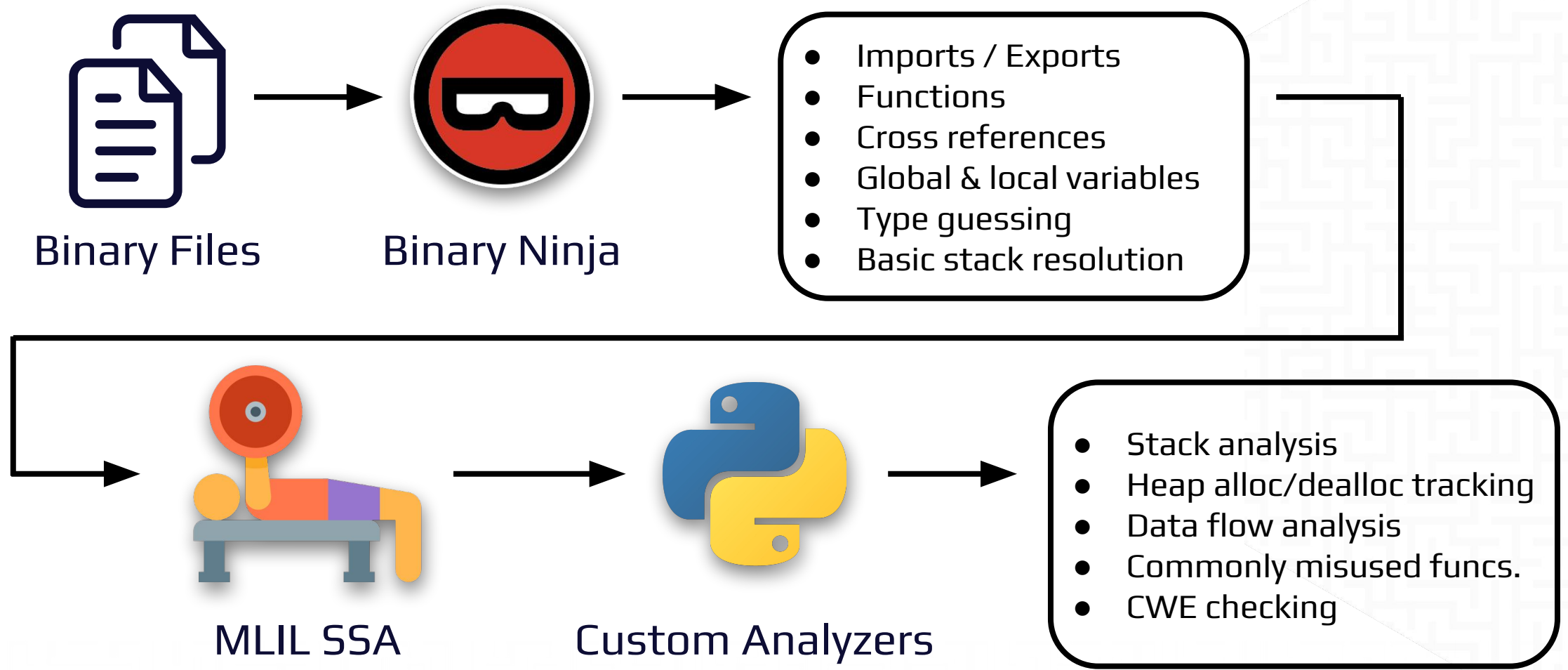
Binary Analyzers

- Zero extra effort due to Binary Ninja's IL, one additional step to disassembly
- Supports all common IoT architectures, with support to add new processors

<pre>str fp, [sp, #-4]! add fp, sp, #0 sub sp, sp, #12 str r0, [fp, #-8] ldr r3, [fp, #-8] ldr r2, [fp, #-8] mul r3, r2, r3 mov r0, r3 sub sp, fp, #0 ldr fp, [sp], #4 bx lr</pre>	<pre>addiu \$sp, \$sp, -8 sw \$fp, 4(\$sp) move \$fp, \$sp sw \$4, 8(\$fp) lw \$3, 8(\$fp) lw \$2, 8(\$fp) nop mult \$3, \$2 mflo \$2 move \$sp, \$fp lw \$fp, 4(\$sp) addiu \$sp, \$sp, 8 j \$31 nop</pre>	<pre>stwu 1, -32(1) stw 31, 28(1) mr 31, 1 stw 3, 8(31) lwz 10, 8(31) lwz 9, 8(31) mullw 9, 10, 9 mr 3, 9 addi 11, 31, 32 lwz 31, -4(11) mr 1, 11 blr</pre>	<pre>int32_t var_c = arg1 uint32_t reg2 = zx.d(arg1) uint32_t reg2_1 = zx.d(reg2 * var_c) uint32_t reg1 = zx.d(reg2_1) return reg1</pre>
ARM	MIPS	PPC	Generalized MLIL



Binary Analyzers: Process



Binary Analyzers: Demo 1

ARM ELF, abnormal string comparisons & frequency analysis

- BusyBox analysis

```
strncmp at 0x41108 (in function 0x410c0) checks for '/dev/hd'  
strncmp at 0x11a28 (in function 0x118a0) checks for 'username='  
strncmp at 0x11dcc (in function 0x118a0) checks for '7ujMko0'  
strncmp at 0x11ab4 (in function 0x118a0) checks for 'passwd='  
strcmp at 0xd014 (in function 0xce60) checks for '--install'  
strcmp at 0xd12c (in function 0xce60) checks for '--help'
```

```
Finished disassembly phase in 20.8 seconds  
Finished analysis in phase 21.8 seconds  
Total analysis time: 42.5 seconds
```



Binary Analyzers

ARM ELF, abnormal string comparisons & frequency analysis

- Freq. analysis of 9,574 unique versions of busybox
- Sampling of strings referenced in **strcmp** and **strncmp**:

7614: 'default'	10: 'http://'	2: 'mfgrout'
4151: '--help'	10: 'b'	2: '.deb'
3897: 'inet'	9: 'PROCESS_ACCOUNTING'	2: 'lst'
3357: 'rootfs'	9: 'opts='	2: '%TGBnhy6m'
2939: '255.255.255.255'	9: '7ujMko0'	2: 'noarp'
2683: 'gz'	9: 'username='	2: 'forever'
2667: 'auto'	9: 'confold'	2: 'qaZ*IK<9o1.'
2049: '-net'	9: 'TERM=linux'	2: 'boundary='
2049: '-host'	9: 'pw'	2: 'show'
1980: 'login'	9: 'endcmd'	2: 'y'



Binary Analyzers

ARM ELF, abnormal string comparisons & frequency analysis

- Freq. analysis of 9,574 unique versions of busybox
- Sampling of strings referenced in **strcmp** and **strncmp**:

```
7614: 'default'
4151: '--help'
3897: 'inet'
3357: 'rootfs'
2939: '255.255.255.255'
2683: 'gz'
2667: 'auto'
2049: '-net'
2049: '-host'
1980: 'login'
10: 'http://'
10: 'b'
9: 'PROCESS_ACCOUNTING'
9: 'opts='
9: '7ujMko0'
9: 'username='
9: 'confold'
9: 'TERM=linux'
9: 'pw'
9: 'endcmd'
2: 'mfgroot'
2: '.deb'
2: 'lst'
2: '%TGBnhy6m'
2: 'noarp'
2: 'forever'
2: 'qaZ*IK<9o1.'
2: 'boundary='
2: 'show'
2: 'y'
```



Binary Analyzers: Demo 2

ARM ELF, buffer overflow checking

- Asus “Download Master” feature in `asus_lighttpd`
- Example of bug with no provable state (i.e. *not* a vulnerability)

```
<meta HTTP-EQUIV="REFRESH" content="0;url='http://www.example.com/'" />
```

```
0 @ 0001f800  int32_t r4 = arg1
1 @ 0001f810  void* stackBuffer = &stackBuffer
2 @ 0001f814  memset(stackBuffer, 0, 512)
3 @ 0001f818  int32_t funcArg = r4
4 @ 0001f820  r0 = strstr(funcArg, 0x3bde0) {"HTTP-EQUIV="REFRESH"}
```

```
42 @ 0001f8c0  int32_t srcBuffer = r6_1
43 @ 0001f8c4  uint32_t n = adjustedLength
44 @ 0001f8c8  void* destStackBuffer = &stackBuffer
45 @ 0001f8cc  strncpy(destStackBuffer, srcBuffer, n)
46 @ 0001f8d0  int32_t r0_10 = [stderr].d
```



FINITE  STATE

Research Summary



Research Summary

- Discovery of **4** verified IoT backdoors (**75** unique devices)
 - Modified busybox, custom httpd, CGI handlers
- Discovery of **11 unverified** IoT backdoors (**107** unique devices)
 - Number one source is custom httpd implementations
- Automated verification is no where near a solved problem
 - Unknown configurations, emulation challenges, dead code





Questions?

Feel free to contact me with any questions you think of later

john@finitestate.io

[@cetfor](#)