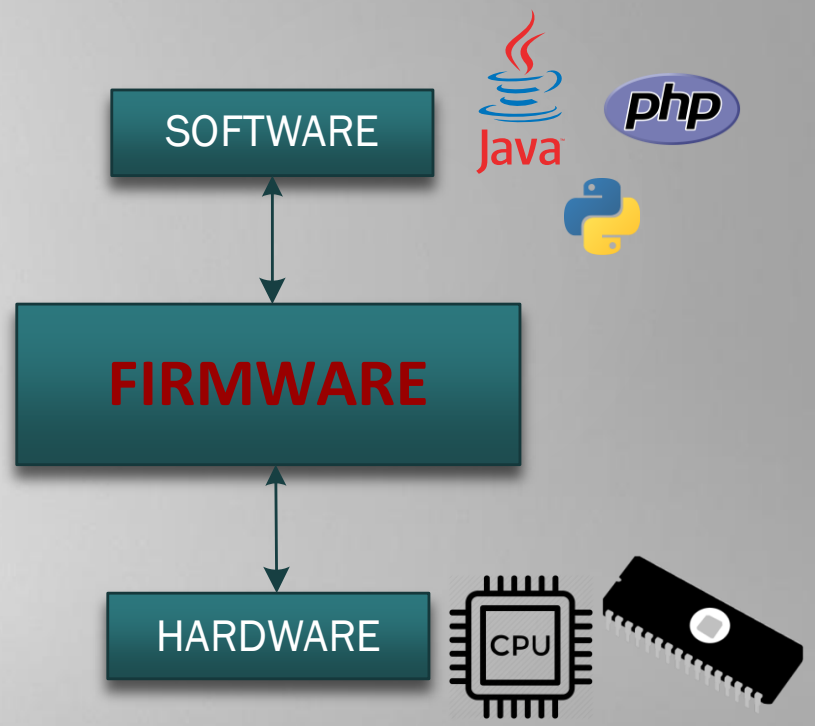
Analisi Dinamica del Firmware in dispositivi IoT

GIAN MARCO DE COLA

KNOW WHAT

COS'È IL FIRMWARE

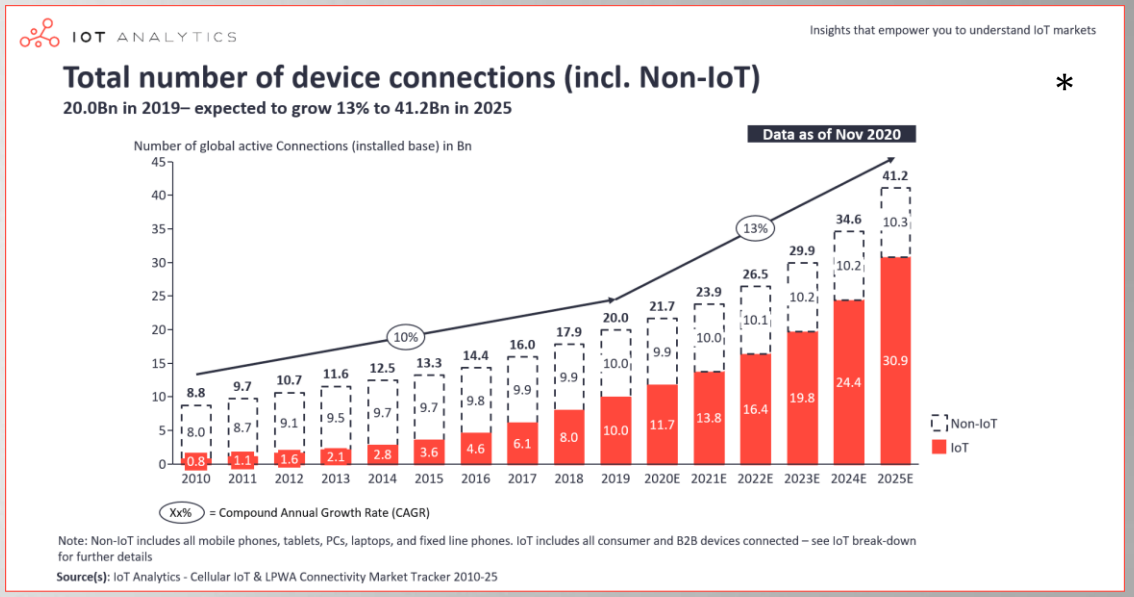
- Programma (o insieme di programmi) **integrato permanentemente** su un dispositivo hardware
- Può essere un **sistema operativo**, un **filesystem** o un singolo programma che dona **funzionalità** ad un dispositivo *embedded*
- Tra la grande varietà di dispositivi che contengono firmware ci si è soffermati su **dispostivi IoT**



KNOW WHY

PERCHÈ L'ANALISI DEL FIRMWARE IoT È IMPORTANTE

- Interesse costantemente **in crescita** verso questo tipo di dispositivi
- Utilizzo di processori ad **architetture RISC** per diminuire costi e dimensioni
- Firmware **altamente riusato**, **poco sanitizzato**
- Sono dispositivi **presenti nelle nostre** case ed **esposti ad Internet**



*: https://iot-analytics.com/state-of-the-iot-2020-12-billion-iot-connections-surpassing-non-iot-for-the-first-time/

KNOW HOW

COME SI ANALIZZA IL FIRMWARE

- Due **approcci opposti** per l'analisi del firmware (è possibile utilizzare un ibrido)

ANALISI STATICA

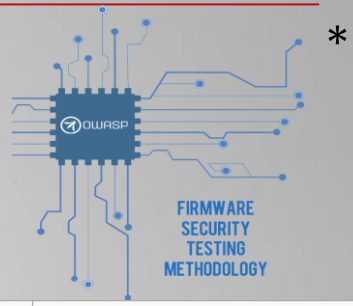
- Ispezione del codice sorgente/codice disassemblato a sistema spento
- Altamente scalabile e automatizzabile 😁
- Grande quantità di falsi positivi 😢
- Difficile analizzare le vulnerabilità di ogni binario/sorgente 😢

ANALISI DINAMICA

- Test a *runtime* del firmware
- Test del comportamento del dispositivo *running* 😁
- Pochi falsi positivi 😁
- Difficilmente automatizzabile e scalabile 😢
- Toolkit difficili da configurare 😢

Metodologia OWASP per l'IoT

- Nel 2014 L'OWASP introduce una **metodologia comune** per tutti i security tester IoT, aggiornata ogni 4 anni
- Si definiscono **9 fasi** in cui suddividere l'analisi del firmware IoT
- Ci si è soffermati sulla studio della fase di **emulazione** del firmware e sulla sua **analisi dinamica** (fasi **6-9**)
- E' possibile un'**automazione** della fase di emulazione?



*

Stage	Description
1. Information gathering and reconnaissance	Acquire all relative technical and documentation details pertaining to the target device's firmware
2. Obtaining firmware	Attain firmware using one or more of the proposed methods listed
3. Analyzing firmware	Examine the target firmware's characteristics
4. Extracting the filesystem	Carve filesystem contents from the target firmware
5. Analyzing filesystem contents	Statically analyze extracted filesystem configuration files and binaries for vulnerabilities
6. Emulating firmware	Emulate firmware files and components
7. Dynamic analysis	Perform dynamic security testing against firmware and application interfaces
8. Runtime analysis	Analyze compiled binaries during device runtime
9. Binary Exploitation	Exploit identified vulnerabilities discovered in previous stages to attain root and/or code execution

*: https://owasp.org/www-project-internet-of-things/#div-validate_and_test

QEMU

- **Emulatore** software e **virtualizzatore** di sistemi hardware sviluppato in C
- Obbiettivo primario: esecuzione di un sistema operativo target all'interno di un altro sistema operativo host
- E' in grado di emulare un grande numero di **architetture CPU**, **schede madri**, **interfacce** di rete, di I\O e di memorizzazione
- Utilizza la ***dynamic translation*** per tradurre istruzioni tra host e target

QEMU: user mode

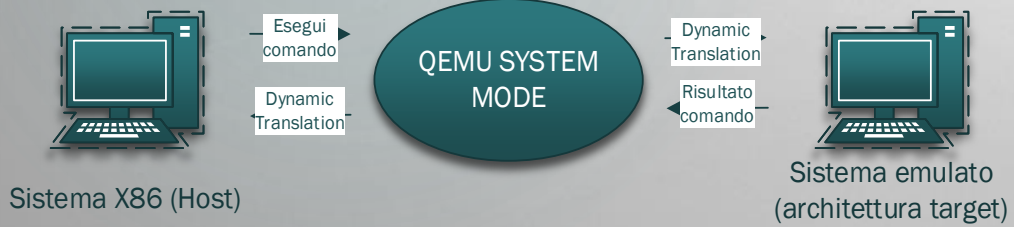
- Permette di eseguire comandi su un architettura di **CPU diversa** da quella dell'host, **senza** dover creare un'intera **macchina virtuale**

- Molto utile se vogliamo emulare **un solo file eseguibile**

  - `qemu-<arch> -L <prefix> -g <port> /path/to/executable`

QEMU: system mode

- Crea una **macchina virtuale** specificando la configurazione che si preferisce
- **Altamente configurabile** tramite parametri da linea di comando
- Diverse metodologie di accesso e comunicazione con la macchina virtuale



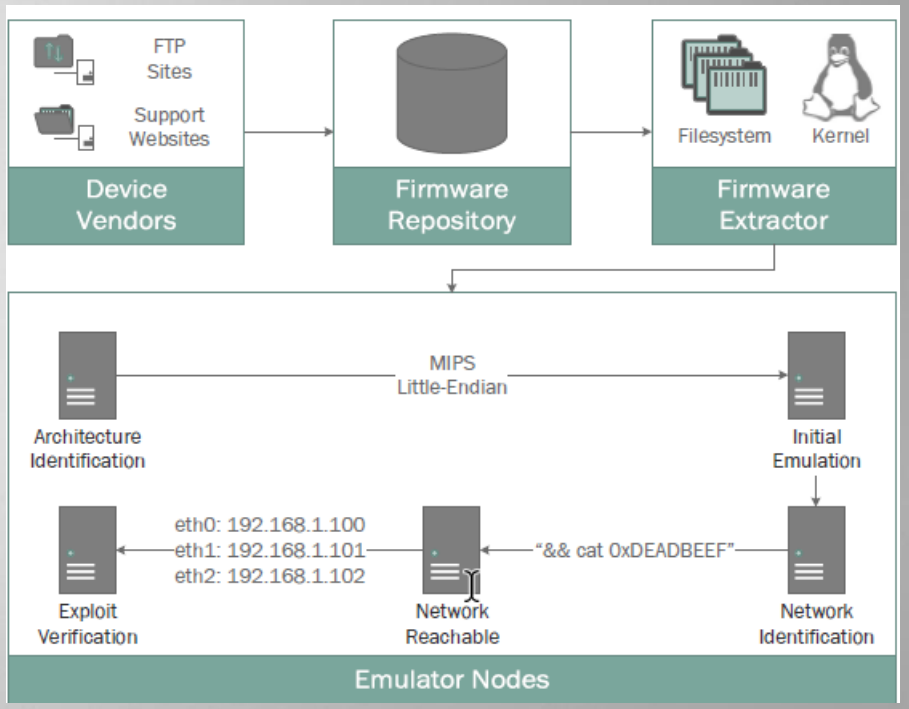

```
if [[ $# != 2 ]]; then
        echo -e "Usage: $0 arch kernel\narch: mips, mipsel, arm\nkernel: old(vmlinux-2.6.32-5), new(vmilux-3.2.0-4)"
        exit -1
fi
if [[ "$1" != "mips" && "$1" != "arm" && "$1" != "mipsel" ]]; then
        echo "Please enter a supported architecture: arch $1 not supported!"
        echo -e "Usage: $0 arch kernel\narch: mips, mipsel, arm\nkernel: old(vmlinux-2.6.32-5), new(vmilux-3.2.0-4)"
        exit -2
fi
ARCH="$1"
case "$2" in
        "old")
                if [[ $ARCH == "mips" || $ARCH == "mipsel" ]]; then
                        KERNEL=$ARCH"/kernel/vmlinux-2.6.32-5-4kc-malta"
                        BOARD="malta"
                        INITRD=""
                        NETDEV="e1000"
                        CONSOLE="ttyS0"
                else
                        KERNEL=$ARCH"/kernel/vmlinuz-2.6.32-5-versatile"
                        KERNEL_VERS="2.6.32-5-versatile"
                        BOARD="versatilepb"
                        INITRD="-initrd arm/kernel/initrd.img-"$KERNEL_VERS
                        NETDEV="virtio-net-pci"
                        CONSOLE="ttyAMA0"
                fi
                IMAGE=$ARCH"/debian_squeeze_"$ARCH"_standard.qcow2"
                ;;
        "new")
                if [[ $ARCH == "mips" || $ARCH == "mipsel" ]]; then
                        KERNEL=$ARCH"/kernel/vmlinux-3.2.0-4-4kc-malta"
                        INITRD=""
                        BOARD="malta"
                        CONSOLE="ttyS0"
                        NETDEV="e1000"
                else
                        BOARD="versatilepb"
                        KERNEL=$ARCH"/kernel/vmlinuz-3.2.0-4-versatile"
                        KERNEL_VERS="3.2.0-4-versatile"
                        NETDEV="virtio-net-pci"
                        CONSOLE="ttyAMA0"
                        INITRD="-initrd arm/kernel/initrd.img-"$KERNEL_VERS
                fi
                IMAGE=$ARCH"/debian_wheezy_"$ARCH"_standard.qcow2"
                ;;
        *)
                echo "Please enter a supported kernel age: kernel $3 not supported!"
                echo -e "Usage: $0 arch kernel\narch: mips, mipsel, arm\nkernel: old(vmlinux-2.6.32-5), new(vmilux-3.2.0-4)"
                exit -3
                ;;
esac
echo "---------------------------------Starting QEMU emulation---------------------------------------"
echo -e "board: $BOARD\narch:$ARCH\nkernel file: $KERNEL\n image file:$IMAGE\ninitrd file: $INITRD\n network device: $NETDEV"
qemu-system-$ARCH -netdev tap,id=t1,ifname=tap1,script=/etc/qemu-ifup,downscript=/etc/qemu-ifdown -device $NETDEV,netdev=t1\
        -M $BOARD -m 256 -nographic -kernel $KERNEL -append "root=/dev/sda1 console=$CONSOLE" $INITRD -hda $IMAGE
echo "-----------------------Cleaning up and stopping emulation---------------------------"
rm /tmp/qemu*
```

firmadyne: verso l'automazione

- firmadyne è un toolkit rivolto **all'automazione dell'analisi dinamica**
- Si è principalmente analizzato le sue features di automazione dell'**emulazione** del firmware *

- Il workflow di emulazione di firmadyne segue **2 fasi**

  1. **Fase di apprendimento**: si raccolgono informazioni sulla configurazione di rete del dispositivo
  2. **Fase di emulazione:** si utilizzano le informazioni della prima fase per la configurazione

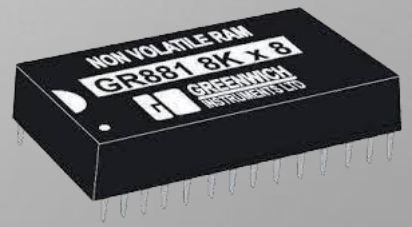


Schema logico di funzionamento di firmadyne, diviso per moduli

*: D. D. Chen, M. Egele, M. Woo and D. Brumley, "Towards Automated Dynamic Analysis for Linux-based Embedded Firmware," in *Proceedings of the 22nd NDSS Symposium*, 2016.

firmadyne: libnvram

- La **NVRAM** è un particolare tipo di memoria presente spesso in **dispositivi *embedded***
- firmadyne offre un modulo per **emulare** questo componente cruciale
- Libnvram **intercetta** le system calls del kernel all'NVRAM e vi risponde con **coppie chiave-valore** costruite *ad-hoc*
- E' possibile **aggiungere** coppie chiave-valore definite dall'utente
  - Direttamente nel **codice** (dovendo ricompilare il modulo) 😢
  - Nella cartella **libnvram.override** da creare sul filesystem del firmware 😄



firmadyne: qualche esempio


```
// Default paths for NVRAM default values.
#define NVRAM_DEFAULTS_PATH \
    /* "DIR-505L_FIRMWARE_1.01.ZIP" (10497) */ \
    PATH("/var/etc/nvram.default") \
    /* "DIR-615_REVE_FIRMWARE_5.11.ZIP" (9753) */ \
    PATH("/etc/nvram.default") \
    /* "DGL-5500_REVA_FIRMWARE_1.12B05.ZIP" (9469) */ \
    TABLE(router_defaults) \
    PATH("/etc/nvram.conf") \
    PATH("/etc/nvram.deft") \
    PATH("/etc/nvram.update") \
    TABLE(Nvrams) \
    PATH("/etc/wlan/nvram_params") \
    PATH("/etc/system_nvram_defaults")

// Default values for NVRAM.
#define NVRAM_DEFAULTS \
    /* Linux kernel log level, used by "WRT54G3G_2.11.05_ETSI_code.bin"
    ENTRY("console_loglevel", nvram_set, "7") \
    /* Reset NVRAM to default at bootup, used by "WNR3500v2-V1.0.2.10_23
    ENTRY("restore_defaults", nvram_set, "1") \
    ENTRY("sku_name", nvram_set, "") \
    ENTRY("wla_wlanstate", nvram_set, "") \
    ENTRY("lan_if", nvram_set, "br0") \
    ENTRY("lan_ipaddr", nvram_set, "192.168.0.50") \
    ENTRY("lan_bipaddr", nvram_set, "192.168.0.255") \
    ENTRY("lan_netmask", nvram_set, "255.255.255.0") \
    /* Set default timezone, required by multiple images */ \
    ENTRY("time_zone", nvram_set, "EST5EDT") \
    /* Set default WAN MAC address, used by "NBG-416N_V1.00(USA.7)C0.zip" (12786) */ \
    ENTRY("wan_hwaddr_def", nvram_set, "01:23:45:67:89:ab") \
    /* Attempt to define LAN/WAN interfaces */ \
    ENTRY("wan_ifname", nvram_set, "eth0") \
    ENTRY("lan_ifnames", nvram_set, "eth1 eth2 eth3 eth4") \
    /* Used by "TEW-638v2%201.1.5.zip" (12898) to prevent crash in 'goahead' */ \
    ENTRY("ethConver", nvram_set, "1") \
    /* Used by "Firmware_TEW-411BRPplus_2.07_EU.zip" (13649) to prevent crash in 'init' */ \
    ENTRY("lan_proto", nvram_set, "dhcp") \
    ENTRY("wan_ipaddr", nvram_set, "0.0.0.0") \
```

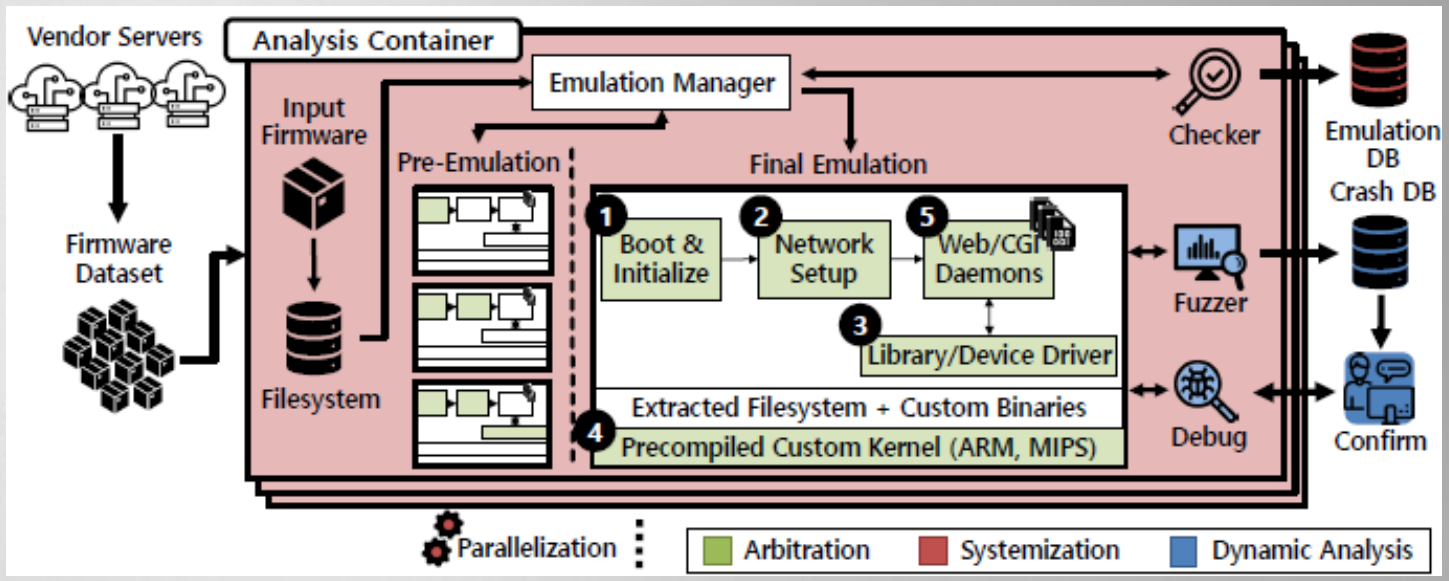
Sources/libnvram/config.h

```
# Get the network interfaces in the router, except 127.0.0.0/8
def findNonLoInterfaces(data, endianness):
    #lines = data.split("\r\n")
    lines = stripTimestamps(data)
    candidates = filter(lambda l: l.startswith("__inet_insert_ifa"), lines) # logs for the inconfig process
    if debug:
        print("Candidate ifaces: %r" % candidates)
    result = []
    if endianness == "eb":
        fmt = ">I"
    elif endianness == "el":
        fmt = "<I"
    for c in candidates:
        g = re.match(r"^__inet_insert_ifa\[[^\]]+\]: device:([^ ]+) ifa:0x([0-9a-f]+)", c)
        if g:
            (iface, addr) = g.groups()
            addr = socket.inet_ntoa(struct.pack(fmt, int(addr, 16)))
            if (not addr.startswith("127.")) and addr != "0.0.0.0":
                result.append((iface, addr))
    return result
```

Scripts/makeNetwork.py

FirmAE: l'emulazione mediata

- FirmAE è un toolkit rivolto all'**automazione dell'analisi dinamica** fortemente basato su firmadyne
- Mantiene il workflow di **emulazione in due fasi** ma aggiunge il concetto di **emulazione mediata**
- Diverse modalità di esecuzione (**prova**, **debug**, **esecuzione**, **analisi**)



Schema logico di funzionamento di FirmAE, diviso per moduli

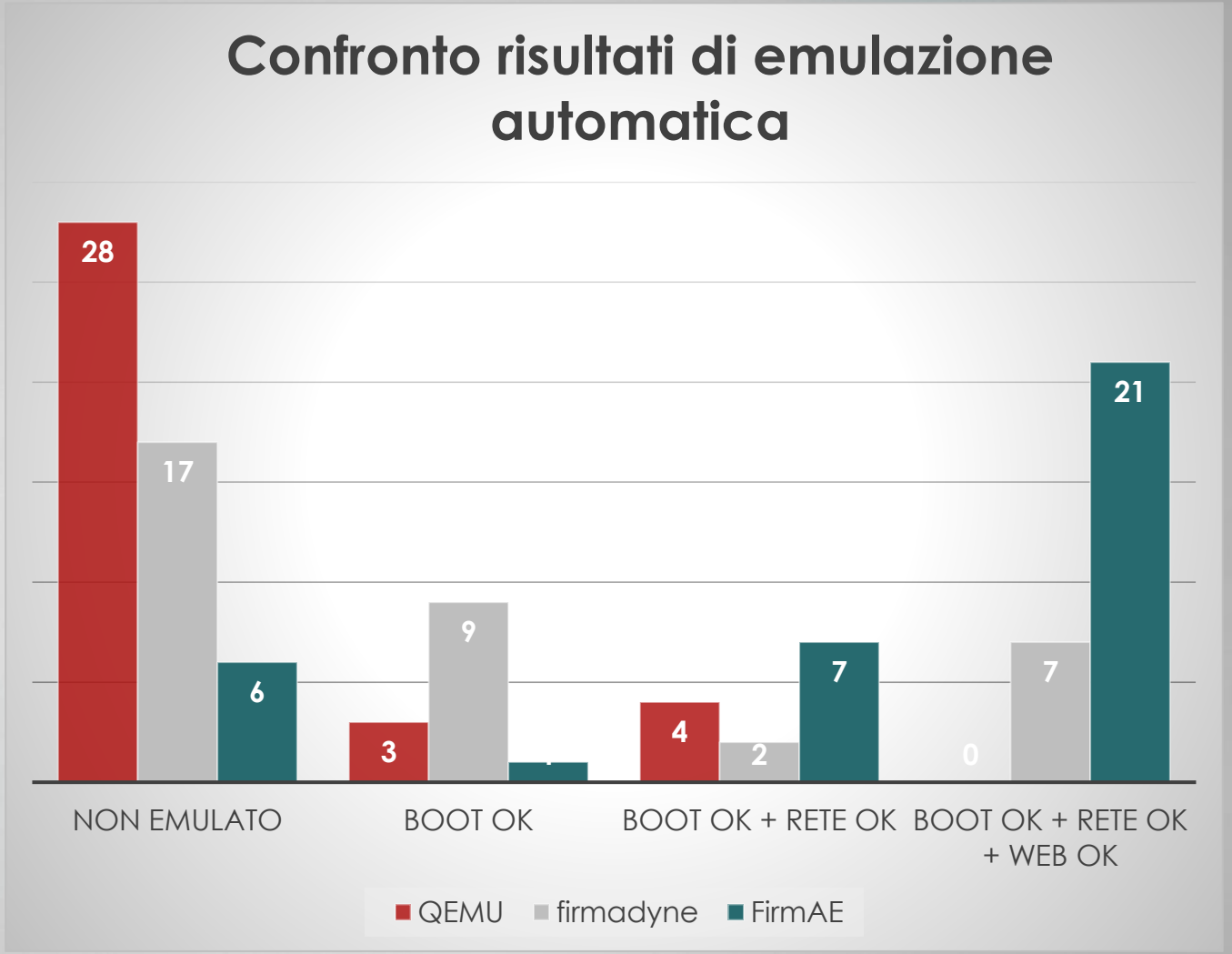
*: M. Kim, D. Kim, E. Kim, S. Kim, Y. Jang e Y. Kim, «FirmAE: Towards Large-Scale Emulation of IoT Firmware for Dynamic Analysis,» in *Annual Computer Security Applications Conference*, 2020

FirmAE: le mediazioni

- Interventi volti a **massimizzare le probabilità** di emulare correttamente un firmware
  - introdotte in seguito all'**analisi di fallimenti** di firmadyne in 437 firmware
- **Modifiche automatiche** al firmware in fase di emulazione finale, dopo aver dedotto i punti dove agire in modalità di prova
- *"We do not aim to resolve all the discrepancies in emulated environment. Instead, we aim at a concise emulation for dynamic testing, and our emulation goal can be illustrated with the following properties: 1) booting without any kernel panic, 2) network reachability from the host, and 3) web service availability for dynamic analysis. […] we support this hypothesis by successfully running emulated web services in 892 firmware images from 1,124 images, and we found 12 0-day vulnerabilities by conducting dynamic security analysis." **

*: M. Kim, D. Kim, E. Kim, S. Kim, Y. Jang e Y. Kim, «FirmAE: Towards Large-Scale Emulation of IoT Firmware for Dynamic Analysis,» in *Annual Computer Security Applications Conference*, 2020

I tool a confronto



- Emulazione senza tuning ne troubleshooting da parte dell'utente
  - Dati relativi ad un dataset di 35 firmware

DEMO TIME!