HACKER NIGHTMARES

GIVING HACKERS A HEADACHE WITH EXPLOIT MITIGATIONS

By Maria Markstedter

Arm Research Summit 2020

Arm Innovator

Member of the Arm Innovator program since its inception.

Collaboration with Arm

Ongoing collaboration with Arm on research and developer education.

MARIA MARKSTEDTER

Security Researcher

Mobile and IoT security, with focus on exploit development.

CEO of Azeria Labs

Providing IoT and Mobile security services and trainings.

Formal Education

Bachelor's degree in IT Security.
Master's in Enterprise Security.

Forbes 30 under 30

List member of Forbes 30 under 30 in technology, Europe, 2018.

AZERIA

Azeria
@Fox0x01

Azeria Labs
@azeria_labs

Security researchers and founder of Azeria Labs, working with large tech companies and law-enforcement agencies on projects surrounding IoT and mobile security. Interests include reverse engineering Arm-based devices, security research, and exploit development for mobile and IoT.

Currently writing two books on Arm Assembly and Reverse Engineering, and Vulnerability Discovery and Exploit Mitigations.

Website: http://azeria-labs.com/

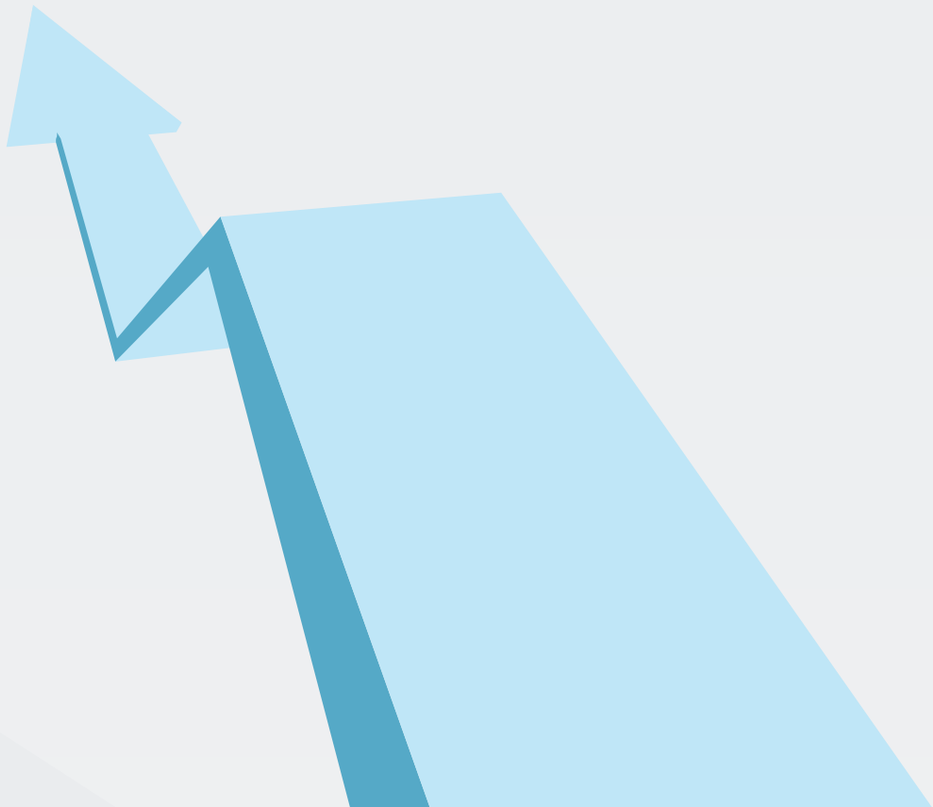
Book updates Mailing list: https://arm-exploitation.com/

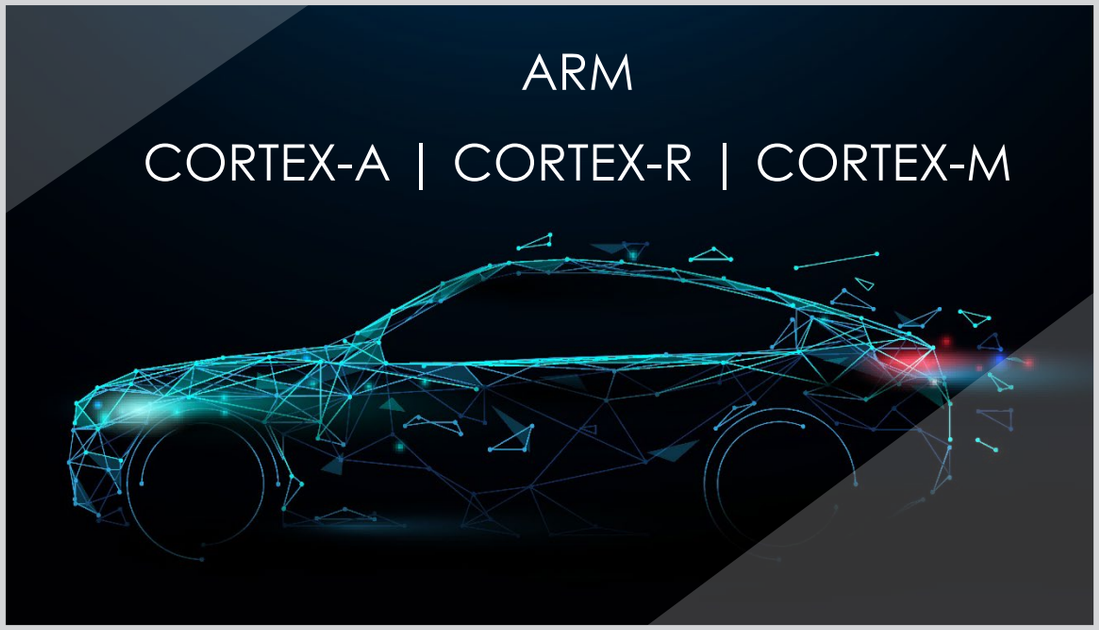
Email: contact@azeria-labs.com

IOT DEVICES
GROWTH

How many IoT devices will be connected to the internet by 2030?

Some say 120 billion, other say 1 trillion. In any case, the number is BIG.



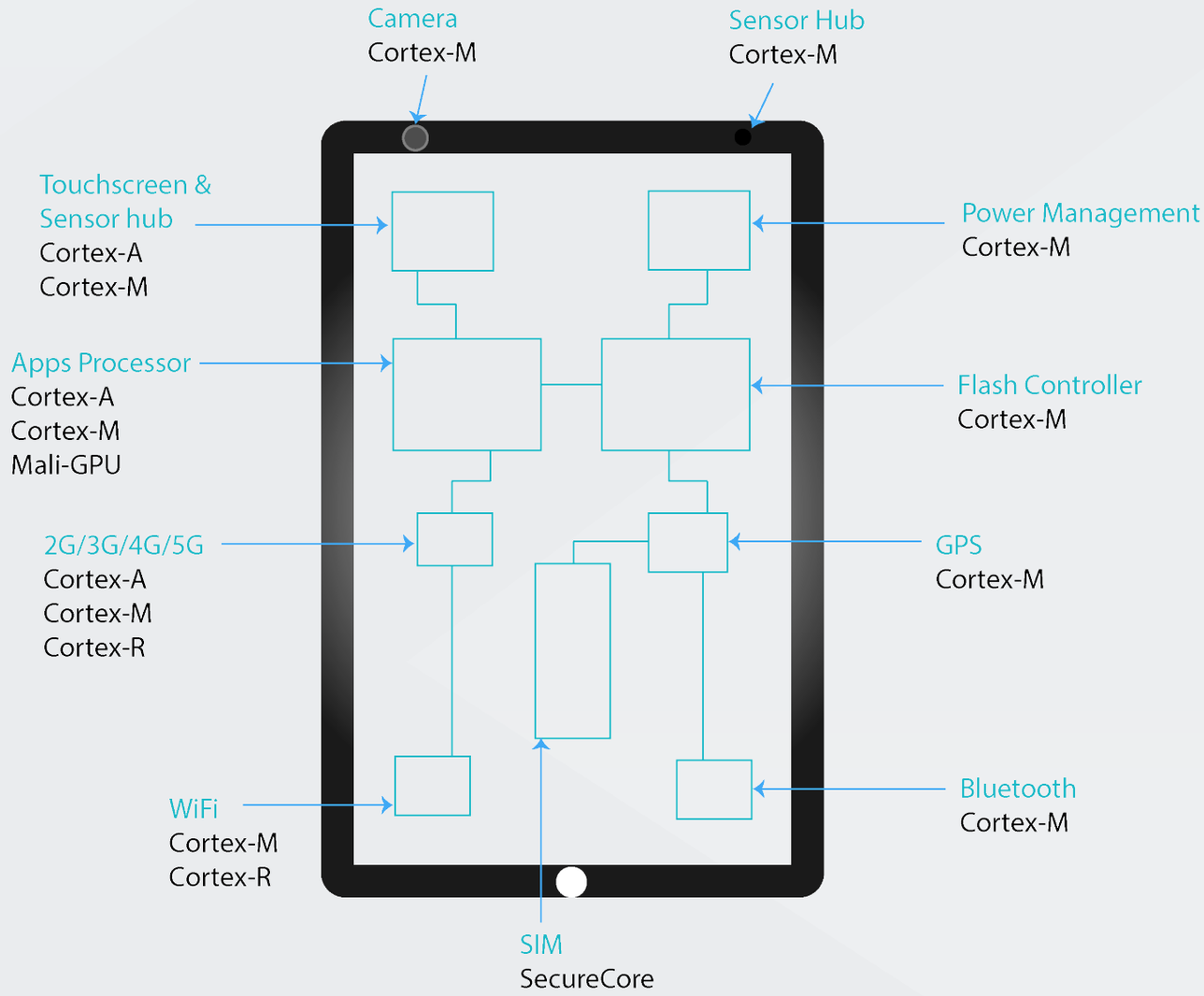


ARM PROCESSORS
AUTOMOTIVE

Modern cars are equipped with Arm processor classes used for components such as Powertrain, Vision ADAS, Navigation and Infotainment, and Autonomous driving.

Processor Class	Designed For	Automotive Applications
Arm Cortex-A	• High performance • Optimized for rich Operating Systems and hypervisors	• ADAS Vision • Autonomous • IVI • Digital Cockpit • Connectivity
Arm Cortex-R	• Optimized for high performance • Hard real-time deterministic applications and RTOS	• Autonomous • ADAS radar • Connectivity • Powertrain • Chassis
Arm Cortex-M	• Smallest area and lowest power • Optimized for discrete processing and microcontrollers	• Body electronics • Gateway • SoC management • Sensors

ARM PROCESSORS
MOBILE



IOT ATTACKS IN THE WILD

Microsoft: Russian state hackers are using IoT devices to breach enterprise networks

Microsoft said it detected Strontium (APT28) targeting VoIP phones, printers, and video decoders.

MORE FROM CATALIN CIMPANU

By Catalin Cimpanu for Zero Day | August 5, 2019 -- 18:30 GMT (19:30 BST) | Topic: Security

Security
Black Hat and DEF CO

Article by zdnet

CYBER-ESPIONAGE GROUPS INCREASINGLY USING IOT DEVICES

Strontium going after IoT devices isn't a novel tactic. The same group previously created a botnet of tens of thousands of home routers using the VPNFilter malware.

Experts believed Strontium was preparing to use the botnet to launch DDoS attacks on the night of the UEFA Champions League final that was going to be held in Kyiv, Ukraine that year.

But besides Strontium, other state-sponsored groups have also started targeting IoT devices, and primarily routers. Examples include the LuckyMouse, Inception Framework, and Slingshot groups.

Microsoft said hackers used the compromised IoT devices as an entry point into their targets' internal networks, where they'd scan for other vulnerable systems to expand this initial foothold.

"After gaining access to each of the IoT devices, the actor ran tcpdump to sniff network traffic on local subnets," Microsoft said.

"They were also seen enumerating administrative groups to attempt further exploitation. As the actor moved from one device to another, they would drop a simple shell script to establish persistence on the network which allowed extended access to continue hunting," the OS maker added.

Using IoT to hide behind proxies

Inception is continuing to use chains of infected routers to act as proxies and mask communications between the attackers and the cloud service providers they use. Certain router manufacturers have UPnP listening on WAN as a default configuration. Akamai research has found that there are 765,000 devices vulnerable to this attack. These routers are hijacked by Inception and configured to forward traffic from one port to another host on the internet. Abuse of this service requires no custom malware to be injected on the routers and can be used at scale very easily. Inception strings chains of these routers together to create multiple proxies to hide behind.

Symantec: Inception Framework

This paper in a nutshell:

- Slingshot is a new, previously unknown cyber-espionage platform which rivals Project Sauron and Regin in complexity
- Slingshot has been active since at least 2012 until February 2018
- We observed almost one hundred Slingshot victims, mainly in the Middle East and Africa
- The attackers exploited an unknown vulnerability in Mikrotik routers as an infection vector

Kaspersky: The Slingshot APT

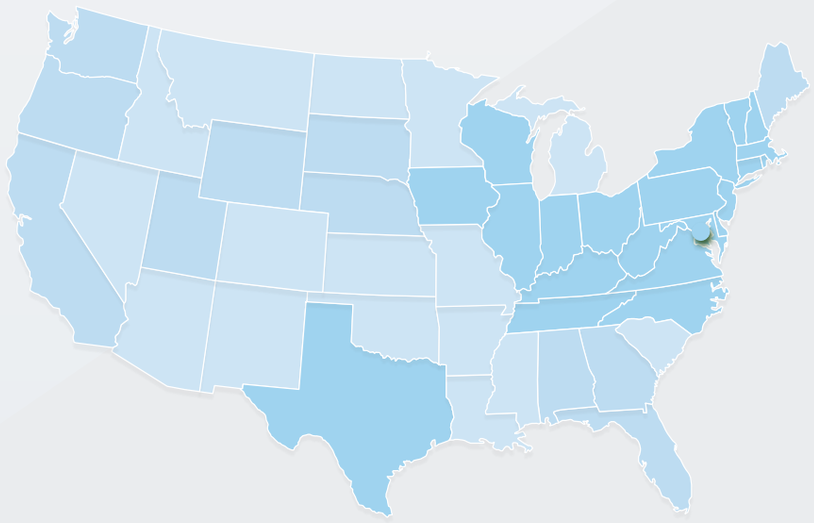
LILY HAY NEWMAN 10.21.16 01:04 PM

What We Know About Friday's Massive East Coast Internet Outage

GETTY IMAGES

MIRAI

#NEVERFORGET



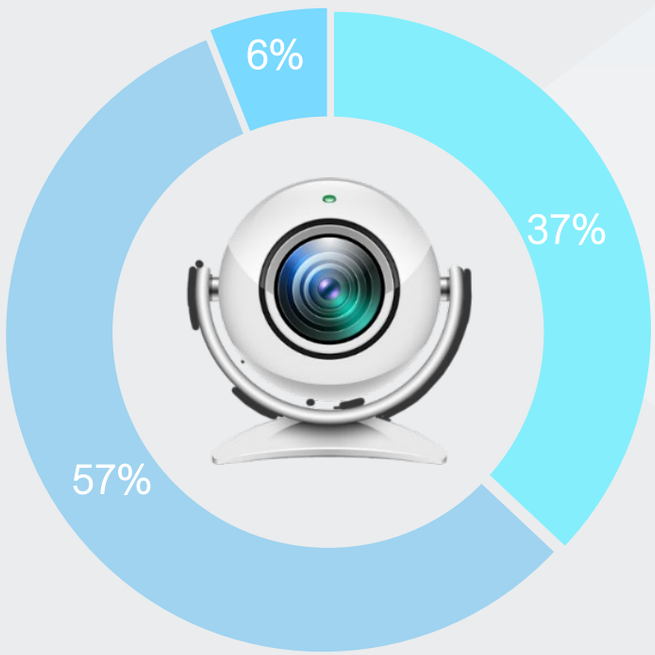
DDoS attack on Dyn

October 2016

Attack on DNS provider resulted in downtime of high-profile websites, including GitHub, Netflix, Reddit, Twitter, PayPal, AWS.

1TB per second DDoS

600K



ROUTERS

CAMERAS / DVR

UNKNOWN / OTHER

MIRAI VARIANTS

12 DECEMBER 2017

Satori variant exploiting 0day in Huawei HG532 router. Botnet as a Service $15k for 600k devices with up to 700GBps

14 JANUARY 2018

New variant extended processor support: ARM, MIPS, x86, PowerPC, ARC, Argonaut RISC

18 JANUARY 2018

Successor of Mirai designed to hijack Cryptocurrency mining operations.

26 JANUARY 2018

New variants which weaponizes D-Link router exploit to enlist further vulnerable IoT devices.

MARCH 2018

New variant "OMG“ with added configurations to turn vulnerable IoT devices into proxy servers.

MAY / JUNE 2018

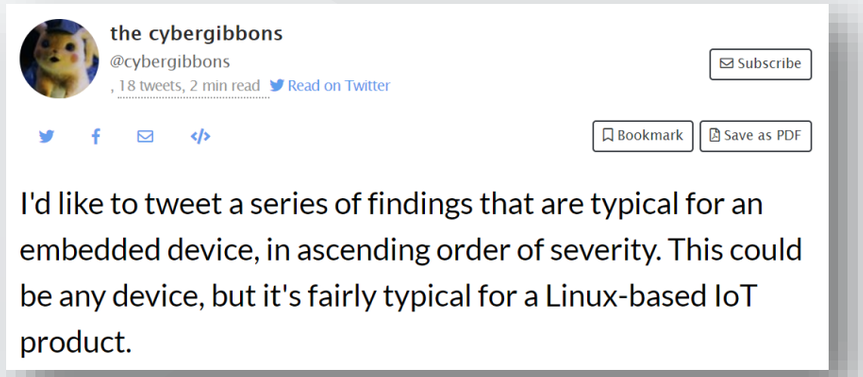
New variant “Wicked” added at least three new exploits, including Netgear routers and CCTV-DVRs.

EARLY JULY 2018

At least thirteen versions actively infecting Linux IoT, including Android devices with ADB on TCP/5555 enabled.

END OF 2018

Variant “Miori”, spreads via RCE in ThinkPHP. New variants in 2019 still target this vulnerability.



Shout-out to Andrew Tierney (@cybergibbons)

Link to Tweet thread:
https://threadreaderapp.com/thread/931826171003723776.html

ARE YOU A *LOW HANGING FRUIT?*

Even the most trivial and well-known security measures are being neglected by IoT vendors. Andrew Tieney, researcher at Pen Test Partners, listed the most common issues he encounters in embedded devices.

In this talk, we will look at the relationship between exploit mitigations and the use of unsafe functions.

1. Out-of-date CA bundle - lots of devices have a CA bundle from 2012 or before. This means that many certs are out-of-date and there are some no longer trusted CAs. Developers will switch off cert validation as a result.

2. Device lacks secure storage - the SoC used by the device has no provision to securely store keys or confidential material. An attacker with physical access can recover keys, certificates, hashes, and passwords.

3. Factory reset not correctly implemented - either configuration (such as user's SSID/PSK), authentication information (certificates etc.) or data (stored videos) are not deleted or renewed when reset.

4. Encryption implementation issues - a custom protocol used by the device does not implement crypto correctly. Examples - encrypt without MAC, hardcoded IV, weak key generation.

5. System not mimimised - the system is running services and processes that aren't used. It's common to find a web UI running on a system, but undocumented and useless to consumers.

6. Serial consoles enabled - either raw serial or USB serial is enabled, allowing either the bootloader, a login prompt, or a unprotected shell to be accessed.

7. WiFi connection process exposes SSID/PSK - it's very common for devices to use a WiFi AP or BLE to allow the app to communicate the user's SSID/PSK for connection. Often in the plain. Attacker needs proximity physically.

8. Firmware not signed - this allows someone to create malicious firmware and deploy it to devices. Firmware not encrypted - this makes it much easier to examine firmware and find issues.

9. Busybox not minimised - busybox has been built with every single tool possible, providing a rich set of tools for an attacker to use.

10. Root user allowed to login - the root user either has no password, or a hardcoded password. Another vulnerability will allow them to login and use the system.

11. Compile time hardening not used. PIE/NX/ASLR/RELRO/Fortify haven't been used. They make exploiting buffer overflows harder.

12. Unsafe functions used - strcpy/sprintf/gets are used heavily in binaries found on the system. These are closely associated with buffer overflow-tastic systems.

13. All processes run as root - no principle of least privilege followed. Lots of devices could do this, but don't. No need to privesc when compromised.

14. Device does not validate SSL certificates - the HTTPS communications used by the device can be man-in-the-middled by an attacker. Can lead to serious compromise, especially if firmware updates delivered by this mechanism.

CONCLUSION: The device and system can't be immediately compromised. But in the event of another vulnerability being found, there is little stopping an attacker from totally owning the device.

Once again, the problem is adversarial research has conditioned vendors to ignore these level of findings. They aren't remote root access, not every device has been compromised. So often, these issues will not be fixed.

This is a real problem. The solution isn't pen-testing and remediation. The solution is secure development practices. All of these are known.

COMMON PROBLEMS

OLD CODE BASES

Millions of lines of code written without security standards in mind.

DEVELOPER EDUCATION

Lack of accessible secure coding training for developers.

MITIGATION MYTHS

Performance concerns about implementation of exploit mitigations.

OUTDATED CURRICULUMS

Use of unsafe functions taught in Computer Science lectures.

SECURITY NEGLECTED

Security often treated as an afterthought after project completion.

LACK OF REGULATION

Lack of incentive for many vendors to take security seriously.

VULNERABLE
SMART HOMES

Smart Home Hubs

Smart Home Hubs is the central controller that allows to remotely connect and manages IoT devices such as: voice assistants, smart plugs, light bulbs, thermostats, cameras, Doorbells & locks, speakers, & more.

VULNERABLE
SMART HOMES

The Samsung Smart Things Hub v2

Central controller that allows to remotely connect and manages IoT devices such as: voice assistants, smart plugs, light bulbs, thermostats, cameras, Doorbells & locks, speakers, & more.

Cisco Talos Intelligence discovered several vulnerabilities that made it possible for an attacker to control various devices connected to the Smart Hub.

Potential Risks

Smart Locks
Gain physical access to the home by unlocking connected smart locks.

Cameras
Remotely monitor occupants via cameras deployed within the home.

Alarm System
Disable the motion detectors used by the home alarm system.

Smart Plugs
Cause physical damage to devices connected to smart plugs.

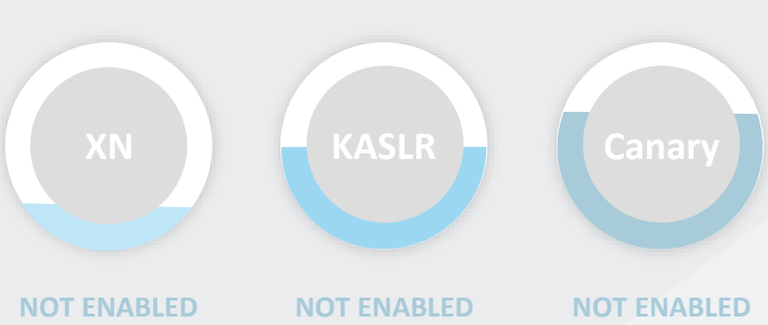
CVE-2018-3863
CVE-2018-3864
CVE-2018-3865
CVE-2018-3866
CVE-2018-3867
CVE-2018-3872
CVE-2018-3878
CVE-2018-3880
CVE-2018-3897
CVE-2018-3902
CVE-2018-3904
CVE-2018-3905
CVE-2018-3906
CVE-2018-3912
CVE-2018-3913
CVE-2018-3914
CVE-2018-3915
CVE-2018-3916
CVE-2018-3917
CVE-2018-3919
CVE-2018-3925

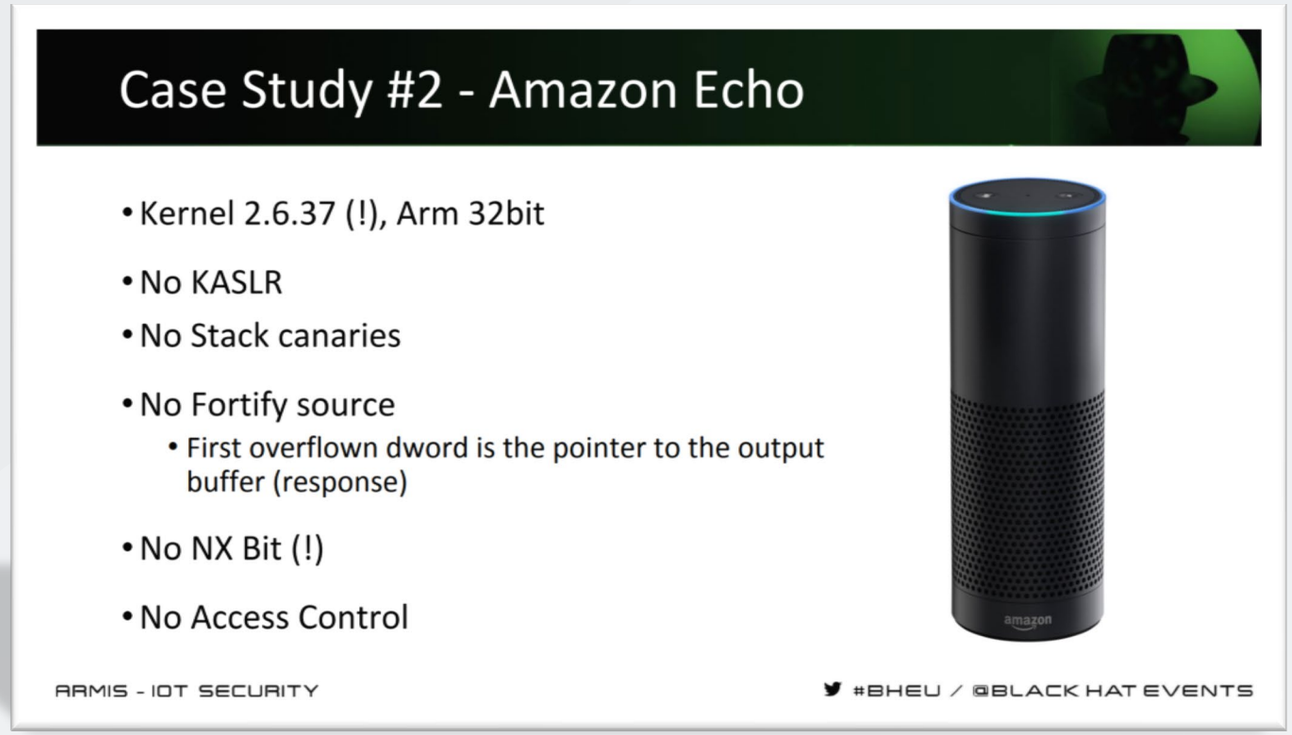
By Cisco Talos: Vulnerability Spotlight: Multiple Vulnerabilities in Samsung SmartThings Hub

VULNERABLE *SMART HOMES*

Amazon Echo

Presentation at Black Hat EU 2017 mentioned lack of exploit mitigations in Amazon Echo:





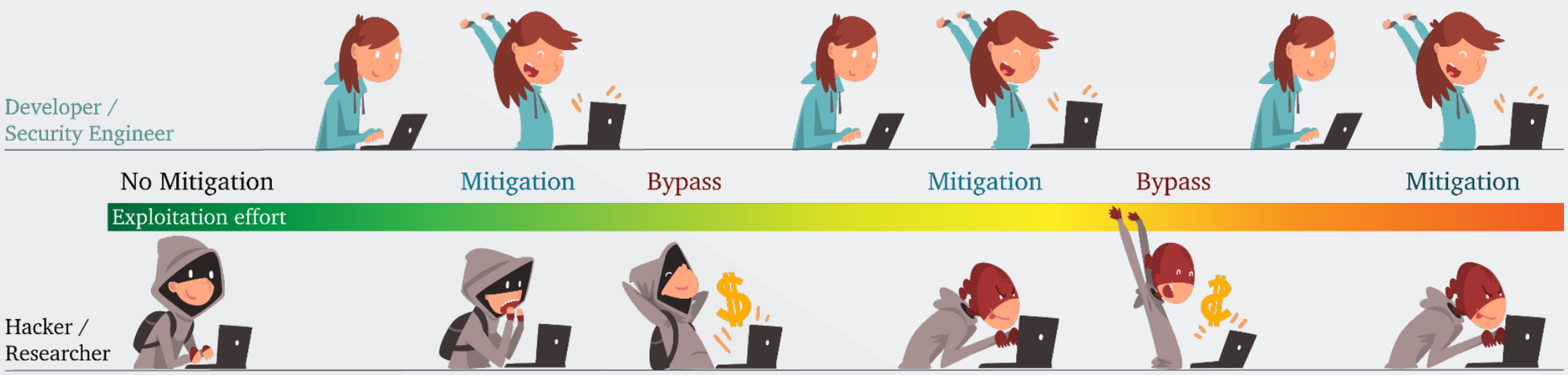
BLUEBORNE - A New Class of Airborne Attacks that can Remotely Compromise Any Linux/IoT Device

EXPLOIT MITIGATIONS

For every mitigation, hackers try to find bypass techniques.

Every new mitigation tries to mitigate these techniques in order to prevent exploitation of existing vulnerabilities or at least increase the effort for attackers.

Combining mitigations is important as they often mitigate against different exploitation techniques.



EXPLOIT MITIGATION IMPLEMENTATION

Paper published in 2019 lists which embedded operating systems implement three well-known exploit mitigations.

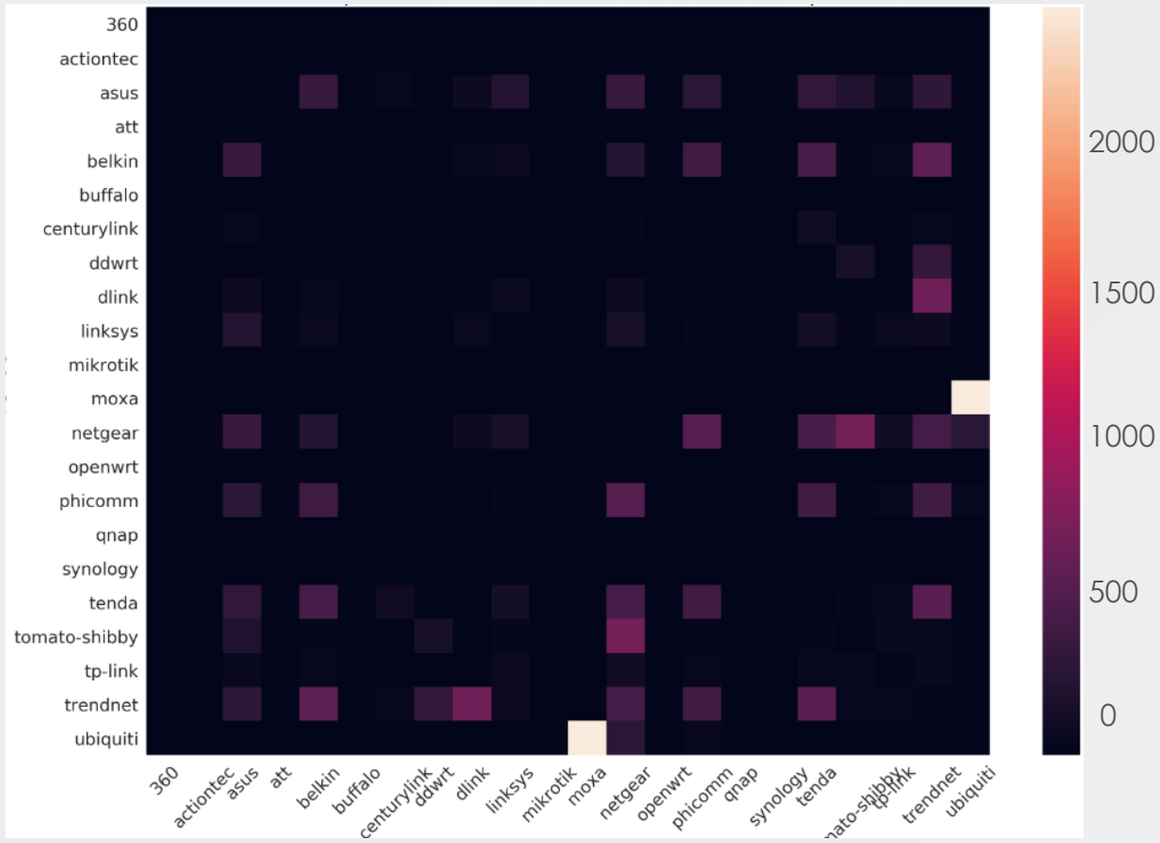
Note: Implementation on OS level only means that these mitigations are supported. Developers must compile their software with these mitigations in order to make use of them.

- Regular embedded OS
- Mobile embedded OS
- Deeply embedded OS

OS	ESP	ASLR	Canarie	OS	ESP	ASLR	Canaries
BlackBerry OS	✓	✓	✓	Android*	✓	✓	✓
iOS*	✓	✓	✓	Win 10 Mob.*	✓	✓	✓
Sailfish OS*	✓	✓	✓	Tizen*	✓	✓	✓
Ubuntu Core*	✓	✓	✓	Brillo*	✓	✓	✓
Yocto Linux*	✓	✓	✓	Windows Embedded*	✓	✓	✓
OpenWRT*	✓	✓	✓	Junos OS*	✓	×	✓
μClinux*	✓	×	✓	CentOS*	✓	✓	✓
NetBSD*	✓	✓	✓	IntervalZero RTX*	✓	×	✓
ScreenOS	×	×	×	Enea OSE	×	×	×
QNX	✓	✓	✓	VxWorks	✓	×	×
INTEGRITY	✓	×	×	RedactedOS [2]	×	×	×
Cisco IOS	×	×	×	eCos	×	×	×
Zephyr	✓	×	✓	ThreadX	×	×	×
Nucleus	×	×	×	NXP MQX	×	×	×
Kadak AMX	×	×	×	Keil RTX	×	×	×
RTEMS	×	×	×	freeRTOS	×	×	×
Micrium μC/OS [1]	✓	×	×	TI-RTOS	×	×	×
DSP/BIOS	×	×	×	TinyOS	×	×	×
LiteOS	×	×	×	RIOT	×	×	×
ARM mbed	✓	×	×	Contiki	×	×	×
Nano-RK	×	×	×	Mantis	×	×	×

A. Abbasi, J. Wetzels, T. Holz and S. Etalle, "Challenges in Designing Exploit Mitigations for Deeply Embedded Systems," *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, Stockholm, Sweden, 2019, pp. 31-46.

EXPLOIT MITIGATIONS
IN VENDOR BINARIES



DUPLICATE BINARIES

CITL showed how common duplicate binaries were between different vendors.

Data from CITL Report: State of Binary Hardening features in IoT Firmware

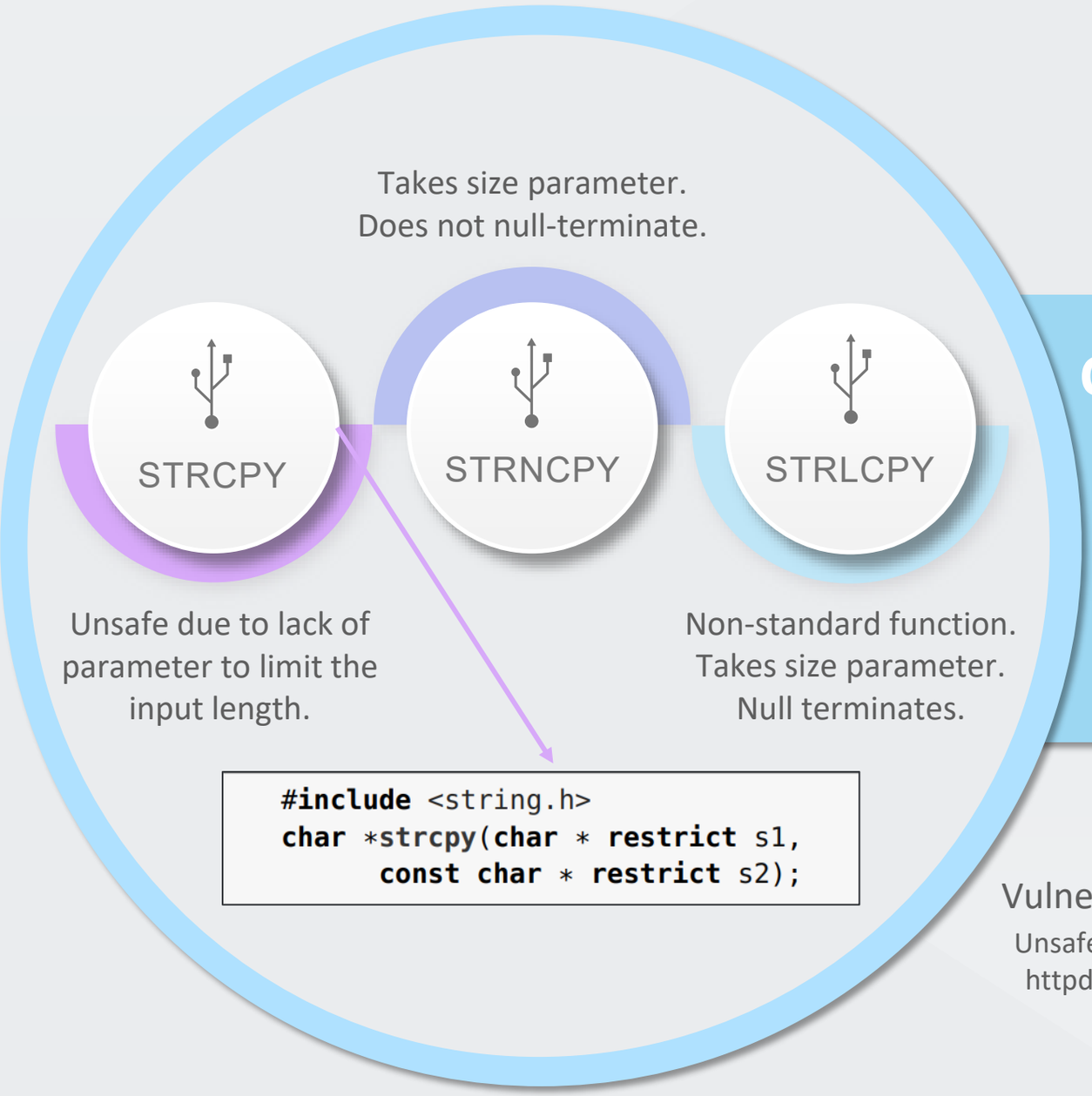
2018 Each column indicates the % of binaries shipped by that vendor with the given feature.

VENDOR	NON-EXEC STACK	ASLR	STACK GUARDS	FORTIFY	RELRO	COUNT
ASUS	49.70	1.76	2.40	0.08	2.96	234
BELKIN	0.00	0.75	0.00	0.00	1.76	3
BUFFALO	65.56	0.00	2.20	0.00	65.08	3
DDWRT	98.94	1.17	3.04	0.00	58.10	208
DLINK	42.84	0.65	0.08	0.86	7.42	14
LINKSYS	39.98	1.53	16.22	0.00	22.05	10
MIKROTIK	45.76	0.00	0.00	0.00	2.88	24
MOXA	78.12	11.98	9.86	6.64	19.43	57
OPENWRT	99.59	0.00	32.00	0.00	98.72	14
PHICOMM	59.62	3.58	0.00	0.00	11.44	5
QNAP	99.59	7.48	68.29	1.23	1.56	22
TENDA	24.95	0.60	0.95	0.00	7.13	16
TP-LINK	16.52	0.00	0.86	0.05	6.19	12
TRENDNET	30.61	8.70	18.09	0.39	27.81	23
UBIQUITI	24.74	0.34	1.68	5.88	20.30	298

BINARY HARDENING

Percentage of binaries shipped by a given vendor with different build-hardening features.

VULNERABLE *STRCPY*



Cisco Router Buffer Overflow CVE-2019-166

Pre-authentication buffer overflow vulnerability in unsafe function **strcpy** in the httpd webserver binary on several Cisco router series.

[Blog article: Cisco RV130 – It's 2019, but yet: strcpy]

Vulnerable Function

Unsafe function strcpy in httpd webserver binary.

Affected Devices

Cisco's router series: RV110W, RV130W, RV215W

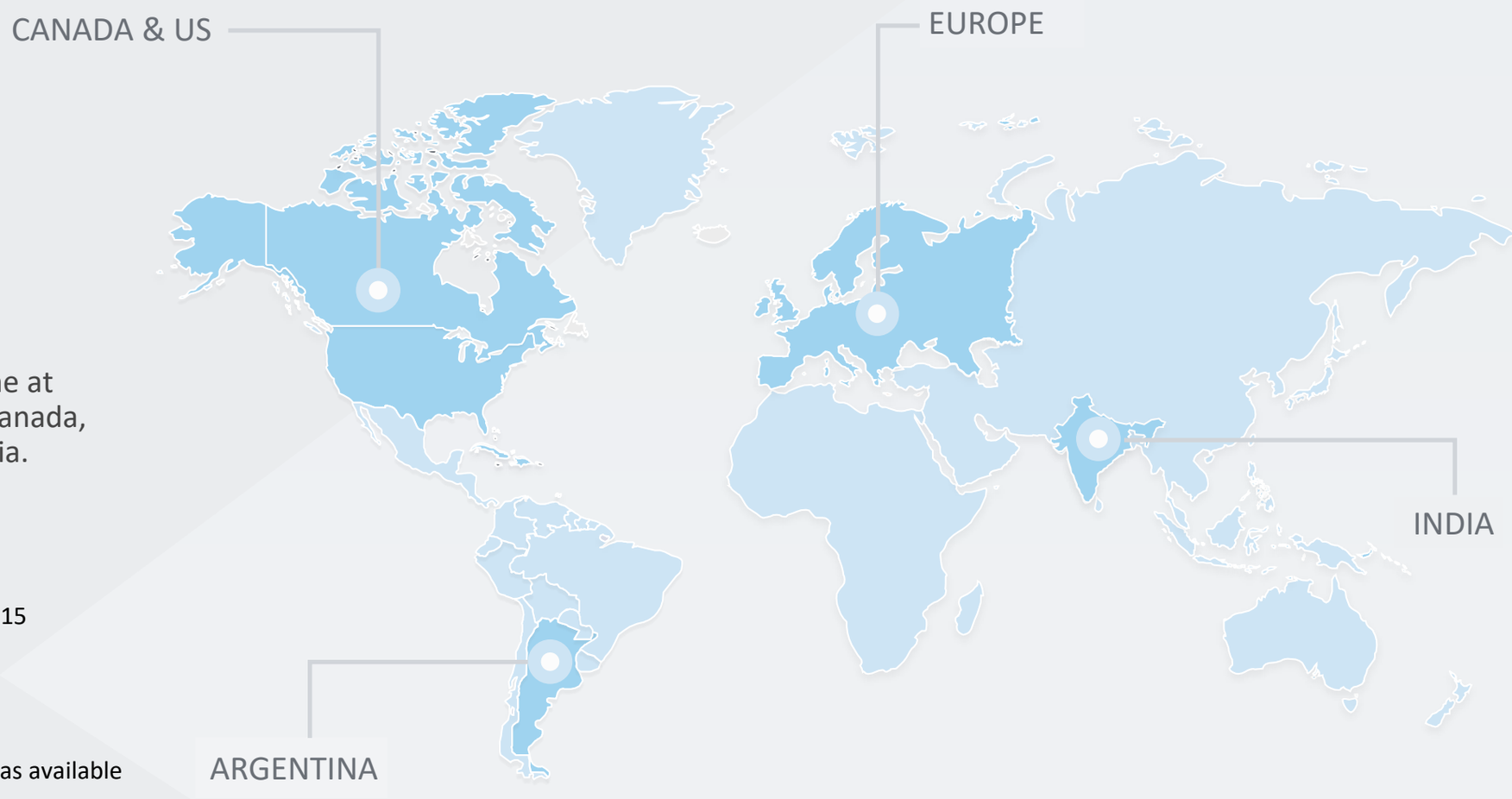
Vulnerability

Buffer overflow due to missing input size limits.

Discovery and Patch

Discovered by PenTest Partners, patched February 27, 2019.

EXPLOITATION *CVE-2019-166*

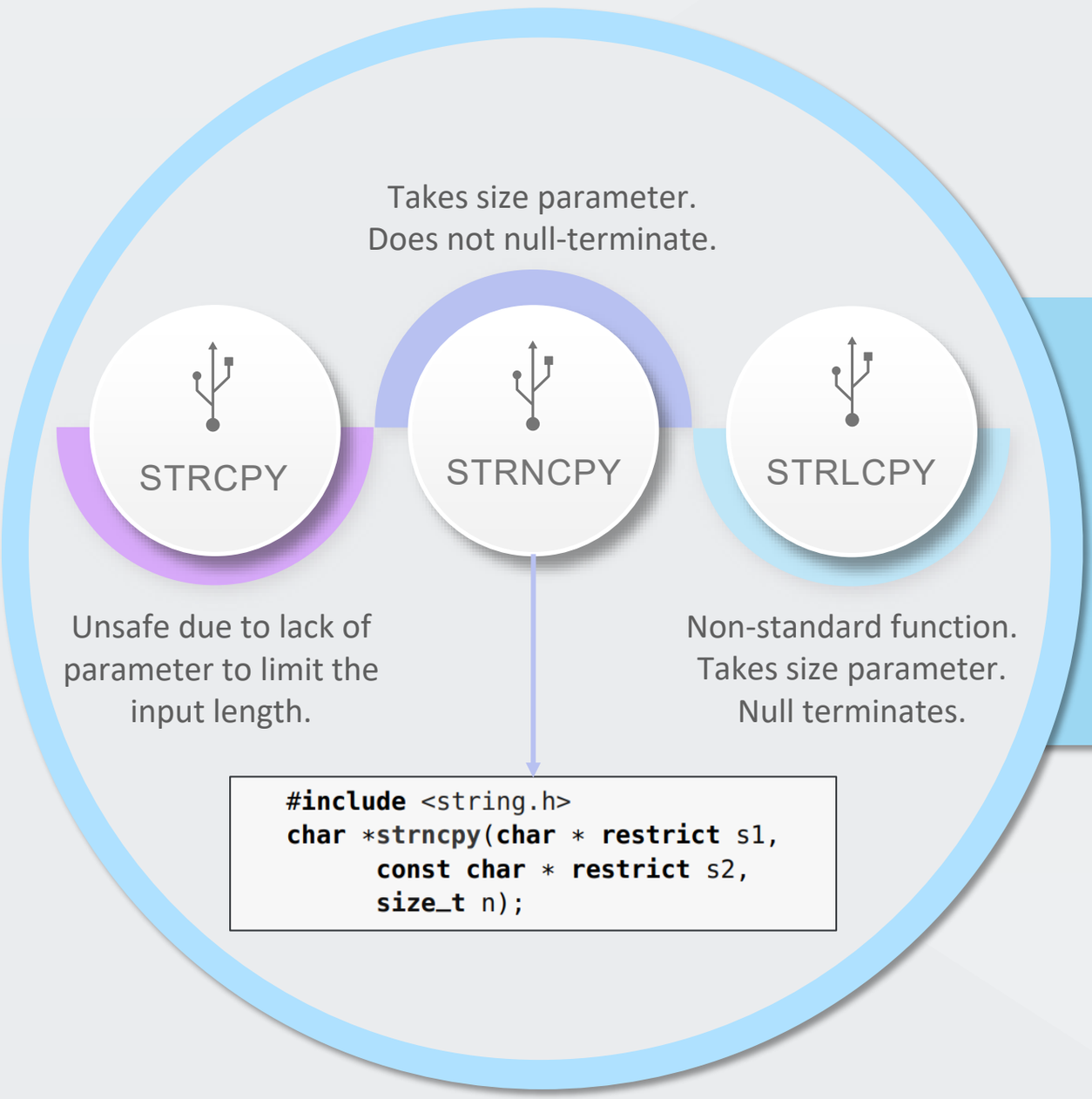


According to Rapid7, around 12,000 affected devices were available online at the time, the predominantly in US, Canada, Poland, Romania, Argentina, and India.

- Affected: Cisco RV110, RV130, and RV215
- 12,000 affected devices online
- Attacks started **two days after patch** was available

https://blog.rapid7.com/2019/02/28/cisco-r-rv110-rv130-rv215-unauthenticated-configuration-export-vulnerability-cve-2019-1663-what-you-need-to-know/
https://www.zdnet.com/article/hackers-have-started-attacks-on-cisco-rv110-rv130-and-rv215-routers/

VULNERABLE *STRNCPY*



D-Link DIR Buffer Overflow CVE-2016-6563

Pre-authentication buffer overflow vulnerability in strncpy that occurs because the argument to **strncpy** is the **strlen** of the input value.

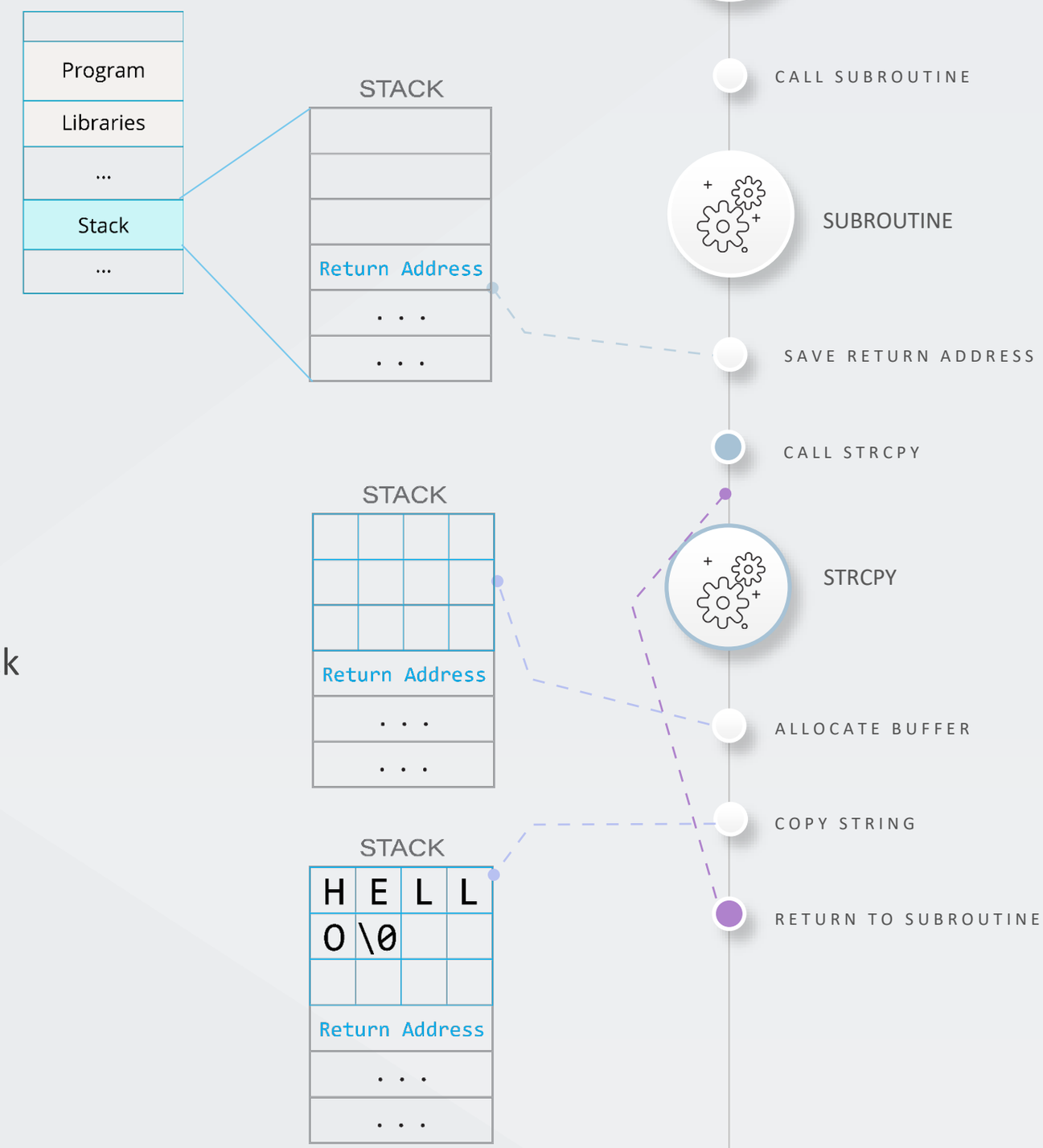
UNSAFE *FUNCTIONS*

`strcpy(destination, source);`

```
#include <string.h>
char *strcpy(char * restrict s1,
             const char * restrict s2);
```

Copies the string pointed to by s2 into the array pointed to by s1.

1) Return address is saved onto the Stack
2) Buffer is allocated
3) Input string is copied into the buffer

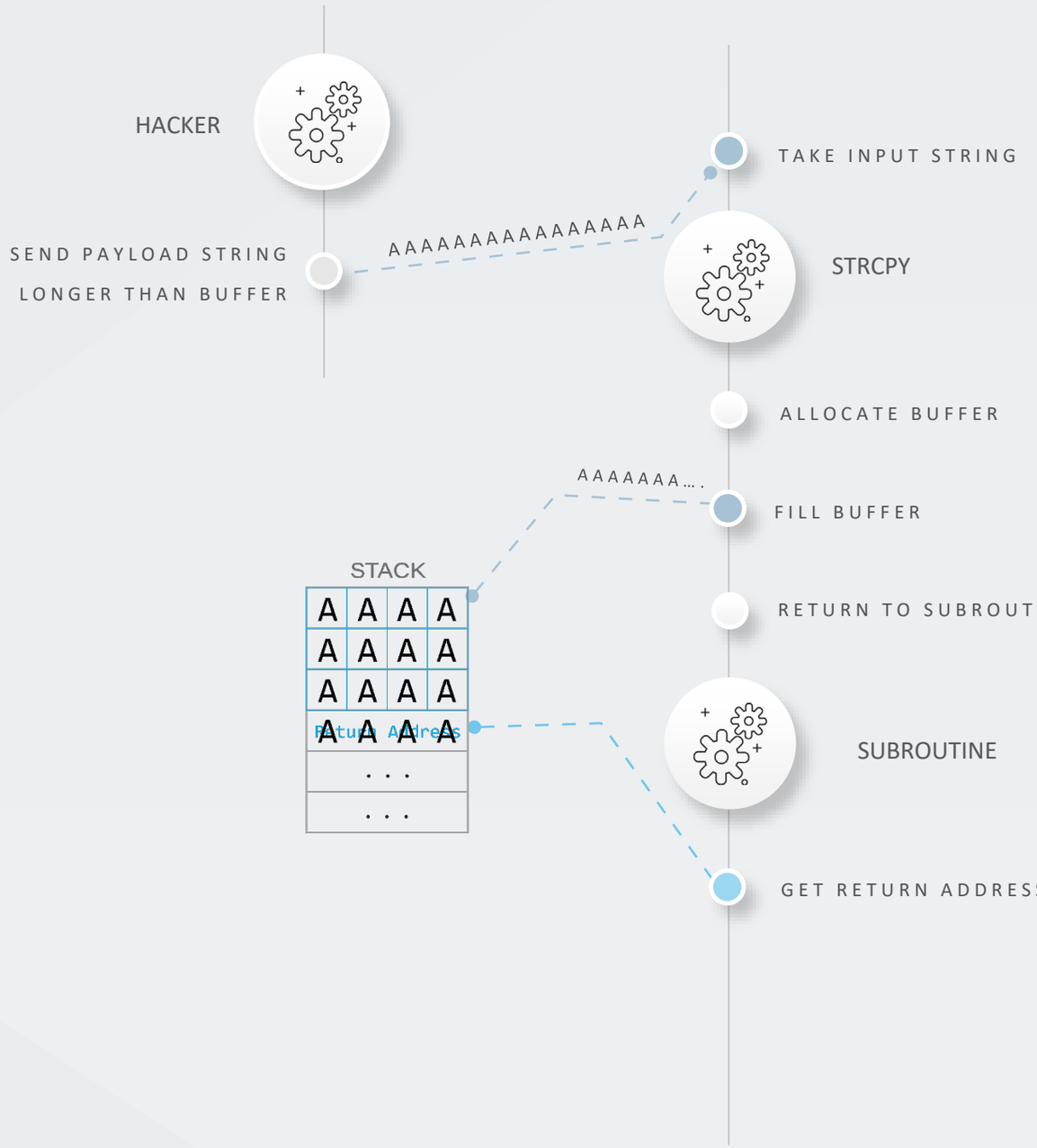


BUFFER OVERFLOW
VULNERABILITIES

WHAT HAPPENS WHEN INPUT IS LARGER THAN BUFFER?

A Buffer Overflow occurs when a function defines a data array as a local variable and fails to prevent input data from overflowing the allocated limits of its buffer.

If the overflowing data corrupts nearby local variables and critical control-flow data, such as a return address saved onto the stack, an attacker can use this vulnerability to seize control of program flow.



BUFFER OVERFLOW
VULNERABILITIES

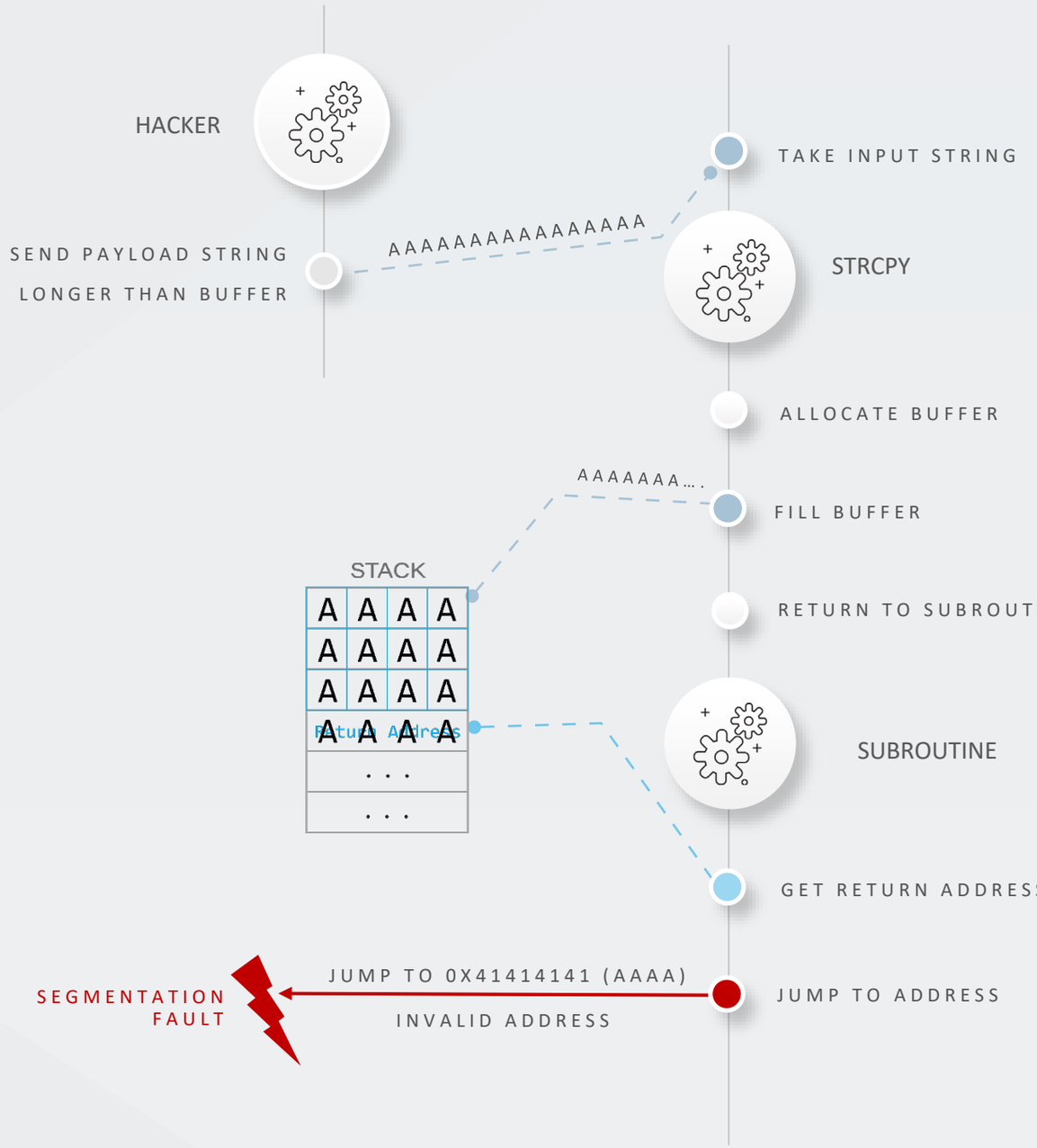
HOW DO BUFFER OVERFLOWS WORK?

In this example, the input string consists of A's. In memory, characters are translated into hex: AAAA = 0x41414141

The function expects the return address to be at an exact position on the stack. Based on that assumption, any value at that position will be written to the Program Counter, which executes the instruction pointed to by this value (address).

But 0x41414140 this is not a valid address.

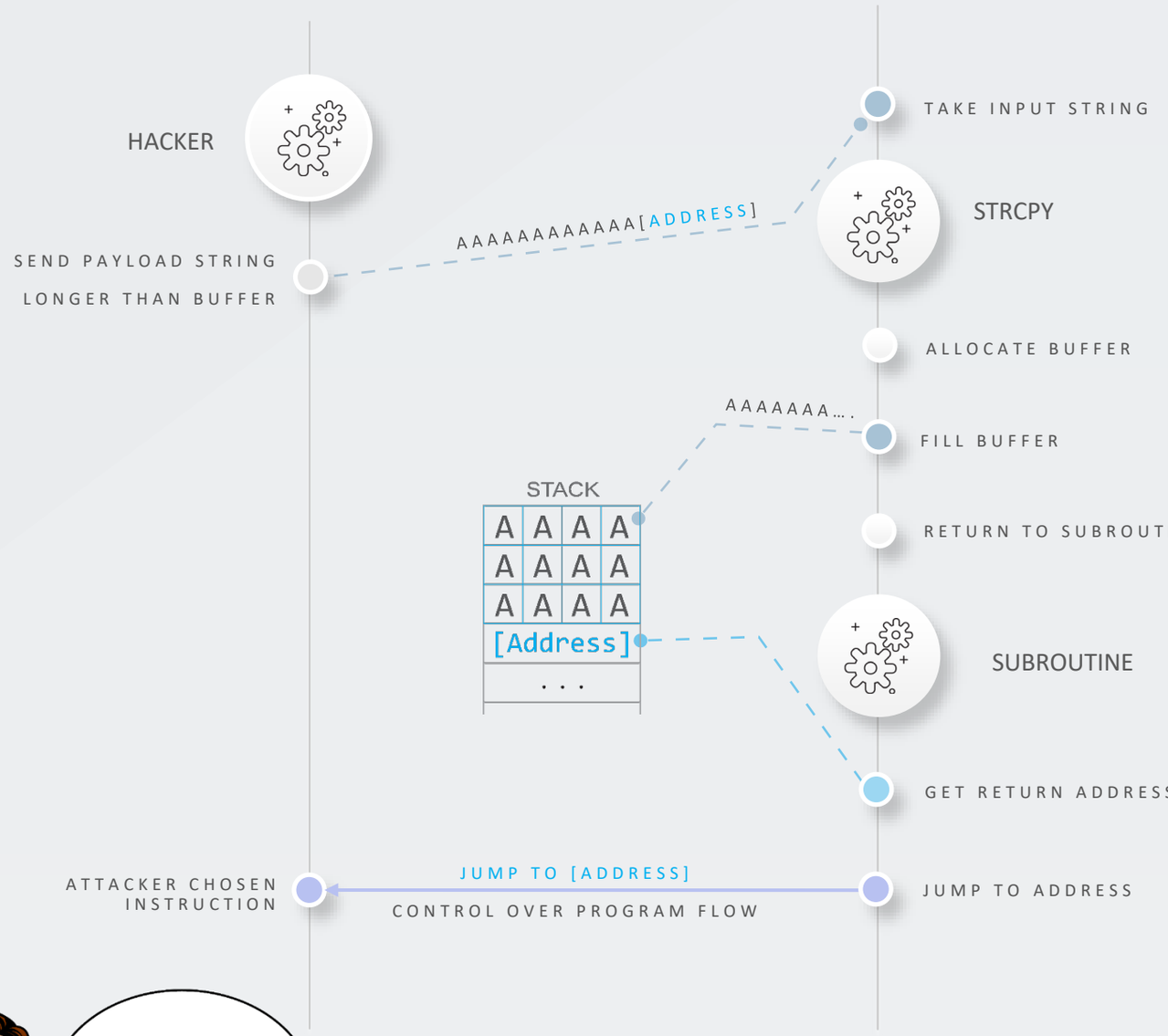
The program crashes: **Segmentation Fault.**



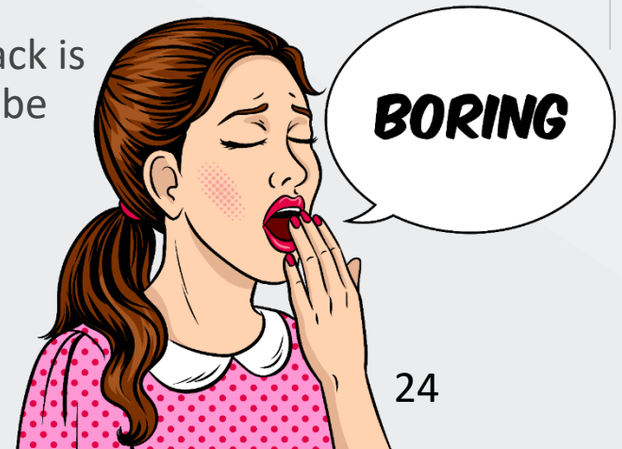
BUFFER OVERFLOW *VULNERABILITIES*

WHAT ARE BUFFER OVERFLOWS?

1. Pick an instruction from an executable section in process memory (e.g. libc library)
2. Append address of instruction to the input string at exact position of return address
3. Return address is overwritten with new address pointing to different instruction
4. Program fetches the return address from the stack and jumps to it.



Without XN exploit mitigation, the stack is executable, and shellcode can simply be executed on the stack.

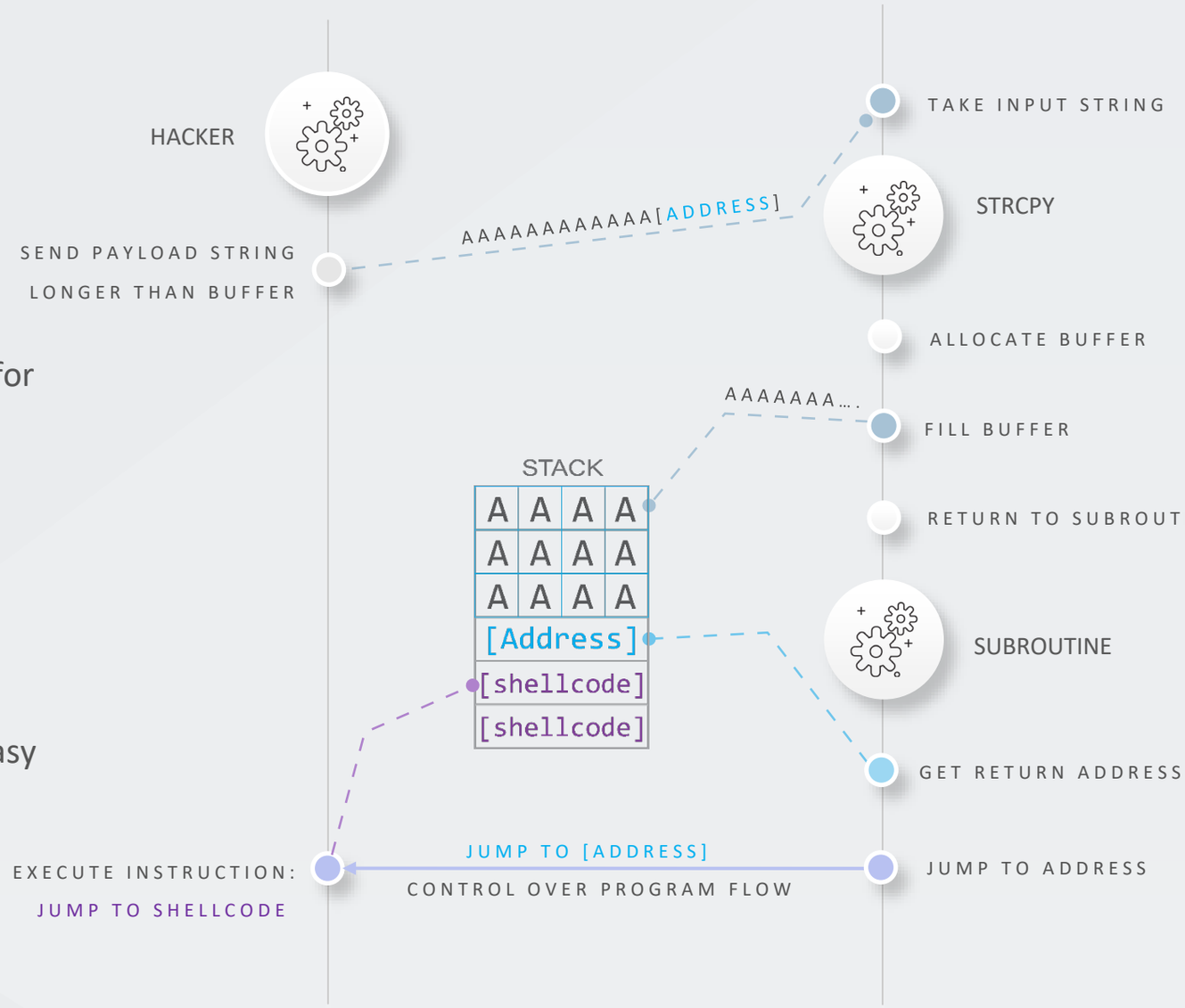


BUFFER OVERFLOW
VULNERABILITIES

Researcher Andy Nguyen created a fully chained exploit for the PS Vita™ consisting of six vulnerabilities.

One of the components (MIPS Kernel) was particularly easy to exploit because it came without exploit mitigations.

> Do we have to bypass any security mitigations? Nope, there are none! Zero!



Shellcode: Opcode string of a sequence of instructions that achieve a goal. For example, a reverse shell that establishes a connection to a remote host for remote shell access to the target device.

EXECUTE NEVER (XN) *MITIGATION*

How to prevent shellcode execution on the stack

XN marks certain memory areas as Not Executable, including the Stack.

The vulnerability still exists; i.e. the return address can still be overwritten to redirect control-flow.

But when the processor attempts to fetch instructions the stack a translation fault will occur because the stack's memory is marked non-executable.

Purpose

Marks certain memory areas, including the stack, as Not Executable.

Prevents

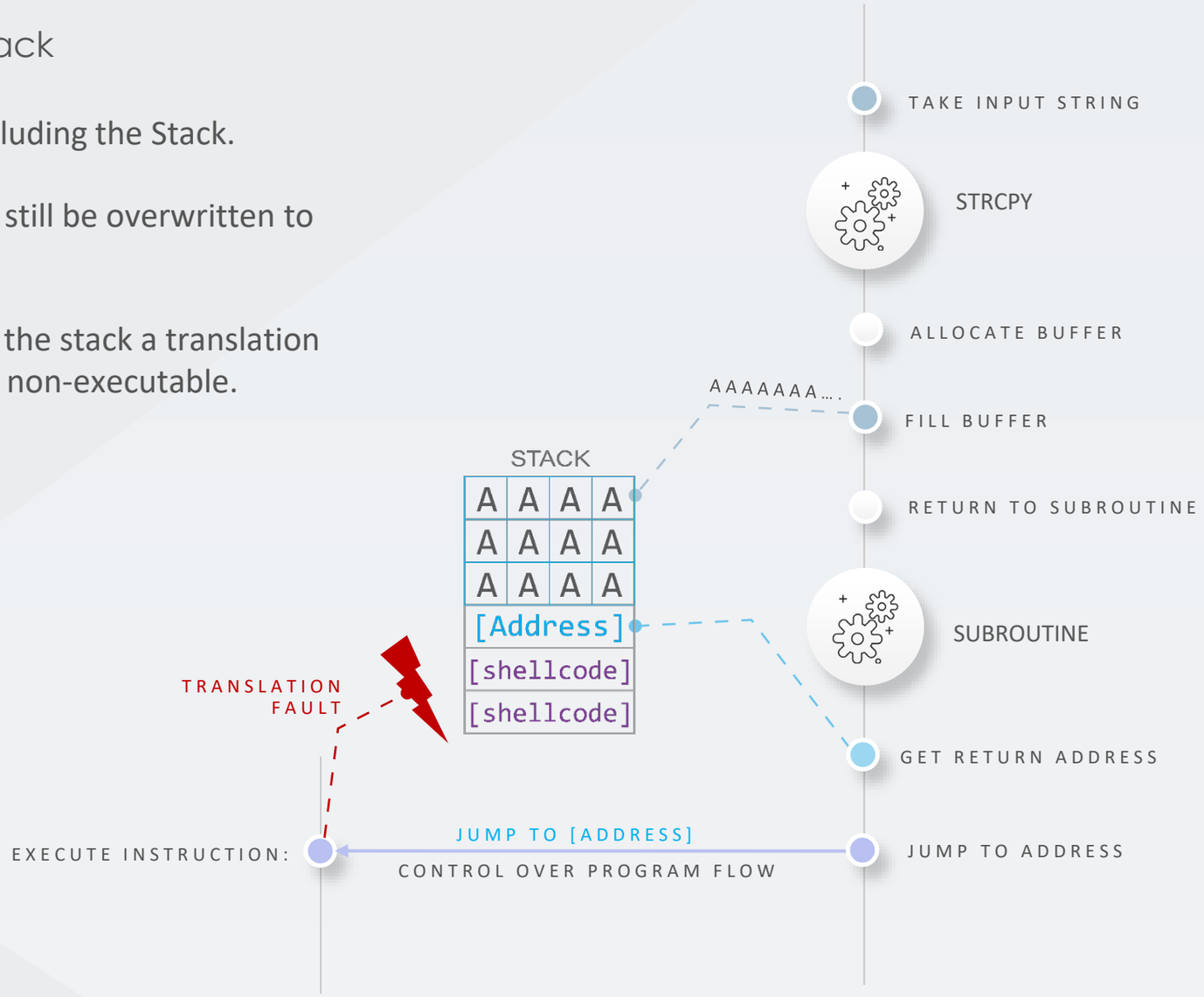
Executing shellcode on the stack, making exploitation less trivial.

WITHOUT ASLR

The addresses of ROP gadgets are static, making the exploit more reliable.

XN MITIGATION

The XN mitigation prevents execution from certain memory regions, including stack.



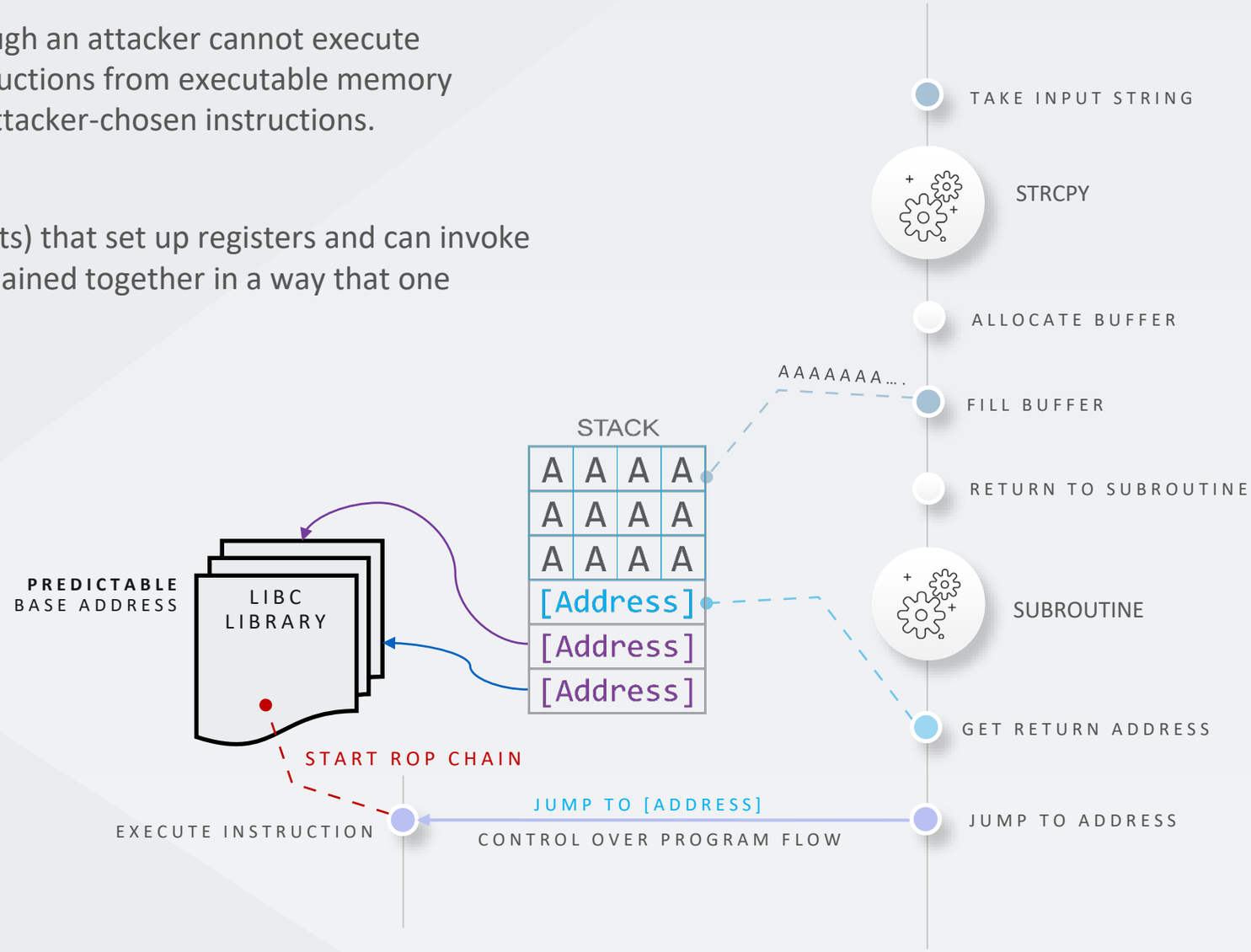
EXECUTE NEVER (XN)
B*YPASS*

ROP is based on the observation that although an attacker cannot execute code on XN-marked pages (e.g. Stack), instructions from executable memory (e.g. libraries) can still be used to execute attacker-chosen instructions.

ROP chain: addresses of instructions (gadgets) that set up registers and can invoke library functions or API calls. Gadgets are chained together in a way that one gadget ends with triggering the next.

XN BYPASS

ROP chain using gadgets in the victim process' address space to achieve goal



ASLR *MITIGATION*

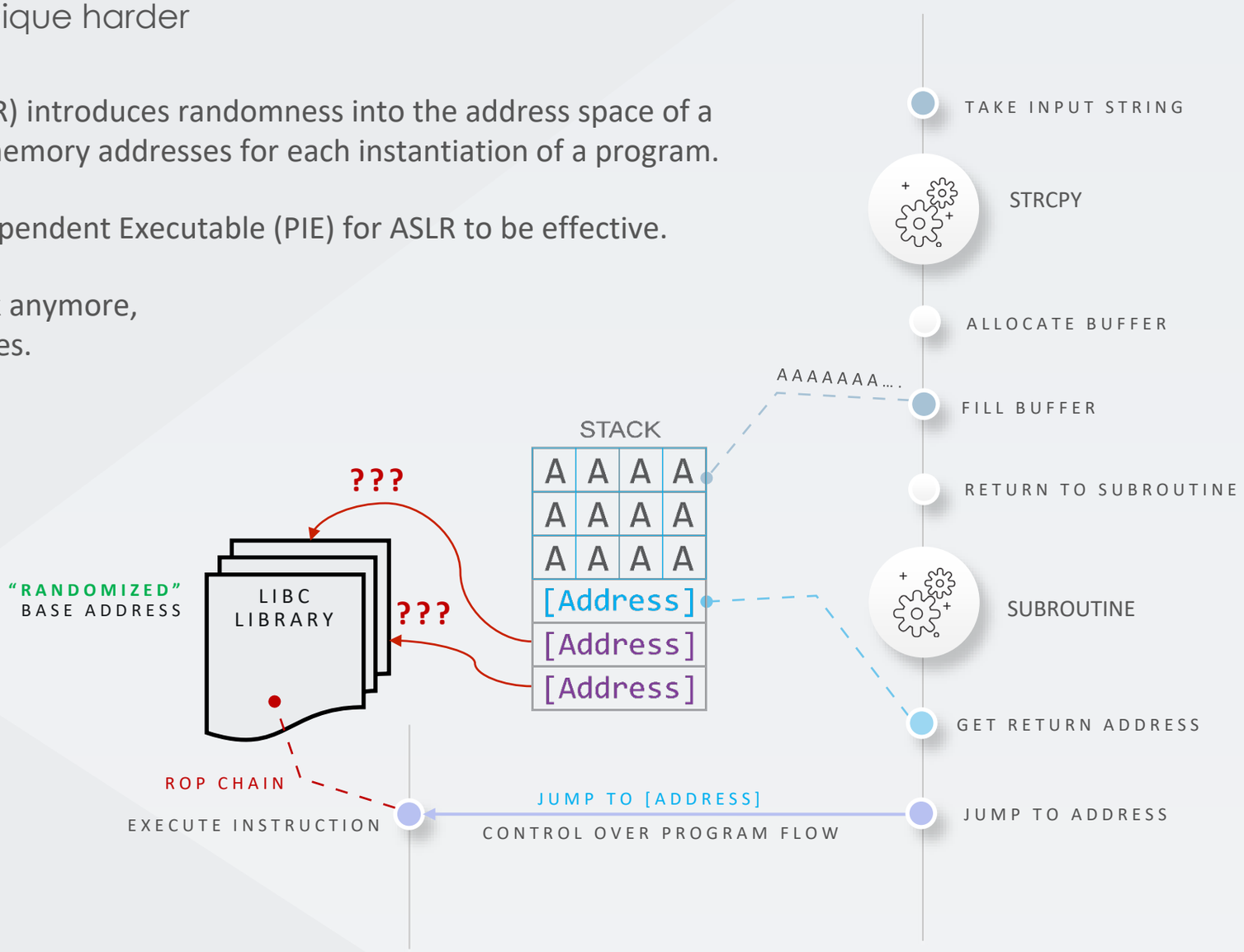
> When it comes to MIPS, I usually encounter only one mitigation, ASLR or NX. Only this year I've seen MIPS routers with both enabled.
>
> - @B1ACK0WL

How to make the ROP bypass technique harder

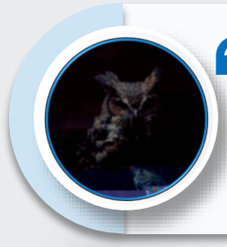
Address Space Layout Randomization (ASLR) introduces randomness into the address space of a process and reduces the predictability of memory addresses for each instantiation of a program.

Binaries must be compiled as Position Independent Executable (PIE) for ASLR to be effective.

Previously chosen ROP gadgets won't work anymore,
since the base address of the library changes.



ASLR *BYPASS*



> ASLR played a big role in complicating this exploit since the overflow was a one-time shot. If it fails, there is no second chance.
>
> - @B1ACK0WL

Bypassing ASLR: Brute Force

This technique relies on the low entropy of 32-bit environments where not all bits of an address are randomized. Only viable if process restarts after crash.

Bypassing ASLR: Information Leaks

This bypass involves the use of information leak exploit primitives to leak relevant addresses from the victim process.

Infer the address of a loaded binary and dynamically set the correct base address for ROP gadgets.

Various other bypass techniques exist, such as:

Spray-Based Attacks

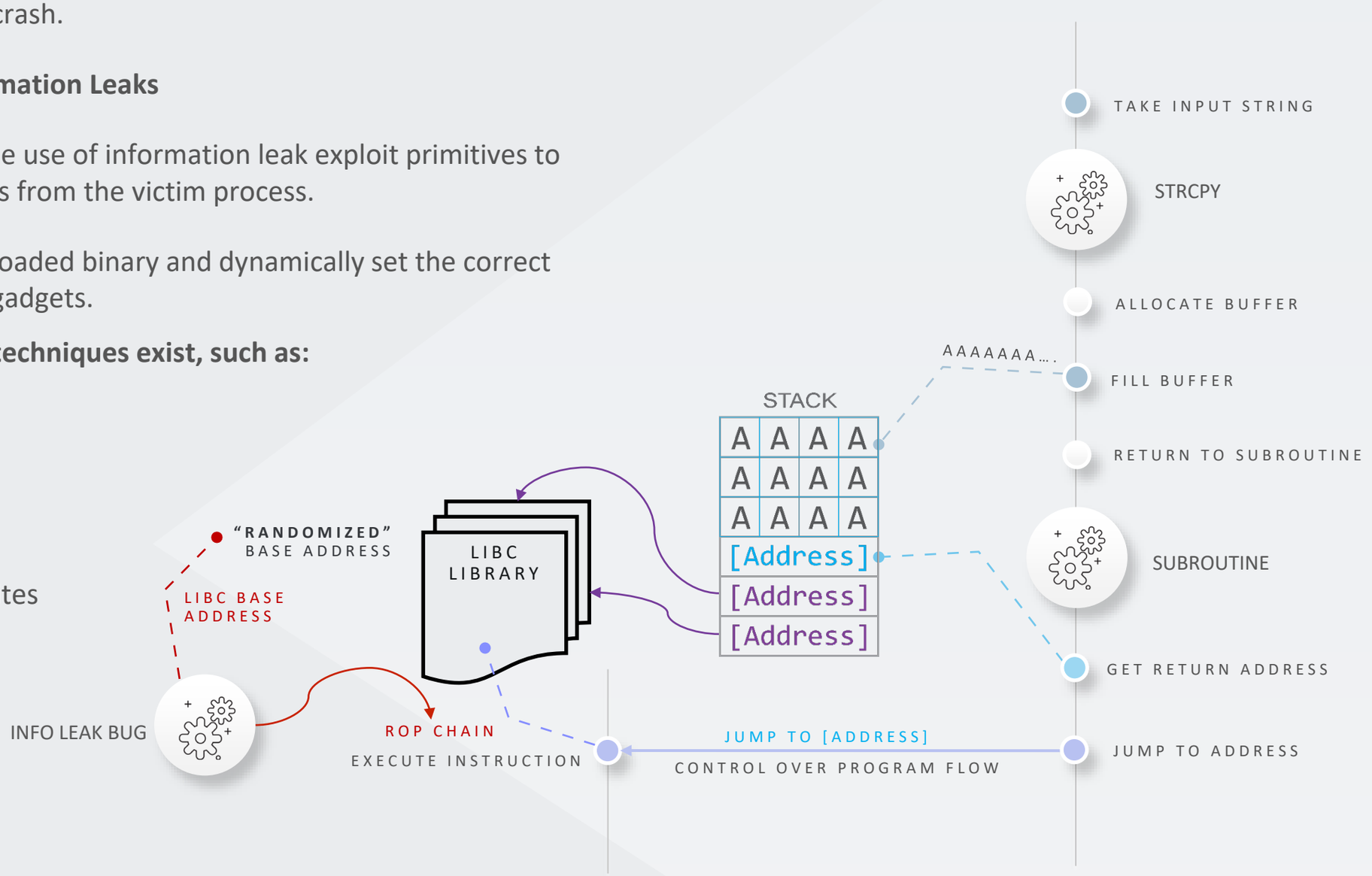
JIT Spray

Offset2lib attacks

Partial Pointer Overwrites

Side-Channel Attacks

. .

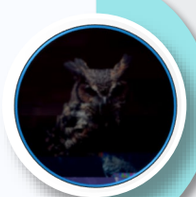


ASLR MITIGATION
BYPASS EFFORT

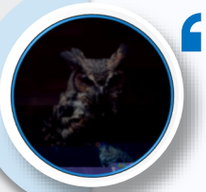
Effort of Bypassing ASLR

The effort and chosen bypass technique depends on various factors, such as the environment, and vulnerability context. Often the bypass requires an additional vulnerability to succeed.

> For my UPNP vulnerability, if ASLR was disabled, then my exploit would've taken seconds to execute instead of 30 minutes.
>
> - @B1ACK0WL



Services that restart infinitely following a crash or are present on multiple devices make this attack more probable.



> ASLR played a big role in complicating this exploit since the overflow was a one-time shot. If it fails, there is no second chance.
>
> - @B1ACK0WL

In some cases, attackers don't manage to successfully bypass ASLR or spend significantly more time on this bypass than on the rest of the exploit.

XN and ASLR are complementary techniques. Both are designed to increase the effort and make the exploitation process harder.

ASLR BYPASS EFFORT

Time effort much greater and much less predictable than for XN bypass.

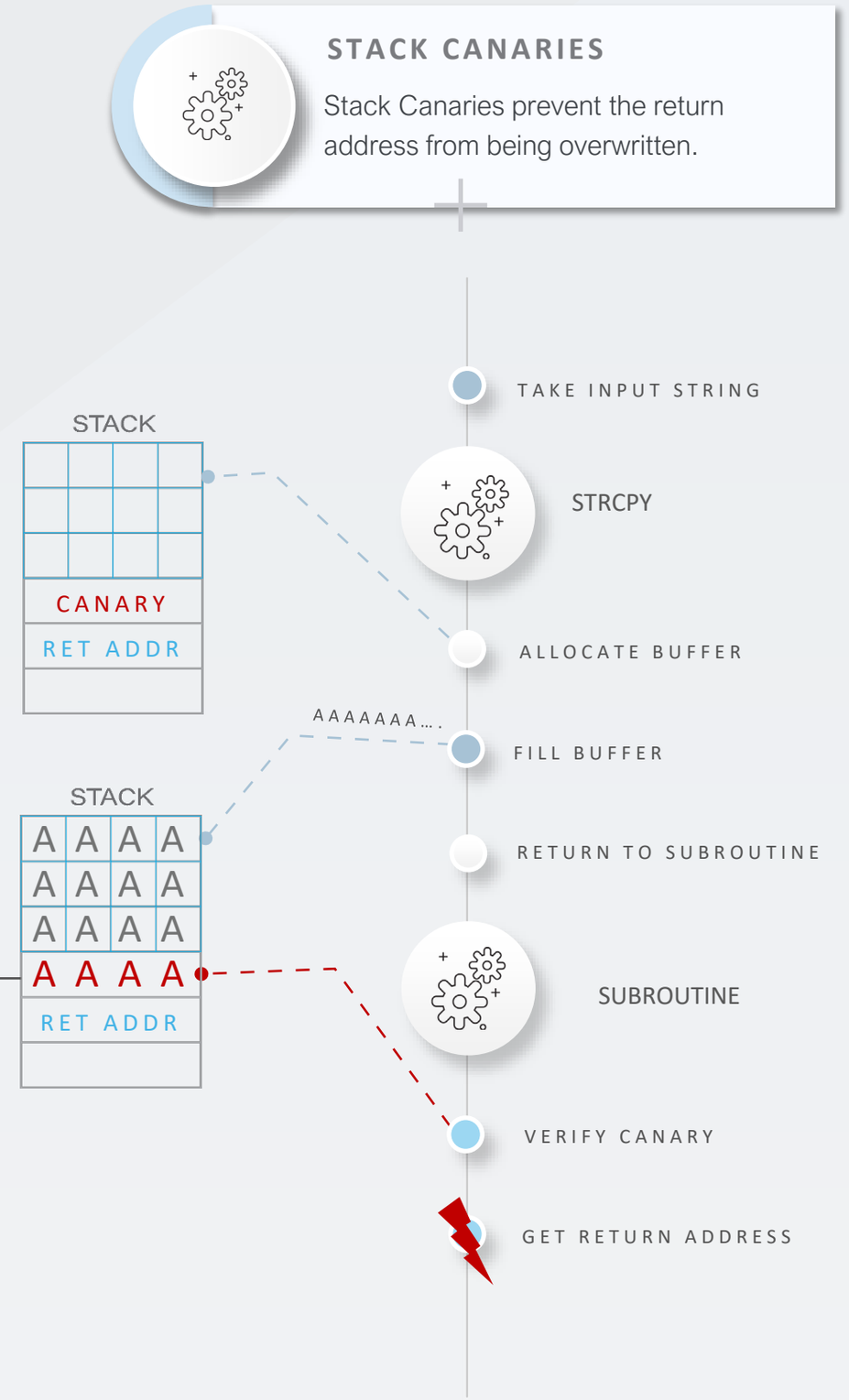
STACK CANARY
MITIGATION

Let's prevent the Return Address from being overwritten.

In our previous example, the saved return address on the stack could be overwritten to take control over the PC and initiate a ROP chain.

Stack Canaries attempt to prevent the return address from being overwritten in the first place. Most stack-buffer overflows can't be used to start a ROP-chain without surgically inserting the stack canary value trough an information leak vulnerability.

A random value is inserted between variables and the return address on the stack. When the return address is accessed, the canary value is first validated by a stack canary check.



STACK CANARY *MITIGATION*

How Stack Canaries are bypassed.

One of the more common ways to bypass the stack canary validation is by using an information leak to overwrite the stack canary with its own value, making the corruption undetectable by the validation routine.

This significantly increases the exploitation effort and requires an additional vulnerability to leak the current canary value and insert it where it belongs.

PS Vita Exploit Chain: After finding the stack overflow, the hardest part was finding an information leak vulnerability to bypass Stack Canary and ASLR mitigations. This took the researcher one full week.

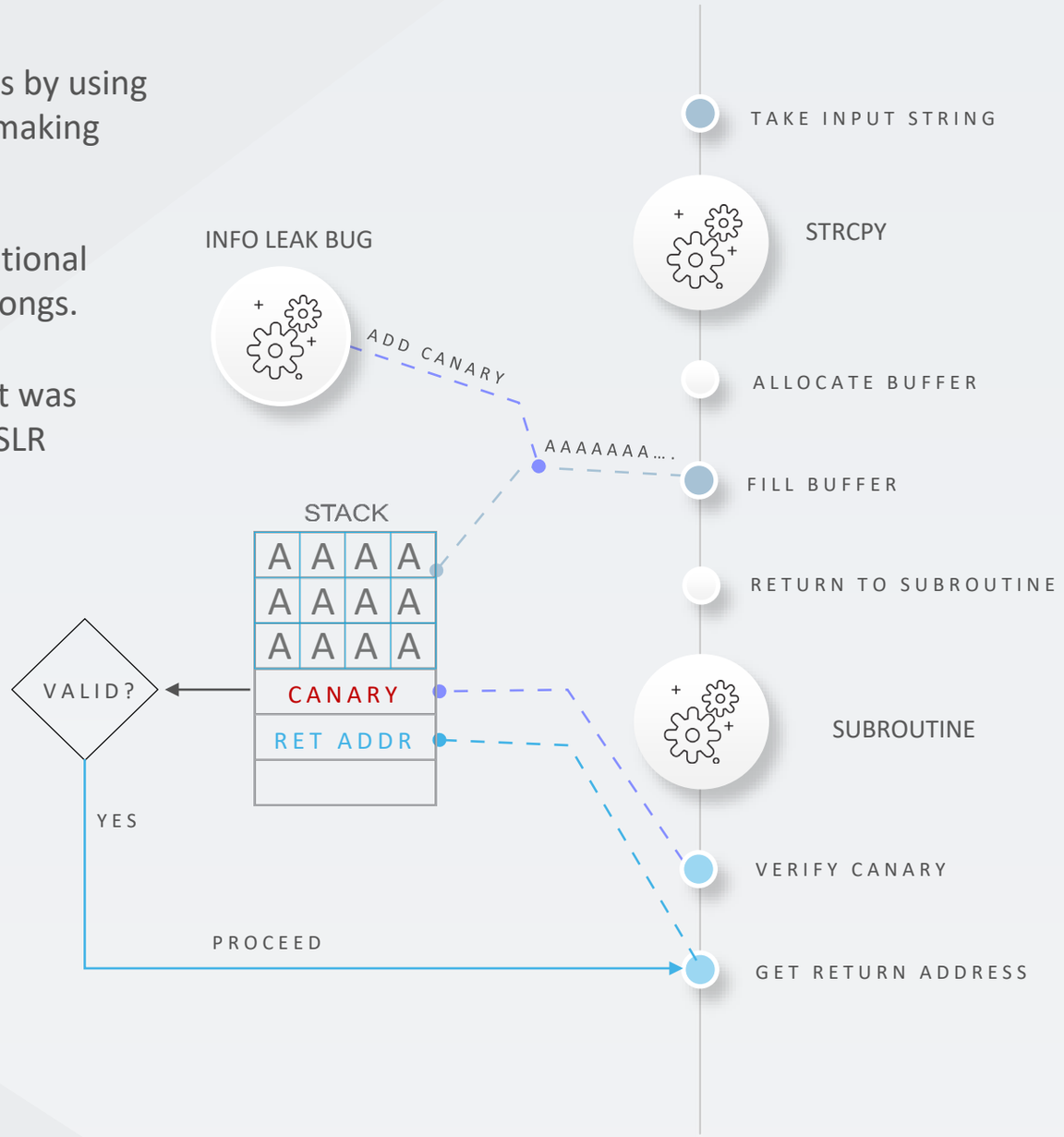
> The stack overflow was quickly discovered. Then, it took me about one full week searching for a stack cookie leak.
>
> - ANDY NGUYEN

This was probably the hardest part of the whole chain and took me a whole week to solve. I was looking for uninitialized buffers on stack that would leak the stack canary and return addresses to defeat ASLR. Surprisingly, I couldn't find such a bug and it seemed like Sony made sure not to forget to `memset()` any buffer there. After a whole week of reverse engineering and digging through all commands, I found a very cool bug that would allow us to read arbitrary memory. Discovered on the 2018-06-04.

Highly recommend reading the full write-up:
https://theofficialflow.github.io/2019/06/18/trinity.html

GCC Stack Canary and ASLR support:

```
-fpic                  // ASLR support
-fstack-protector=all  // stack canaries
```



STACK CANARY
MITIGATION

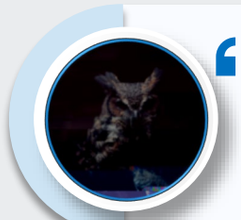
How Stack Canaries are bypassed.

Stack canaries don't protect against **overwriting adjacent local variables** and structures on the stack, especially those that are used as function pointers. This requires more precision and the right vulnerability context to exploit.

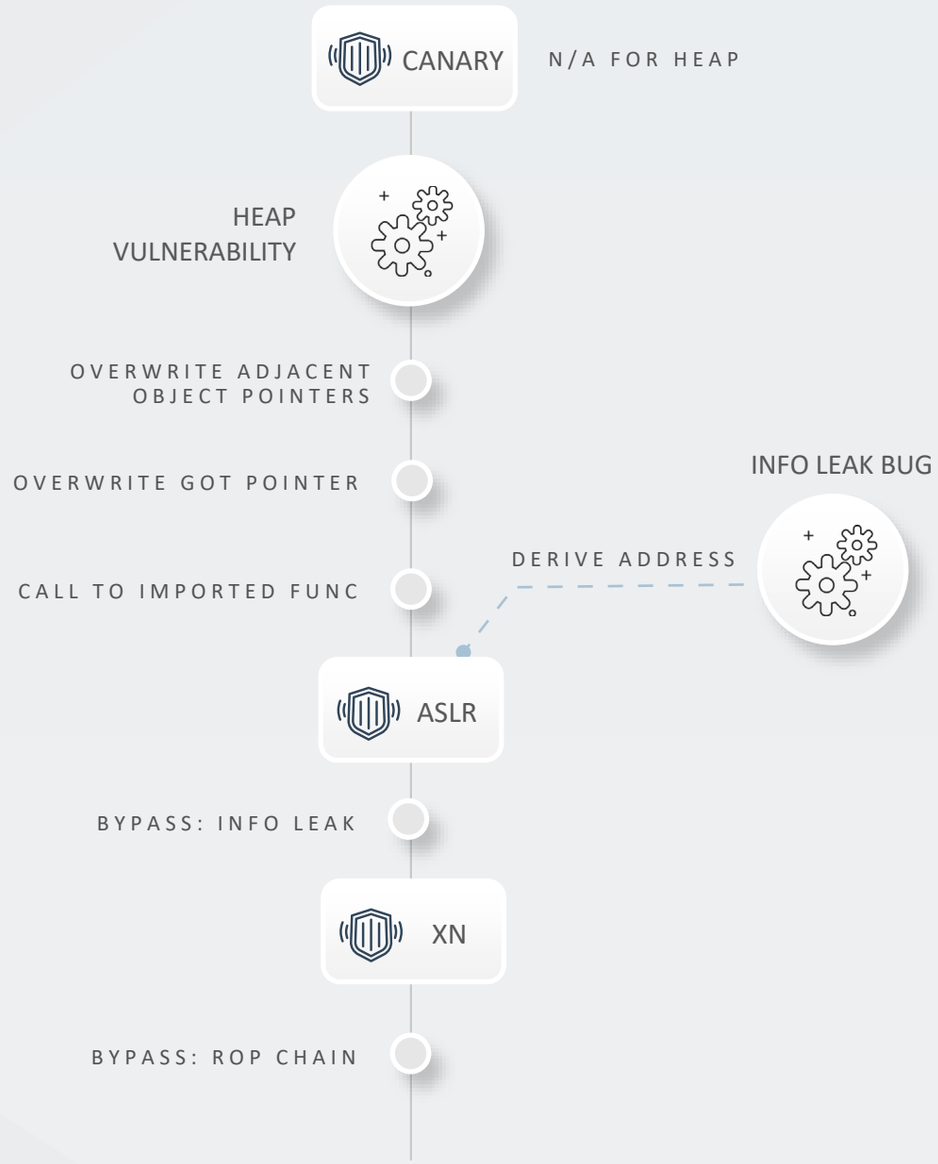
Heap-based vulnerabilities, such as UAFs and heap-overflow vulnerabilities are not mitigated by stack canaries.

Bypass strategies:
1) Use non-linear stack buffer overflow
2) Use an additional info leak to leak canary
3) Use a heap buffer overflow

Note: we are now significantly constraining the hacker's use of vulnerabilities to build an exploit.



> There's been times where I've thrown out an overflow because it was not possible to exploit without another bug.
>
> - @B1ACK0WL

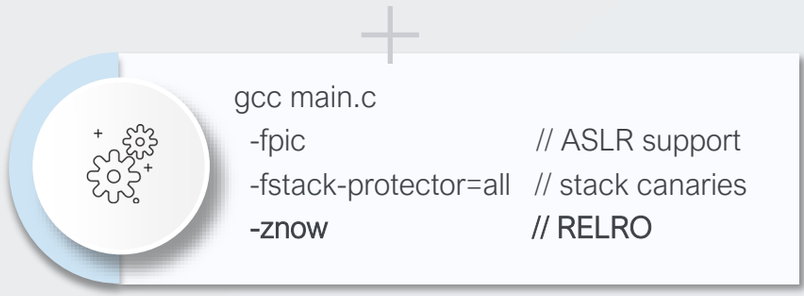


RELOCATION READ-ONLY
RELRO MITIGATION

When the program is loaded, the linker resolves imported symbols to an address, placing these addresses in the program's Global Offset Table (GOT)

With specific vulnerability types that lead to an arbitrary write primitive, hackers can overwrite GOT function pointers to hijack control over the program flow without overwriting the return address.

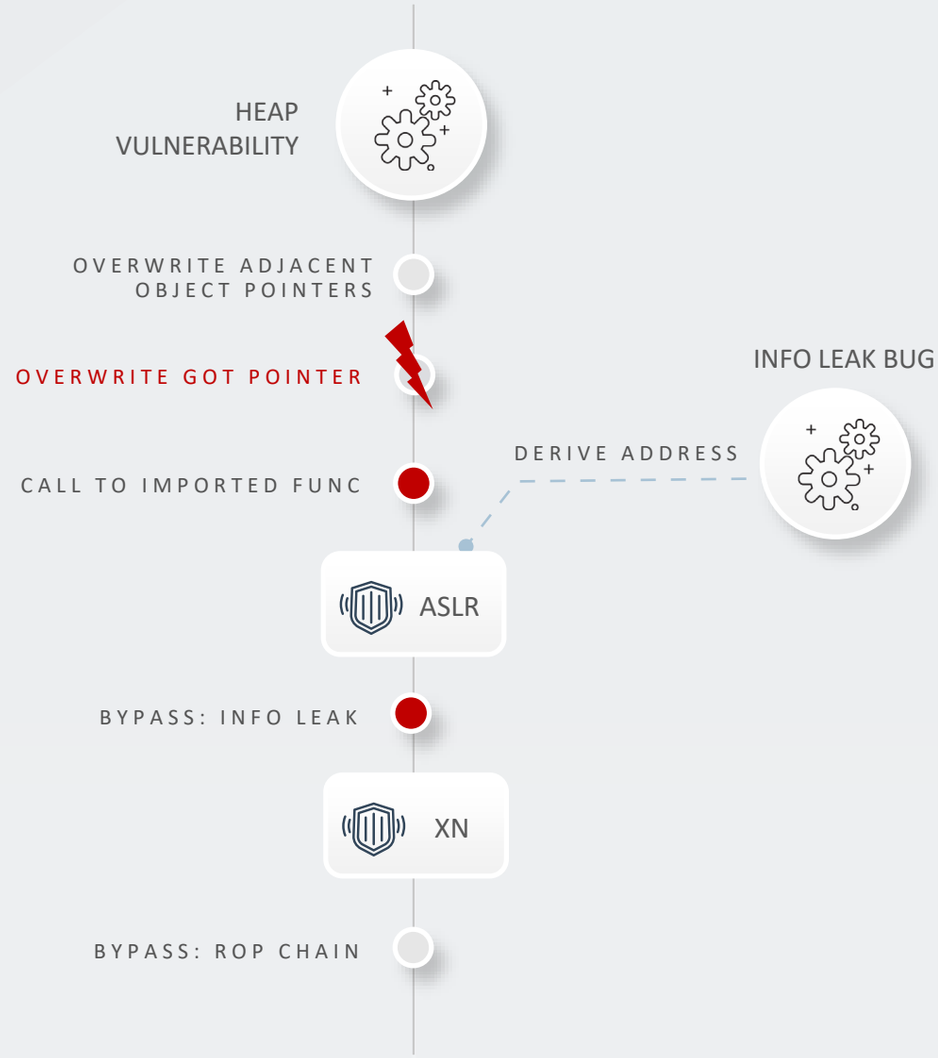
The Full RELRO mitigation proactively resolves the GOT entries and marks the GOT Read-Only to prevent entries from being overwritten.



NEXT GEN
MITIGATIONS

Armv8.3-A Pointer Authentication
Function-pointers, vtables, and return addresses are protected with a hardware-protected key that is very hard for attackers to leak.

Armv8.5-A Memory Tagging
Many use-after-free and buffer-overflow vulnerabilities cannot be used to subvert memory consistency.



ARE YOU GIVING HACKER *A HEADACHE?*

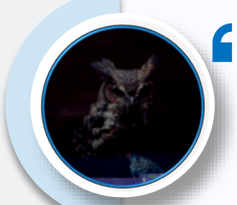
Researcher Andy Nguyen created a fully chained exploit for the PS Vita™ consisting of six vulnerabilities.

One of the components (MIPS Kernel) was particularly easy to exploit because it came without exploit mitigations.

> *[About MIPS Kernel in PS Vita]*
> Do we have to bypass any security mitigations?
> Nope, there are none! Zero!
>
> - ANDY NGUYEN

With exploit mitigations, the exploit development process can take weeks or even months (depending on the target).

> The stack overflow was quickly discovered. Then, it took me about one full week searching for a stack cookie leak.
>
> - ANDY NGUYEN



> There's been times where I've thrown out an overflow because it was not possible to exploit without another bug.
>
> - @B1ACK0WL

> The kernel heap overflow was also discovered quickly. [...] Overall, I think this took me 3-4 weeks.
>
> - ANDY NGUYEN

https://theofficialflow.github.io/2019/06/18/trinity.html

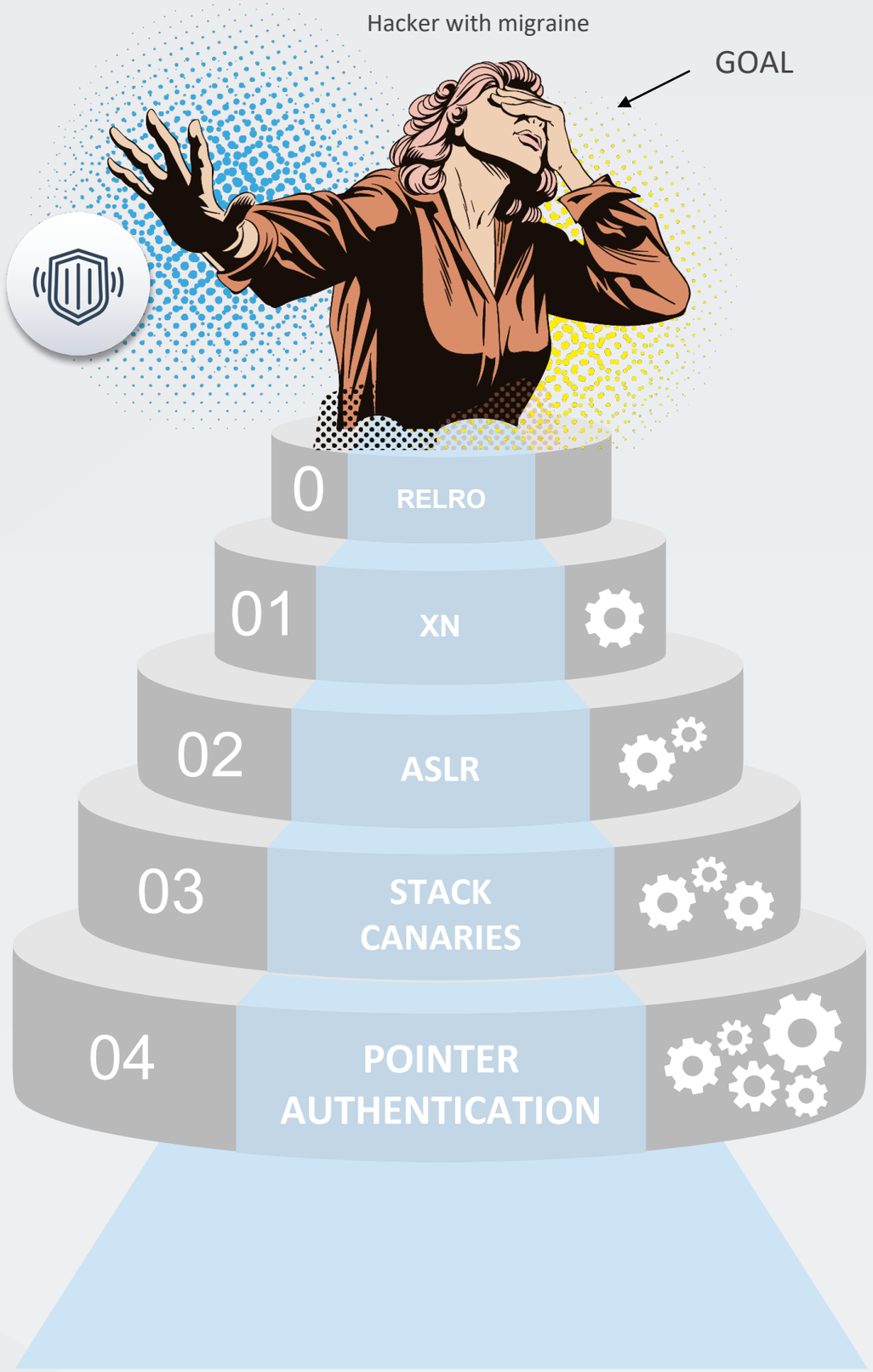
COMBINE.
MITIGATIONS.

DON’T BE A LOW HANGING FRUIT ☺

- Harden your device against most common attack vectors.
- Provide secure coding education to your developers.
- Conduct security reviews of source code.
- Hire hackers (Pentesters) to find overlooked vulnerabilities.
- Avoid unsafe functions in new software.
- Combine mitigations to make exploitation harder.

Each mitigation

- Protects against different exploitation techniques
- Makes exploitation more complicated
- Restricts which techniques hacker can use
- Increases time and effort for attackers significantly
- Can make exploits less reliable
- In some cases, can make hacker give up on a vulnerability



Website: https://azeria-labs.com/

Mailing list: https://arm-exploitation.com/

Email: contact@azeria-labs.com

Azeria
@Fox0x01

Azeria Labs
@azeria_labs

THANK YOU. ☺