ISACA®

Sistemi informativi: averne fiducia e trarne valore

Rome Chapter

Hack the Firmware

L'ecosistema digitale è pervaso da dispositivi IoT/embedded ai quali colleghiamo i nostri sistemi di lavoro e personali e sui quali transitano dati sensibili, tutti questi dispositivi hanno un firmware.
La sicurezza del firmware non è più un optional ed è un obiettivo che può essere raggiunto.
A tal fine si può allestire un "piccolo laboratorio tecnico", dove analizzare il firmware per evidenziare falle di sicurezza usando la metodologia OWASP abbinata a software open source e "community free professionali" capaci di evidenziare anche zero days!

Massimiliano Graziani di Cybera

con i video straordinari di Giuseppe Compare di Mindstorm

Serra Sant'Abbondio (PU) 11/10/2024

ISACA
Sistemi informativi: averne fiducia e trarne valore
Rome Chapter

Massimiliano Graziani

Massimiliano Graziani, CEO e fondatore di Cybera Srl, DES di Innvovery, Consulente in Computer Forensics presso la Procura della Repubblica, le parti Private e le FF.OO., è dotato delle seguenti certificazioni internazionali:

- CISM (Certified Information Security Manager rilasciata da ISACA);
- CHFI (Computer Hacking Forensic Investigator rilasciata da EC-Council);
- CEH (Certified Ethical Hacker rilasciata da EC-Council);
- CIFI (Certified Information Forensics Investigator rilasciata da IISFA);
- CFE (Certified Fraud Examiner rilasciata da ACFE);
- ACE (AccessData Certified Examiner rilasciata da AccessData);
- OPSA (OSSTMM Professional Security Analyst rilasciata da ISECOM);
- CIFIP (Certified Forensic Investigation Professional rilasciata da IICFIP);
- TCNU (Tenable Certified Nessus User rilasciata da Tenable);
- CDFP (Certified Digital Forensics Professional rilasciata da IICFIP);
- CSTMIC (COUNTERSURVEILLANCE FORENSIC CYBER Certification);
- Bsi Lead Auditor BS7799-2:2002 (ISO 27001:2022) rilasciata da BSI
- e altre…

E' stato docente presso:

- Centro Interforze di Formazione Intelligence/GE (CIFI/GE) del Ministero della Difesa (Corso Operativo di Computer Forensics)
- Scuola di Polizia Tributaria della Guardia di Finanza (Corso Operativo di Computer Forensics)
- Università La Sapienza Roma (Corso Informatica Giuridica)
- Università Link Campus (Corso Digital Forensics di diversi Master)
- Università LUMSA Roma (Corso Diritto Penale dell'Informatica)
- Università di Salerno (Convegno La Tecnologia al servizio dell'Indagine Scientifica)
- Università del Molise (Master Information Security Management – Modulo Computer Forensics)

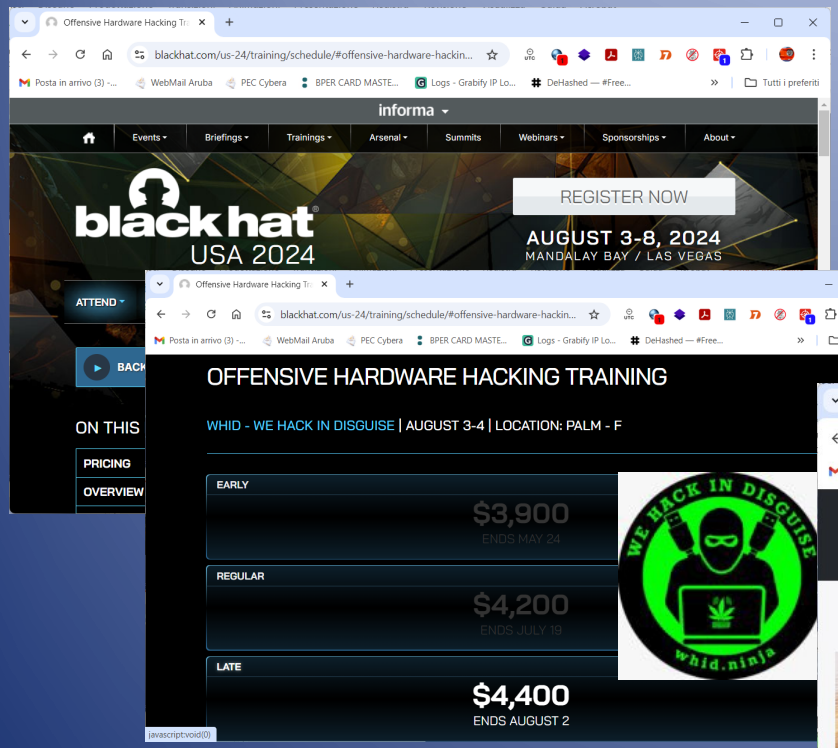
Docente volontario presso IISFA, OLAF, FF.OO., Ufficio Informatico presso Consiglio Superiore Magistratura Milano, Ministero Economia e Finanze UCAMP (anche in ambito frodi telematiche e furto di identità digitale) e ISACA Roma. E' uno dei soci fondatori del capitolo italiano dell'IISFA (International Information Systems Forensics Association www.iisfa.it). Ha conseguito il Corso in Information Security Management del Politecnico di Milano (Center of Excellence For Research, Innovation, Education and industrial Labs), dove ha ideato e sviluppato il project work "E-Forensics Best Practices" e conseguito la qualifica internazionale IQNet Information Security Manager. E' ideatore e uno degli autori del Memberbook IISFA 2010 nel quale ha pubblicato la prima metodologia per la computer forensics aziendale in Italia. Socio fondatore di ONIF (Osservatorio Nazionale Informatica Forense www.onif.it).

E' stato Membro del Direttivo come Responsabile della Formazione ACFE Italia. Ha inoltre conseguito altre certificazioni internazionali come Cisco CCNA, ed altre. Fondatore insieme ad altri del Capitolo italiano OWASP (2005) e membro OWASP International a vita. www.cobrasoft.it

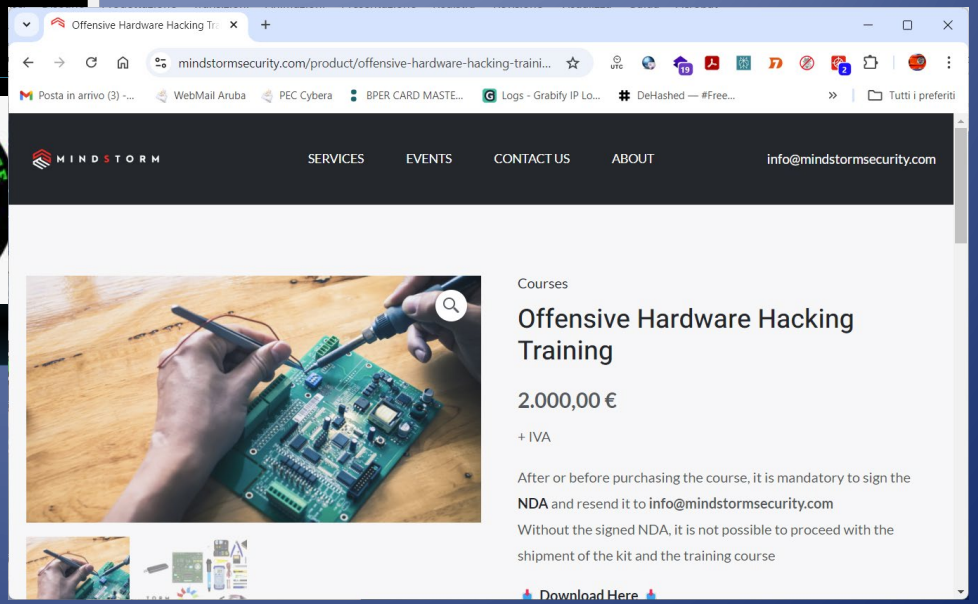
ISACA
Sistemi informativi: averne fiducia e trarne valore
Rome Chapter

Hack the Firmware

tecniche e strumenti di analisi firmware



I migliori specialisti della tematica sono italiani…



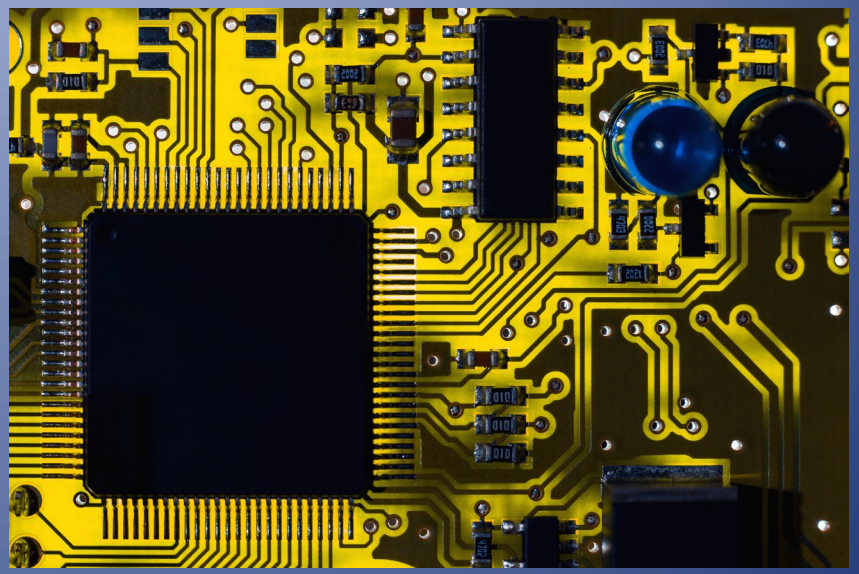
Grazie a
Giuseppe Compare di Mindstorm
e
Luca Bongiorni Whe hack In Disguise

Cos'è il firmware?

Il firmware è un tipo di software che assegna istruzioni operative ai componenti hardware di un device, permettendo il funzionamento di base. Il firmware viene installato dal produttore e in genere non può essere modificato o rimosso.

Come funziona il firmware?

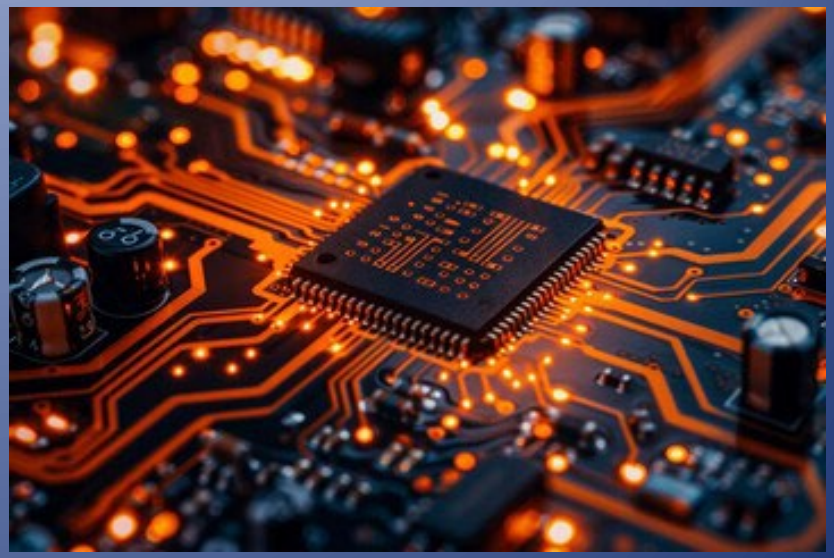
Il firmware ricoprire il ruolo da intermediario tra il sistema operativo e i componenti hardware, ovvero, fornisce al dispositivo le istruzioni operative di base.
Il termine "firmware" deriva proprio da questo ruolo di traduttore tra l'hardware e le altre forme di software presenti sul dispositivo.



Quando si accende un dispositivo digitale, il firmware si attiva e avvia l'esecuzione dei processi necessari per mettere in funzione il dispositivo, come l'attivazione dei componenti hardware che consentono di avviare il boot di un eventuale sistema operativo.

Premessa

Il firmware conserva le informazioni in modo permanente, anche quando il dispositivo viene spento, viene generalmente archiviato nella memoria flash di sola lettura (ROM) del sistema e viene eseguito separatamente dagli altri tipi di software installati sulla RAM riscrivibile. Ma il funzionamento esatto del firmware e la posizione di archiviazione dipendono dal tipo e dallo scopo specifico.



Il firmware di sistema è un software utilizzato per le operazioni di elaborazione fondamentali, quali l'avvio, il caricamento dei sistemi operativi, il riconoscimento dei componenti hardware e la comunicazione tra questi ultimi. Ma il firmware può essere ulteriormente classificato in base a quanto in basso si trova nello stack di un dispositivo.

Secondo il livello di sofisticazione e complessità del firmware, di seguito sono elencati i diversi tipi che è possibile trovare in un dispositivo :

Firmware di basso livello

Il firmware di basso livello è parte integrante dell'hardware di un dispositivo e risiede nei chip della memoria di sola lettura (ROM) o della memoria di sola lettura programmabile (PROM) che non possono essere aggiornati o riscritti. I dispositivi che utilizzano esclusivamente questo tipo di firmware sono in genere dispositivi dedicati (si pensi alle sveglie digitali o ai telecomandi per TV) in cui il firmware funge sostanzialmente da sistema operativo.

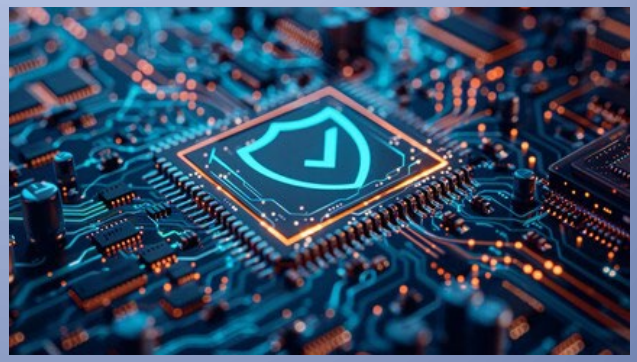
Firmware di alto livello

Il firmware di alto livello viene installo nei chip della memoria flash "non volatile" riscrivibile anziché nella ROM, e di conseguenza può essere modificato e aggiornato.
Presente su dispositivi come PC, Mac e console di gioco, è chiamato firmware "di alto livello" perché si trova al di sopra del firmware di basso livello nello stack software ed esegue istruzioni e funzioni più complesse.

Firmware di sottosistema

Il firmware di sottosistema, o firmware di dispositivo, è un tipo particolare di firmware di alto livello che viene eseguito in modo indipendente rispetto al firmware di sistema principale. Esempi di componenti hardware di sottosistema che hanno in genere un proprio firmware dedicato sono le CPU, le schede audio e i monitor.

Cos’è la protezione del firmware?



La protezione del firmware offre protezione contro gli hacker che sfruttano le vulnerabilità del codice del firmware per accedere senza autorizzazione a un dispositivo e installare malware o sottrarre dati. La sicurezza del firmware è sempre stata motivo di particolare preoccupazione per dispositivi come router e server.

Ma con la diffusione delle smart home, è sempre più importante garantire la sicurezza del firmware per fare in modo che i dispositivi IoT (Internet of Things) non siano un bersaglio facile per minacce come i rootkit, un tipo di malware particolarmente furtivo che si nasconde nel codice del firmware e che può essere rilevato solo dalle migliori tecnologie di scansione antivirus o pratiche di malware forensics.

La cosa più importante da fare per garantire la sicurezza del firmware di un dispositivo è assicurarsi che il firmware sia sempre aggiornato. Questo perché gli aggiornamenti del firmware includono patch progettate per risolvere le vulnerabilità esistenti, che potrebbero essere sfruttate per un attacco.

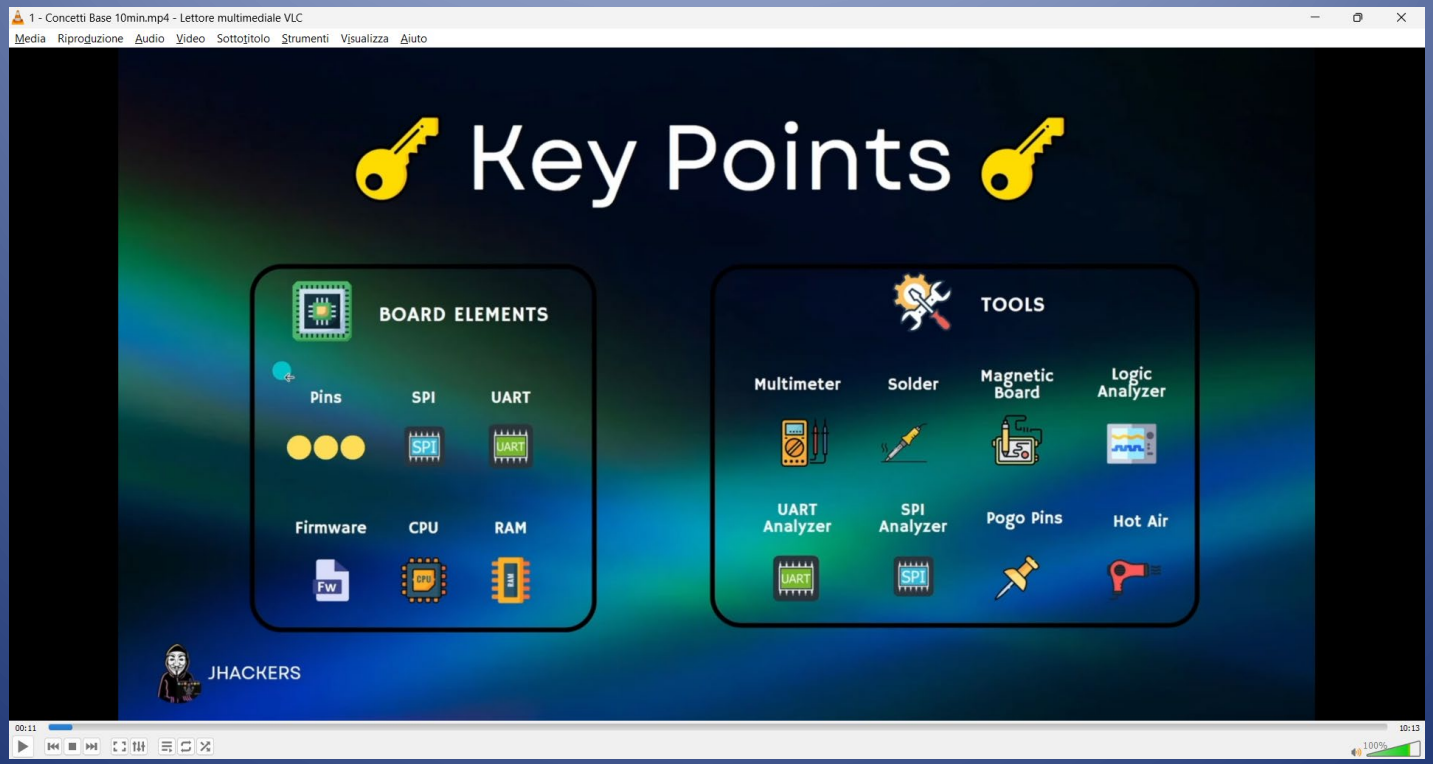
Premessa

Qual è la differenza tra firmware e software?
Tecnicamente, il firmware è un tipo di software, ma esiste una netta distinzione tra i due in termini di funzionalità. il firmware non è progettato per interagire con l'utente, ma per garantire il funzionamento di base del dispositivo e consentirgli di comunicare con i componenti dell'intero sistema.
Il software non dipende direttamente da un componente hardware specifico. In tal senso, si colloca "al di sopra" del firmware nella gerarchia del sistema. È il firmware, insieme ai componenti hardware integrati, a fare funzionare il software sotto forma di sistemi operativi, programmi e applicazioni, che sono i componenti
con cui l'utente interagisce.

I driver sono un tipo di firmware?
Benché si tratti di due tipi di software pensati per semplificare il funzionamento dei componenti hardware, driver e firmware hanno funzioni diverse. A differenza del firmware, che viene programmato direttamente sul dispositivo hardware per fornire istruzioni alla macchina, i driver consentono al sistema operativo di riconoscere e comunicare con un determinato tipo di dispositivo collegato, ad esempio un mouse o un monitor.

CONCETTI BASE

Video 1 - 10 min. di Giuseppe Compare

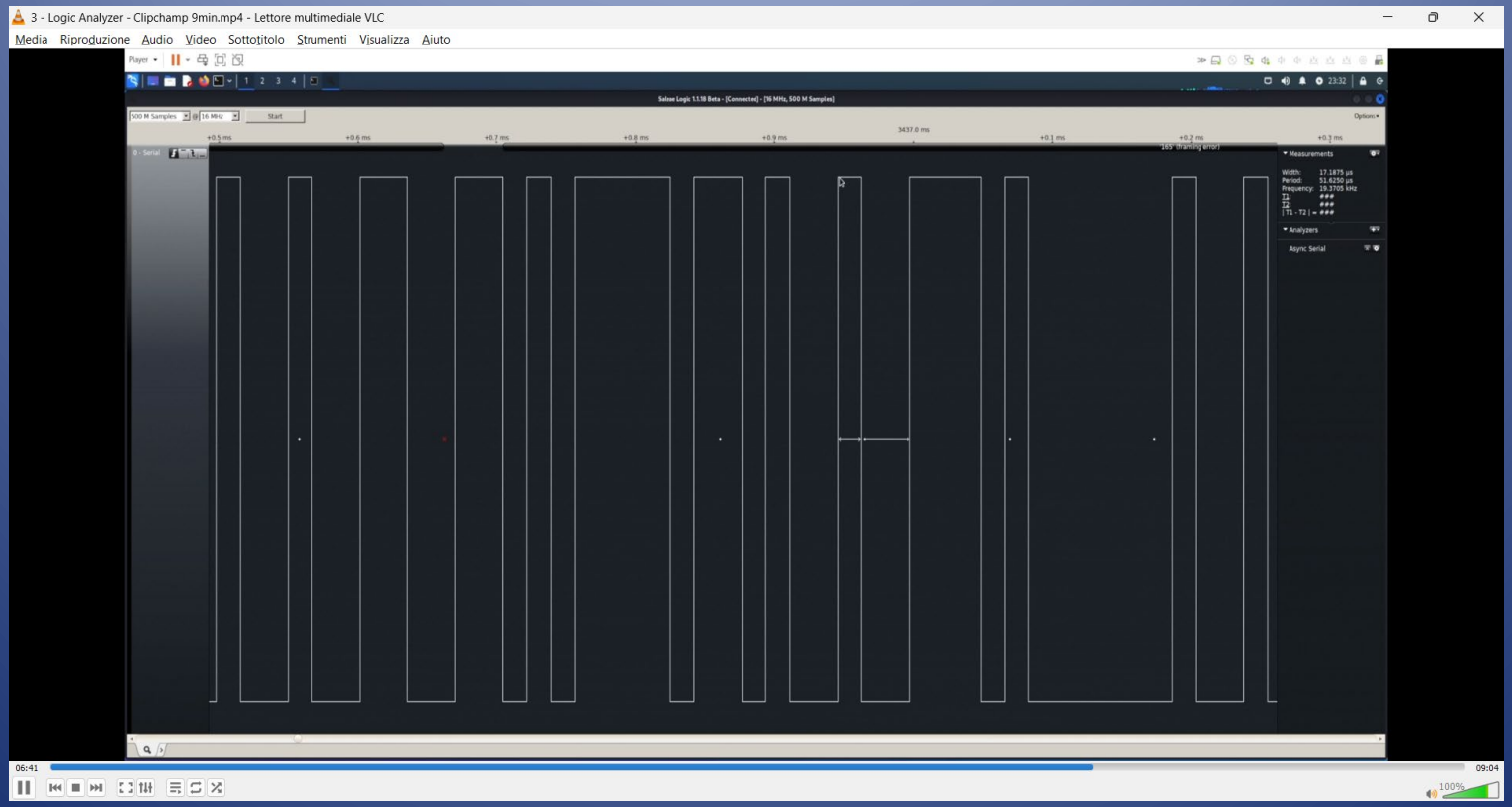


MULTIMETRO

Video 2 - 9 min. di Giuseppe Compare

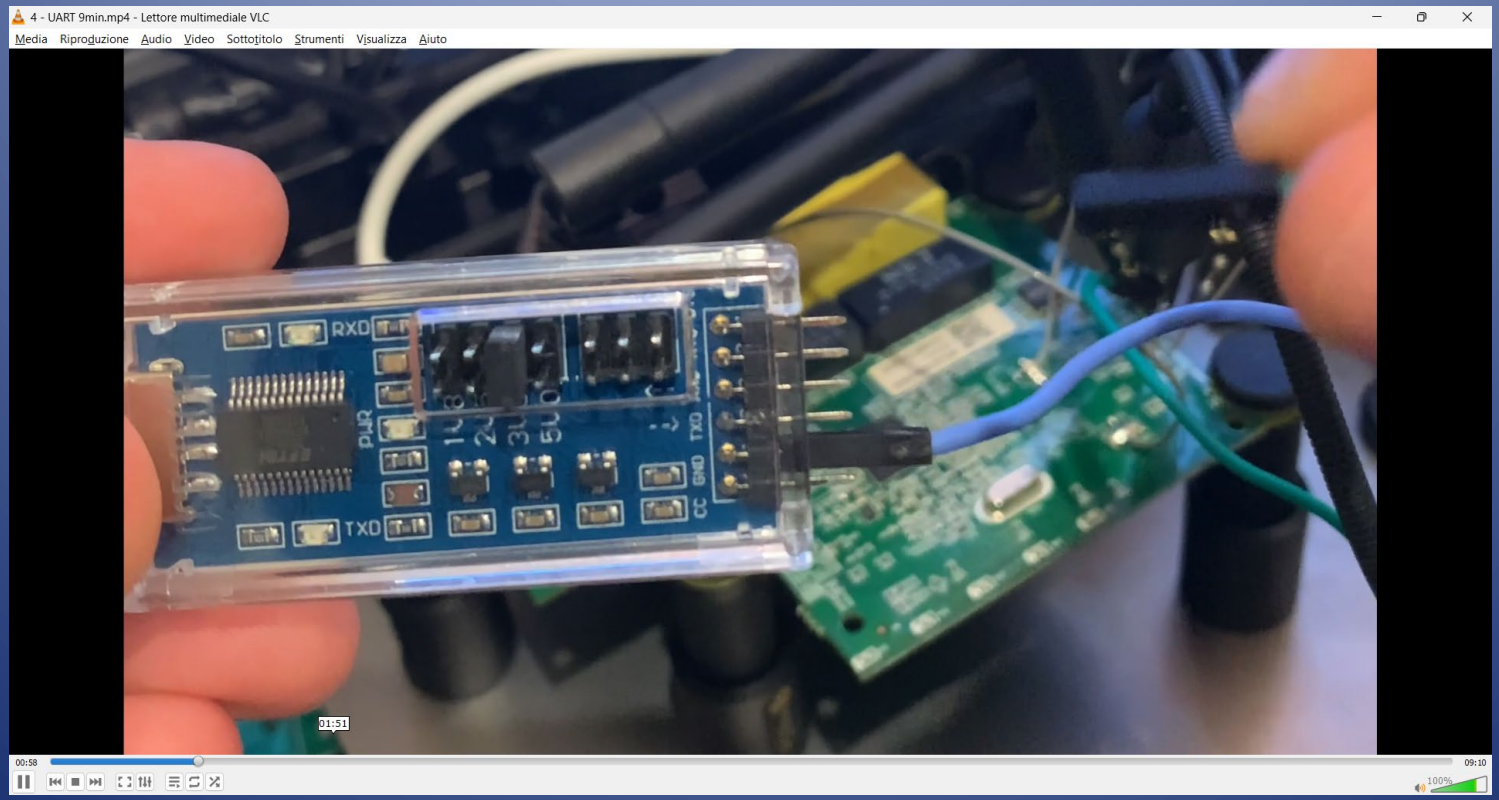
LOGIC ANALYZER

Video 3 - 9 min. di Giuseppe Compare



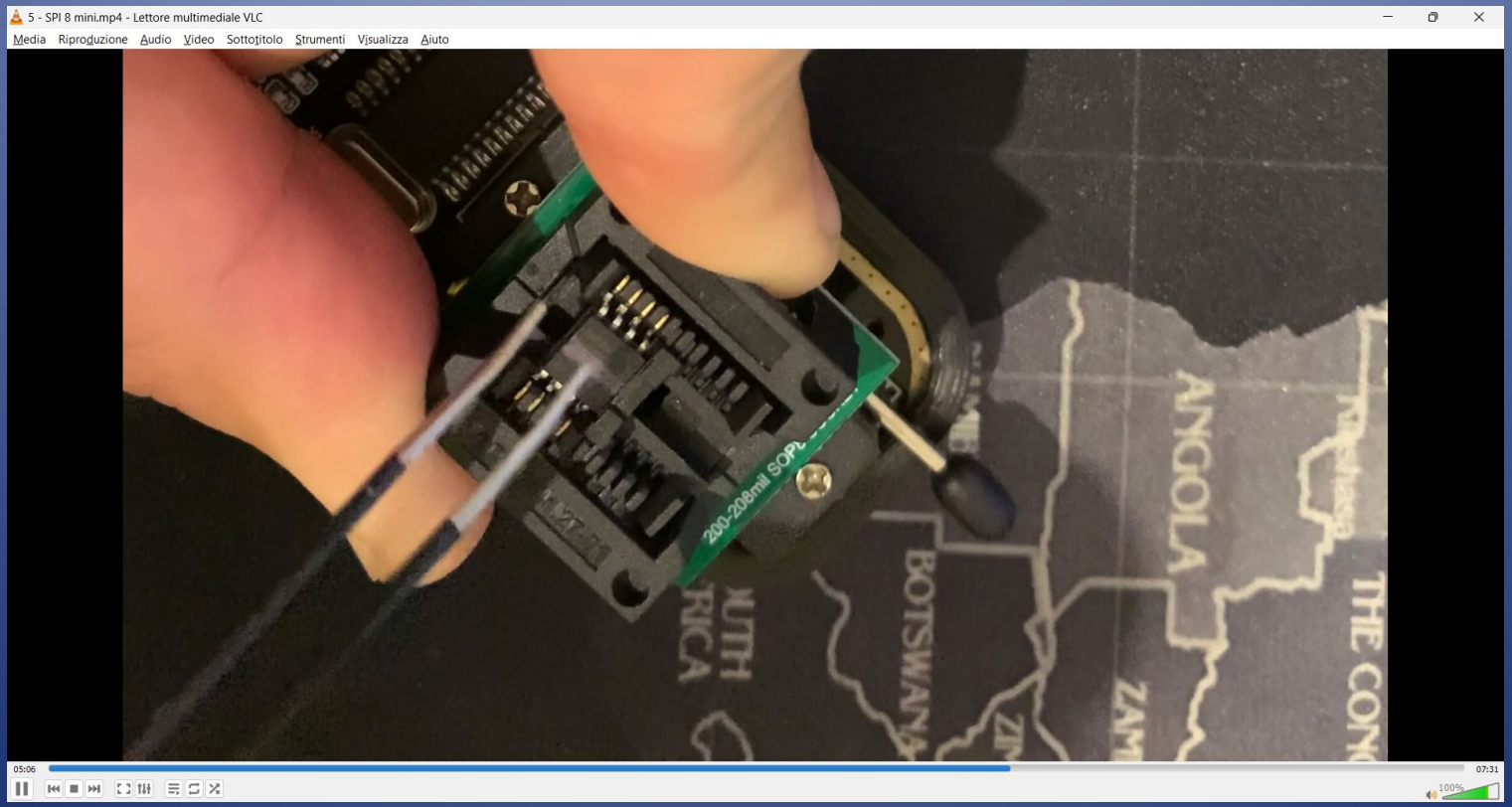
UART

Video 4 - 9 min. di Giuseppe Compare



SPI

Video 5 - 9 min. di Giuseppe Compare



Metodologie di VAPT firmware

Che sia connesso o meno alla rete, il firmware è il centro di controllo di qualsiasi dispositivo embedded. Pertanto, è fondamentale comprendere come il firmware può essere manipolato per eseguire funzioni non autorizzate e potenzialmente compromettere la sicurezza dell'ecosistema. Per iniziare a eseguire test di sicurezza e reverse engineering del firmware, utilizzare la seguente Firmware Security Testing Methodology (FSTM) come guida quando si deve intraprendere una valutazione di security.

La metodologia è composta da NOVE fasi studiate su misura per consentire a ricercatori di sicurezza, sviluppatori di software, consulenti, hobbisti e professionisti della sicurezza informatica di condurre valutazioni di sicurezza del firmware.

OWASP Firmware 9 FASI

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

Fase1: Ricognizione e Ricerca Informazioni

Durante questa fase, raccogli quante più informazioni possibili sul target per comprenderne la composizione complessiva e la tecnologia sottostante.
Prova a raccogliere quanto segue:

• **Architettura CPU supportata** • **Piattaforma del sistema operativo** • **Configurazioni del bootloader** • **Schemi hardware** • **Schede tecniche** • **Stime delle linee di codice (LoC)** • **Posizione del repository del codice sorgente** • **Componenti di terze parti** • **Licenze open source (ad esempio GPL)**	• **Changelog** • **ID FCC** • **Diagrammi di flusso di dati e progettazione** • **Modelli di minaccia** • **Precedenti report di penetration testing** • **Ticket di tracciamento dei bug (ad esempio Jira e piattaforme bug bounty come BugCrowd o HackerOne)**

Fase2: Ottieni il Firmware

Per iniziare a esaminare il contenuto del firmware, è necessario acquisire il file immagine del firmware. Tentare di ottenere il contenuto del firmware utilizzando uno o più dei seguenti metodi:

- Direttamente dal team di sviluppo, produttore/fornitore o cliente
- Creare da zero utilizzando le procedure dettagliate fornite dal produttore
- Dal sito di supporto del fornitore
- Query di Google Dork mirate a estensioni di file binari e piattaforme di condivisione di file come Dropbox, Box e Google Drive
    - Potresti trovare immagini di firmware tramite utenti che caricano contenuti su forum, blog o commentano siti in cui hanno contattato il produttore per risolvere un problema e hanno ricevuto il firmware tramite un file zip o un'unità flash inviata.
- Comunicazione tra dispositivi Man-in-the-middle (MITM) durante gli aggiornamenti
- Scarica build da posizioni di archiviazione del provider cloud esposte come bucket S3 di Amazon Web Services (AWS)
- Estrai direttamente dall'hardware tramite UART, JTAG, PICit, ecc.
- Sniffa la comunicazione seriale all'interno dei componenti hardware per le richieste del server di aggiornamento
- Tramite un endpoint hardcoded all'interno delle applicazioni mobili o thick
- Scarica il firmware dal bootloader (ad esempio U-boot) nell'archiviazione flash o sulla rete tramite tftp
- Rimuovi il chip flash (ad esempio SPI) o MCU dalla scheda per analisi offline ed estrazione dati (ULTIMA RISORSA).
    - Avrai bisogno di un programmatore di chip supportato per l'archiviazione flash e/o l'MCU.
- Ognuno dei metodi elencati varia in difficoltà e non deve essere considerato un elenco esaustivo. Seleziona il metodo appropriato in base agli obiettivi del progetto e alle regole di ingaggio. Se possibile, richiedi sia una build di debug che una build di release del firmware per massimizzare i casi d'uso della copertura di test nel caso in cui il codice di debug o la funzionalità vengano compilati all'interno di una release.

Fase3: Analisi del Firmware

Una volta ottenuta l'immagine del firmware, esplora gli aspetti del file per identificarne le caratteristiche. Utilizza i seguenti passaggi per analizzare i tipi di file del firmware, i potenziali metadati del root filesystem e ottenere una comprensione aggiuntiva della piattaforma per cui è compilato.
Sfrutta istruzioni binutils come:

```
file <bin>
strings
strings -n5 <bin>
binwalk <bin>
hexdump -C -n 512 <bin> > hexdump.out
hexdump -C <bin> | head (might find signatures in header)
```

Se nessuno dei metodi sopra indicati fornisce dati utili, è possibile quanto segue:

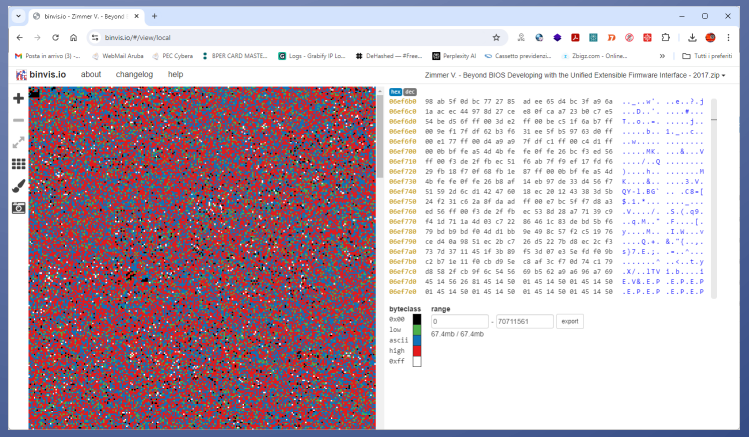
- Il binario potrebbe essere BareMetal
- Il binario potrebbe essere per una piattaforma di sistema operativo in tempo reale (RTOS) con un file system personalizzato
- Il binario potrebbe essere crittografato

Se il binario potrebbe essere crittografato, controlla l'entropia usando binwalk con il seguente comando:

```
$ binwalk -E <bin>
```

Bassa entropia = Non è probabile che sia crittografato
Alta entropia = È probabile che sia crittografato (o compresso in qualche modo).

Sono disponibili anche strumenti alternativi, ad esempio è disponibile BINVIS online oppure la versione da installare.



https://code.google.com/archive/p/binvis/downloads
https://binvis.io/#/

Fase4: Estrazione del file system

Questa fase prevede l'analisi del firmware e l'analisi dei dati del file system relativo per iniziare a identificare quanti più potenziali problemi di sicurezza possibili. Utilizzare i seguenti passaggi per estrarre i contenuti del firmware per la revisione del codice non compilato e delle configurazioni del dispositivo utilizzate nelle fasi successive. Di seguito sono illustrati sia i metodi di estrazione automatizzati che quelli manuali.

1. Utilizzare i seguenti strumenti e metodi per estrarre il contenuto del filesystem:

```
$ binwalk -ev <bin>
```

I file verranno estratti in "_binaryname/filesystemtype/etc/."
Tipi di filesystem supportati: squashfs, ubifs, romfs, rootfs, jffs2, yaffs2, cramfs, initramfs

Fase5: Analisi del contenuto del file system

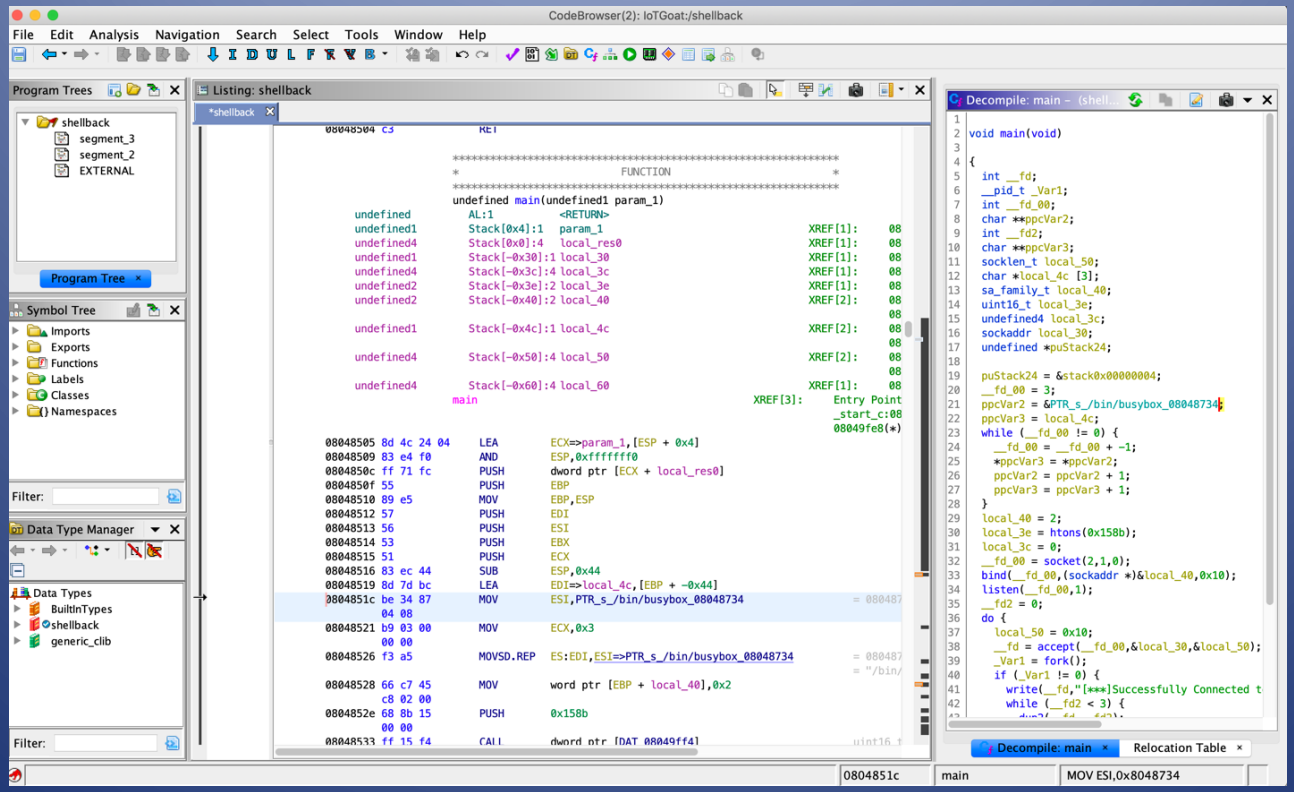
Durante questa fase, vengono raccolti indizi per le fasi di analisi dinamica e runtime. Indagare se il firmware di destinazione contiene quanto segue (non esaustivo):

- Demoni di rete non sicuri legacy come telnetd (a volte i produttori rinominano i binari per camuffarli)
- Credenziali hardcoded (nomi utente, password, chiavi API, chiavi SSH e varianti backdoor)
- Endpoint API hardcoded e dettagli del server backend
- Aggiornare la funzionalità del server che potrebbe essere utilizzata come punto di ingresso
- Rivedere il codice non compilato e avviare gli script per l'esecuzione di codice remoto
- Estrarre i binari compilati da utilizzare per l'analisi offline con un disassembler per le fasi future

Analizzare staticamente i contenuti del file system e il codice non compilato manualmente o sfruttando strumenti di automazione come firmwalker che analizzano quanto segue:

- /etc/shadow e /etc/passwd
- elenca directory /etc/ssl
- cerca elementi relative a SSL come .pem, .crt, ecc.
- cerca file di configurazione
- cerca script file
- cerca altri .bin file
- cerca parole chiave come admin, password, remote, AWS keys, ecc.
- ricerca di server web comuni utilizzati sui dispositivi IoT
- cerca binari comuni come ssh, tftp, dropbear, ecc.
- ricerca funzioni C vietate
- ricerca di funzioni vulnerabili all'iniezione di comandi comuni
- ricerca URL, indirizzi email e IP
- e molto altro…

Disassemblare i binari target sospetti con i dati raccolti da FACT usando IDA Pro, Ghidra, Hopper, Capstone o Binary Ninja. Analizza i binari per potenziali chiamate di sistema di esecuzione di codice remoto, stringhe, elenchi di funzioni, vulnerabilità di corruzione della memoria e identifica «Xref to system()» o chiamate di funzioni simili. Annota le potenziali vulnerabilità da usare per le fasi successive.
Qui sotto viene mostrato il binario “shellback” smontato utilizzando Ghidra



Fase6: Emulazione del firmware

Utilizzando dettagli e indizi identificati nelle fasi precedenti, il firmware e i suoi binari incapsulati devono essere emulati per verificare potenziali vulnerabilità.

Per eseguire l'emulazione del firmware, sono elencati di seguito alcuni approcci.

1. Emulazione parziale: emulazione di binari autonomi derivati dal file system estratto di un firmware, come /usr/bin/shellback

2. Emulazione completa: emulazione del firmware completo e delle configurazioni di avvio sfruttando la NVRAM falsa.

3. Emulazione tramite un dispositivo reale o una macchina virtuale: a volte, l'emulazione parziale o completa potrebbe non funzionare a causa di dipendenze hardware o architettura. Se l'architettura e l'endianness corrispondono a un dispositivo di proprietà, come un Raspberry Pie, il file system root o il binario specifico possono essere trasferiti al dispositivo per ulteriori test. Questo metodo si applica anche alle macchine virtuali pre-costruite che utilizzano la stessa architettura e endianness del target.

Fase7: Analisi Dinamica

In questa fase, esegui test dinamici mentre un dispositivo è in esecuzione nel suo ambiente normale o emulato. Gli obiettivi in questa fase possono variare a seconda del progetto e del livello di accesso concesso. In genere, ciò comporta la manomissione delle configurazioni del bootloader, test Web e API, fuzzing (servizi di rete e applicativi), nonché scansione attiva utilizzando vari set di strumenti per acquisire accesso elevato (root) e/o esecuzione del codice.
Gli strumenti che possono essere utili sono (non esaustivi):

- Burp Suite
- OWASP ZAP
- Commix
- Fuzzer come - American fuzzy loop (AFL)
- Fuzzer di rete come - Mutiny
- Nmap
- NCrack
- Metasploit

Fase8: Analisi Runtime

L'analisi runtime comporta l'associazione a un processo o binario in esecuzione mentre un dispositivo è in esecuzione nel suo ambiente normale o emulato. I passaggi di base dell'analisi runtime sono forniti di seguito:

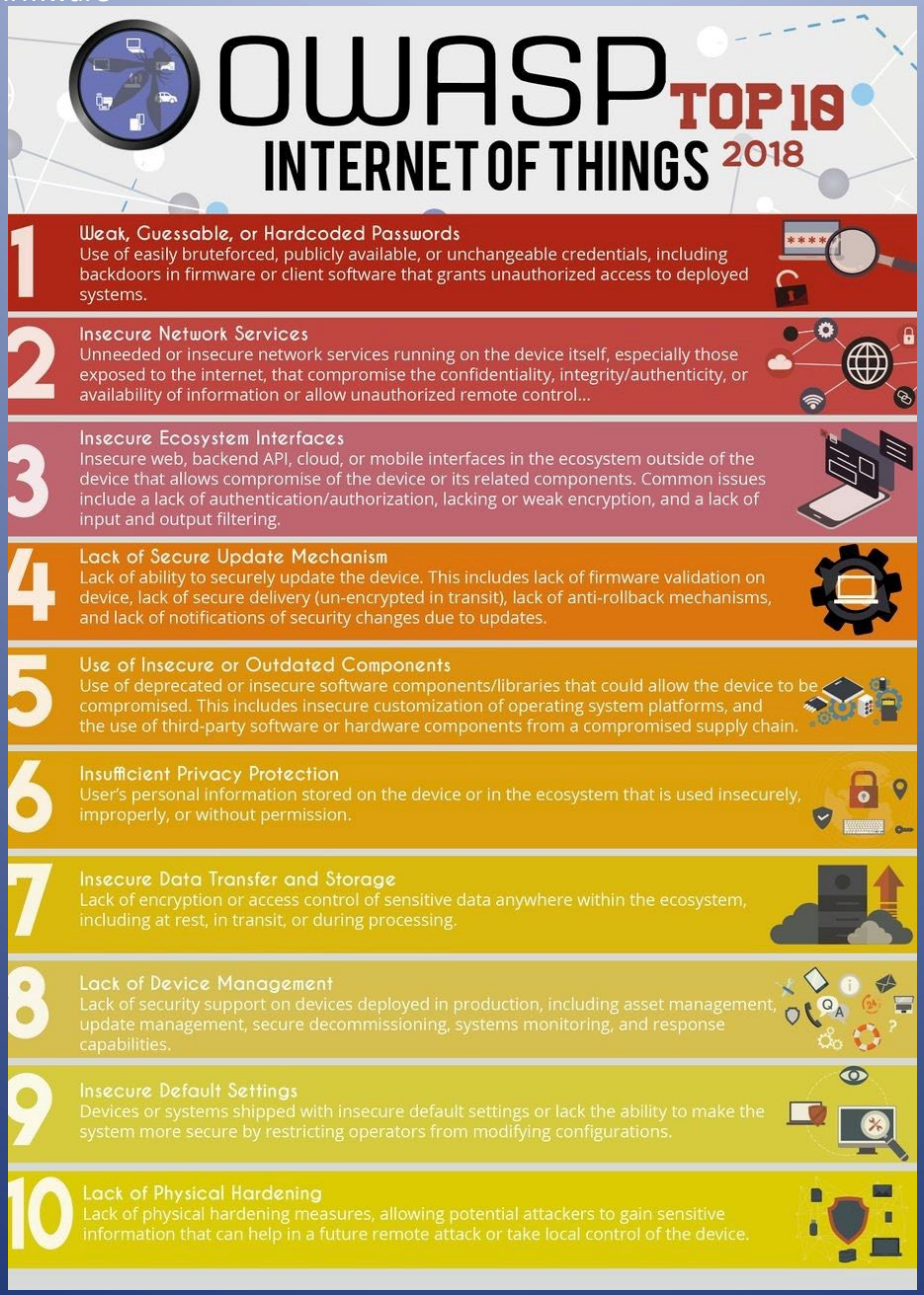
1. sudo chroot . ./qemu-arch -L <optionalLibPath> -g <gdb_port> <binario>
2. Allega gdb-multiarch o usa IDA per emulare il binario
3. Imposta i breakpoint per le funzioni identificate durante il passaggio 4 come memcpy, strncpy, strcmp, ecc.
4. Esegui stringhe di payload di grandi dimensioni per provocare un buffer overflow o crash di processo usando un fuzzer
5. Procedi alla fase di Binary Exploitation se viene identificata una vulnerabilità

Gli strumenti che possono essere utili sono (non esaustivi):

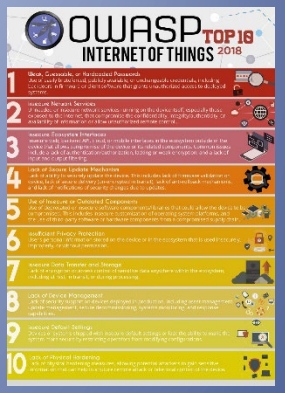
- gdb-multiarch
- Peda
- Frida
- ptrace
- strace
- IDA Pro
- Ghidra
- Binary Ninja
- Hopper

Fase9: Binary Exploitation

Dopo aver identificato una vulnerabilità in un binario dalle fasi precedenti, è richiesta una proof-of-concept (PoC) adeguata per dimostrare l'impatto e il rischio nel mondo reale.
Lo sviluppo del codice exploit richiede esperienza di programmazione in linguaggi di livello inferiore (ad esempio ASM, C/C++, shellcode, ecc.) e background nell'architettura target specifica (ad esempio MIPS, ARM, x86, ecc.). Il codice PoC comporta l'ottenimento di un'esecuzione arbitraria su un dispositivo o un'applicazione controllando un'istruzione in memoria. Non è comune che le protezioni runtime binarie (ad esempio NX, DEP, ASLR, ecc.) siano in atto nei sistemi embedded, tuttavia quando ciò accade, potrebbero essere necessarie tecniche aggiuntive come la programmazione orientata al ritorno (ROP).
ROP consente a un aggressore di implementare funzionalità dannose arbitrarie concatenando il codice esistente nel codice del processo/binario target noto come gadget. Saranno necessari dei passaggi per sfruttare una vulnerabilità identificata come un buffer overflow formando una catena ROP. Uno strumento che
può essere utile per situazioni come queste è il gadget finder di Capstone o ROPGadget
https://github.com/JonathanSalwan/ROPgadget
Utilizza i seguenti riferimenti per ulteriori indicazioni:
https://azeria-labs.com/writing-arm-shellcode/
https://www.corelan.be/index.php/category/security/exploit-writing-tutorials/


OWASP TOP 10
INTERNET OF THINGS 2018
1
Weak, Guessable, or Hardcoded Passwords
Use of easily bruteforced, publicly available, or unchangeable credentials, including backdoors in firmware or client software that grants unauthorized access to deployed systems.
2
Insecure Network Services
Unneeded or insecure network services running on the device itself, especially those exposed to the internet, that compromise the confidentiality, integrity/authenticity, or availability of information or allow unauthorized remote control...
3
Insecure Ecosystem Interfaces
Insecure web, backend API, cloud, or mobile interfaces in the ecosystem outside of the device that allows compromise of the device or its related components. Common issues include a lack of authentication/authorization, lacking or weak encryption, and a lack of input and output filtering.
4
Lack of Secure Update Mechanism
Lack of ability to securely update the device. This includes lack of firmware validation on device, lack of secure delivery (un-encrypted in transit), lack of anti-rollback mechanisms, and lack of notifications of security changes due to updates.
5
Use of Insecure or Outdated Components
Use of deprecated or insecure software components/libraries that could allow the device to be compromised. This includes insecure customization of operating system platforms, and the use of third-party software or hardware components from a compromised supply chain.
6
Insufficient Privacy Protection
User's personal information stored on the device or in the ecosystem that is used insecurely, improperly, or without permission.
7
Insecure Data Transfer and Storage
Lack of encryption or access control of sensitive data anywhere within the ecosystem, including at rest, in transit, or during processing.
8
Lack of Device Management
Lack of security support on devices deployed in production, including asset management, update management, secure decommissioning, systems monitoring, and response capabilities.
9
Insecure Default Settings
Devices or systems shipped with insecure default settings or lack the ability to make the system more secure by restricting operators from modifying configurations.
10
Lack of Physical Hardening
Lack of physical hardening measures, allowing potential attackers to gain sensitive information that can help in a future remote attack or take local control of the device.

Metodologie di VAPT firmware



l'Internet of Things è stata definita come: "Uno sviluppo di Internet in cui gli oggetti di uso quotidiano avranno connettività di rete, consentendo loro di inviare e ricevere dati".

Il progetto OWASP Internet of Things è progettato per aiutare produttori, sviluppatori e consumatori a comprendere meglio i problemi di sicurezza associati all'Internet of Things e per consentire agli utenti in qualsiasi contesto di prendere decisioni migliori in materia di sicurezza durante la creazione, l'implementazione o la valutazione delle tecnologie IoT.
La Top 10 IoT OWASP ha l'obiettivo di fornire una lista delle 10 vulnerabilità IoT più pericolose. Grazie a questa lista sviluppatori, tecnici e utenti che usufruiscono delle tecnologie IoT possano mantenere controllati tutti i punti deboli dei dispositivi e garantire allo stesso tempo la migliore sicurezza.
Nello specifico, la presenza di queste 10 vulnerabilità IoT si annida in specifici punti dei dispositivi:

- interfaccia amministrativa
- autenticazione/login
- meccanismi di aggiornamento
- interfaccia web cloud
- servizi di rete del dispositivo

1. Password deboli e indovinabili o cablate nel codice

Le password autenticano un utente valido, consentendo l'accesso alle impostazioni di sicurezza di un dispositivo, ai poteri amministrativi e ai dati privati. Una creazione o gestione scadente delle password è un problema di sicurezza critico e continuo, soprattutto perché molti proprietari di dispositivi non modificano la password predefinita.

L'hardcoding semplifica la risoluzione dei problemi sui dispositivi remoti per sviluppatori o ingegneri, ma può essere facilmente utilizzato per l'accesso non autorizzato. Significa anche che se un hacker riesce a ottenere una password, può utilizzarla per entrare in ogni dispositivo simile. I produttori dovrebbero rimuovere tali backdoor e assicurarsi che ogni dispositivo sia dotato di un set di credenziali univoco. I dispositivi dovrebbero essere dotati di password predefinite complesse e non consentire l'impostazione di password deboli.

2. Servizi di rete non sicuri

Le funzionalità di connettività non sicure, come porte aperte o servizi non necessari, aumentano la superficie di attacco di un dispositivo IoT, portando alla possibilità di perdite di dati o esecuzione di codice remoto. I produttori di dispositivi possono affrontare queste vulnerabilità limitando i servizi di connettività al minimo necessario e utilizzando sempre protocolli di trasmissione sicuri.

3. Interfacce di ecosistema non sicure

Anche le interfacce con cui interagisce un dispositivo IoT possono essere interessate da gravi falle di sicurezza. Le interfacce Web, mobili, API backend o cloud offrono agli hacker l'accesso a informazioni significative sul software, sulle funzioni e sui dati di un dispositivo. Un'autenticazione debole consente agli hacker di ottenere un accesso non autorizzato tramite l'interfaccia di un dispositivo, mentre una crittografia scadente o filtri di input e output non protetti mettono a rischio i dati che il dispositivo invia e riceve.

4. Mancanza di un meccanismo di aggiornamento sicuro

Gli aggiornamenti sono un'arma fondamentale per affrontare le vulnerabilità della sicurezza dei dispositivi IoT, poiché gli sviluppatori li usano per eliminare bug e chiudere falle di sicurezza. Tuttavia, senza meccanismi di aggiornamento sicuri, gli aggiornamenti software e firmware possono effettivamente mettere a rischio i dispositivi. Gli aggiornamenti possono essere soggetti a manomissioni, sia alla fonte che in transito. Per evitare ciò, gli aggiornamenti devono essere firmati digitalmente, consegnati tramite canali sicuri e la firma deve essere verificata prima dell'applicazione. Inoltre, i produttori IoT devono includere meccanismi che impediscano agli hacker di annullare gli aggiornamenti e gli utenti devono essere informati di eventuali aggiornamenti di sicurezza urgenti.

5. Utilizzo di componenti non sicuri o obsoleti

La tecnologia legacy compromessa o non più aggiornabile rappresenta un'enorme minaccia per la sicurezza dei dispositivi IoT. I componenti non sicuri possono effettivamente incorporare difetti che gli hacker possono utilizzare per ottenere l'accesso a un'ampia gamma di dispositivi non correlati. Un esempio recente sono gli attacchi di esecuzione speculativa che colpiscono i processori Intel, ARM e AMD. La migliore difesa è quella di non utilizzare la tecnologia legacy e sostituirla il più rapidamente possibile. Nel caso di dispositivi legacy che non sono stati forniti con identità sicure, i produttori possono integrare la sicurezza dopo l'implementazione utilizzando servizi PKI specializzati che utilizzano una soluzione crittografica white-box per distribuire le chiavi in modo sicuro.

6. Protezione della privacy insufficiente

La protezione della privacy non è solo un buon comportamento aziendale; è anche un rischio di conformità importante. Una legislazione come il GDPR definisce le protezioni della privacy previste per tutte le aziende coinvolte nella tecnologia, compresi i produttori di dispositivi IoT. Per i dispositivi IoT, la protezione della privacy può essere una vulnerabilità della sicurezza a causa di un'archiviazione locale dei dati non sicura o persino della raccolta e archiviazione non autorizzata di dati personali.

7. Trasferimento e archiviazione dati non sicuri

Riguarda vulnerabilità relative alla scarsa crittografia dei dati e la mancanza di meccanismi di autenticazione. I dati possono essere esposti in varie fasi: a riposo, in trasmissione o durante l'elaborazione. Ciò offre agli hacker molteplici opportunità di rubare e comprendere i dati. Una crittografia debole, insieme a controlli di accesso scarsi o assenti, rende i dati di un dispositivo un bersaglio facile.

8. Mancanza di gestione dei dispositivi

Il monitoraggio dei dispositivi una volta che sono stati distribuiti è fondamentale per garantire un ambiente sicuro. Senza un'adeguata gestione delle risorse, diventa impossibile monitorare e difendere efficacemente le reti IoT tramite processi quali la gestione degli aggiornamenti, la dismissione sicura e la revoca dei certificati per i dispositivi compromessi in un'infrastruttura a chiave pubblica. Senza un quadro completo di ciò che sta accadendo con tutti i dispositivi IoT su una rete, diventa impossibile gestire le difese e le risposte alle minacce, rendendo tutti i dispositivi più vulnerabili.

9. Impostazioni predefinite non sicure

Le impostazioni predefinite devono sempre essere applicate tenendo a mente la sicurezza dell'utente finale e la sicurezza a lungo termine del dispositivo. Spesso, tuttavia, le impostazioni predefinite rappresentano un approccio "minimo indispensabile" o possono persino introdurre vulnerabilità, ad esempio password hardcoded o servizi esposti in esecuzione con permessi di root. I produttori dovrebbero dare agli amministratori dei dispositivi la possibilità di risolvere questi problemi, nonché di impostare e applicare permessi per impedire agli utenti di modificare le configurazioni senza la dovuta approvazione.

10. Mancanza di Hardening Fisico

È importante non trascurare il rafforzamento fisico del dispositivo contro gli attacchi che estraggono informazioni sensibili che potrebbero essere utilizzate in un attacco remoto o per ottenere il controllo del dispositivo. Alcune misure che possono essere adottate per rafforzare fisicamente un dispositivo includono la disattivazione o l'isolamento delle porte di debug, l'utilizzo dell'avvio sicuro per convalidare il firmware e la mancata memorizzazione di informazioni sensibili su una scheda di memoria rimovibile.

Estrazione del Firmware

Il primo passo per trovare vulnerabilità in un qualche tipo di dispositivo IoT è ottenere il suo firmware. 5-10 anni fa, era estremamente facile: il firmware di ogni dispositivo era disponibile sul sito Web del produttore. E mentre è ancora vero in alcuni casi, ora non esiste più una soluzione unica per tutti. Ora abbiamo diversi modi per ottenere il firmware, che vanno da molto facile a molto difficile.

Molto Facile: Download

Forse il produttore offre ancora il firmware da scaricare a chiunque ne abbia bisogno. Anche se non fosse così, a volte il firmware o alcune sue parti potrebbero essere trapelate o disponibili altrove. Quindi vale la pena cercare (consiglio di cercare specificamente su Github insieme ai soliti motori di ricerca/Google dork) e, se non viola le leggi locali, scaricarlo.

Facile/Moderato: MITM aggiornamento

Se l'opzione precedente non ha funzionato e il tuo dispositivo ha solo un pulsante luccicante etichettato "Installa aggiornamento", vale la pena premerlo, ma non prima di aver avviato un MITM corretto con acquisizione del traffico. Ci sono una varietà di strumenti tra cui scegliere, da Bettercap a Mitm-proxy.

L'idea qui è che nonostante l'adozione diffusa di HTTPS, alcuni sviluppatori potrebbero ancora utilizzare semplici CDN HTTP per distribuire gli aggiornamenti. Quindi, il pacchetto di aggiornamento può essere facilmente estratto dal traffico oppure puoi acquisire l'URL e seguirlo. Ma supponiamo che il tuo target utilizzi HTTPS. Allora abbiamo 2 scenari diversi:

1. Target autonomo

Se il target è autosufficiente e non ha app mobili o client desktop per il controllo, puoi comunque provare a fargli utilizzare il tuo certificato HTTPS autofirmato, con discrete possibilità di successo.

2. Target con app mobile

Molti dispositivi intelligenti non hanno una propria interfaccia utente e richiedono un'app per smartphone per il controllo. In alcuni casi, il download del firmware può essere implementato nell'app stessa. Puoi quindi installare il tuo certificato radice autofirmato sullo smartphone o sull'emulatore che stai utilizzando. Se le funzionalità di connettività Internet nell'app smettono di funzionare, ciò indica l'uso del pinning SSL. Naturalmente, ci sono varie tecniche per aggirare questo problema, come l'utilizzo di Frida.

Puoi vedere un tutorial qui:
https://infosecwriteups.com/hail-frida-the-universal-ssl-pinning-bypass-for-android-e9e1d733d29
https://redfoxsec.com/blog/ssl-pinning-bypass-android-frida/

Se non ti bastano, ti aspetta un po' di reverse engineering. Cerca di scoprire e capire come implementano il pinning SSL o, ancora meglio, come scaricano l'aggiornamento stesso (a volte scaricare le stringhe può richiedere molto tempo).

Moderato/Difficile: Ottenimento del firmware tramite una vulnerabilità

Ping a device

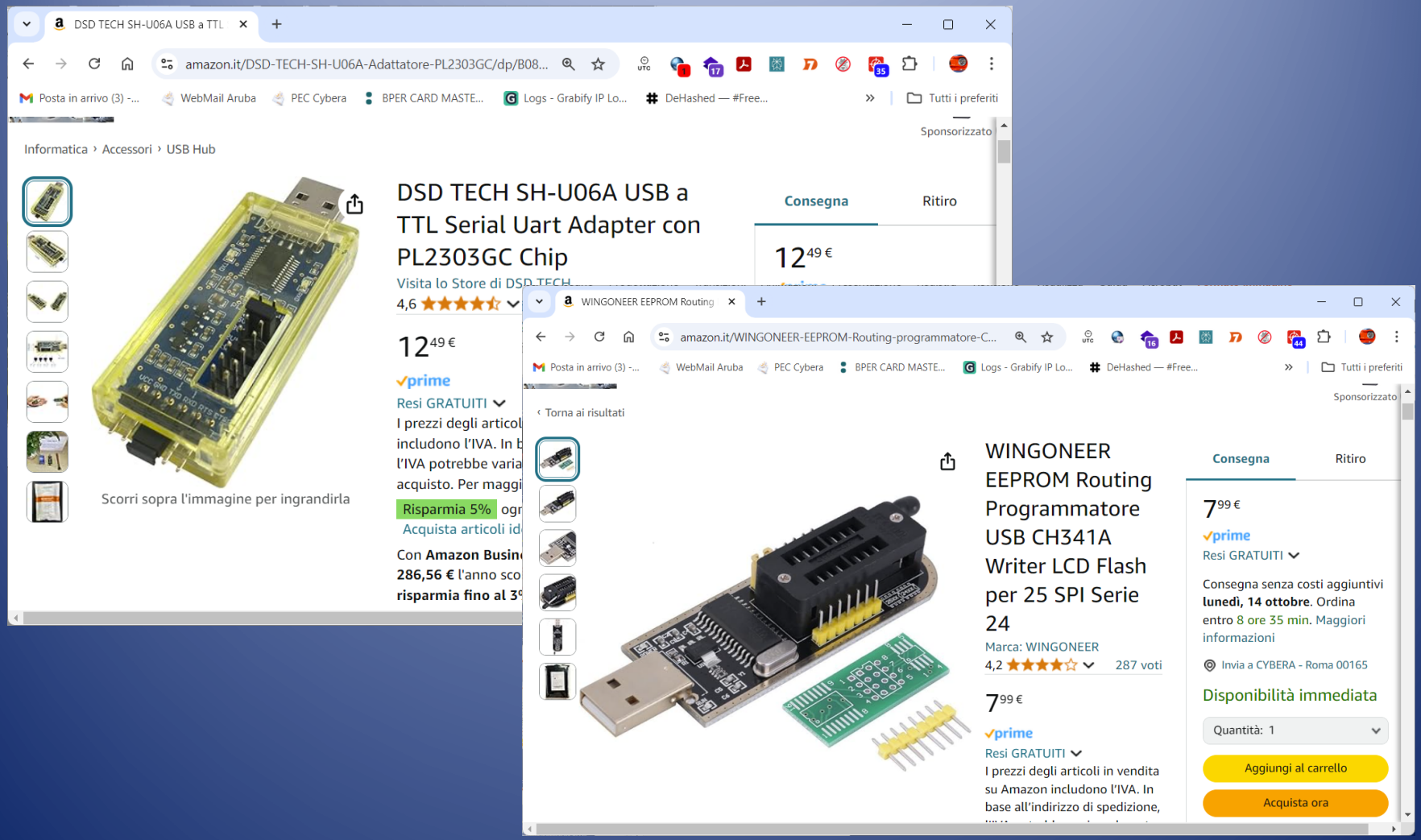
Enter an IP address: 172.217.161.14 && whoami Submit

```
Pinging 172.217.161.14 with 32 bytes of data:
Reply from 172.217.161.14: bytes=32 time=8ms TTL=118
Reply from 172.217.161.14: bytes=32 time=4ms TTL=118
Reply from 172.217.161.14: bytes=32 time=3ms TTL=118
Reply from 172.217.161.14: bytes=32 time=4ms TTL=118

Ping statistics for 172.217.161.14:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 3ms, Maximum = 8ms, Average = 4ms
deepak-pc\deepa
```

Alcuni obiettivi possono essere molto speciali: dispositivi in fase di sviluppo, dispositivi senza funzionalità di aggiornamento software o dispositivi che richiedono un tecnico altamente pagato e certificato dal produttore che utilizzi strumenti speciali per aggiornarli.

Estrazione del Firmware

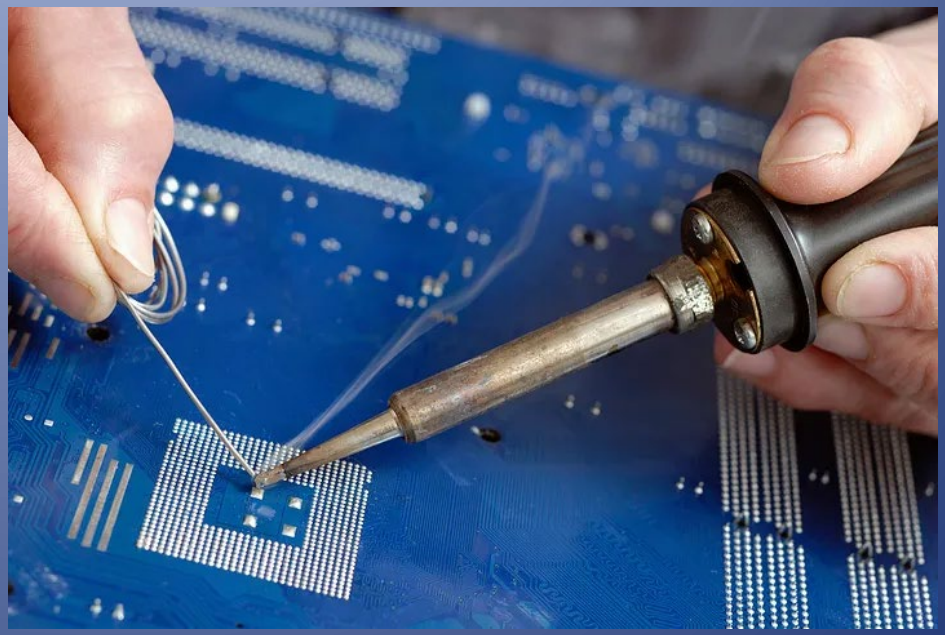


Il tuo dispositivo potrebbe avere una vecchia vulnerabilità non patchata che ti consente di ottenere una shell di root (o qualsiasi altro tipo di esecuzione di codice) e scaricare il firmware. Se c'è anche un exploit pubblico per il bug, sarai avantaggiato, altrimenti, dovrai sviluppare un exploit da solo, ma è già molto più facile quando hai anche una descrizione di base della vulnerabilità.

Le vulnerabilità più pericolose, temute e preziose sono quelle che possono essere sfruttate senza autenticazione. La maggior parte dei ricercatori di sicurezza le cerca e la maggior parte dei produttori si concentra sulla protezione dei propri dispositivi da esse. Ma poi abbiamo vulnerabilità che possono essere sfruttate dopo l'autenticazione, diciamo anche con diritti di amministratore. In quel caso, abbiamo una superficie di attacco più ampia abbinata a obiettivi più facili.

Difficile/Molto Difficile: Estrazione Hardware

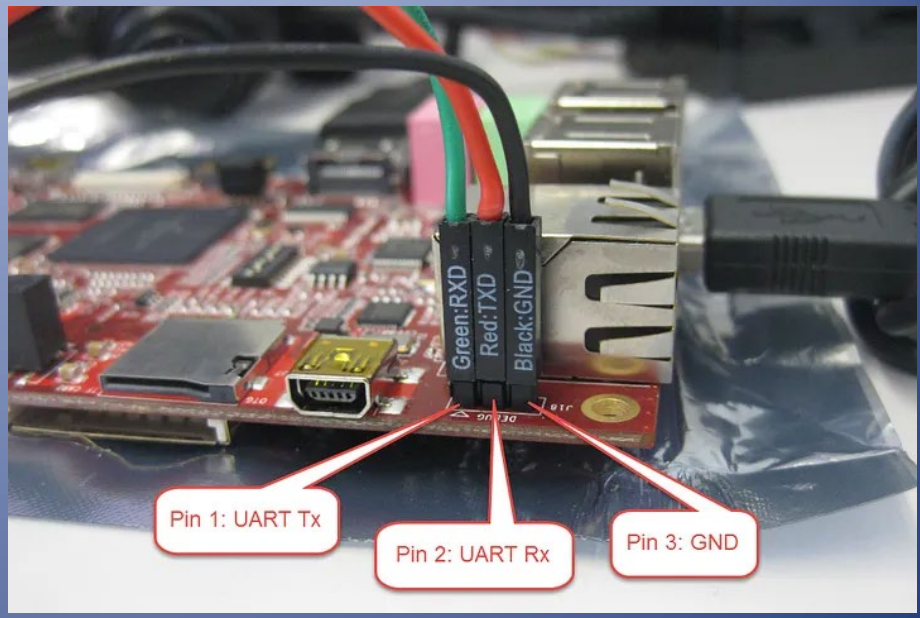
Il metodo hardware è generalmente più difficile, ma potrebbe comunque essere più semplice dell'opzione precedente, in alcuni casi.
Si divide in due categorie: cercare di trovare e sfruttare vulnerabilità hardware e leggere il firmware direttamente dalla memoria flash.



Vulnerabilità hardware
Esistono vari tipi di vulnerabilità hardware, ma qui siamo interessati principalmente alle interfacce di debug esposte come le shell UART o JTAG/SWD.

UART Shells

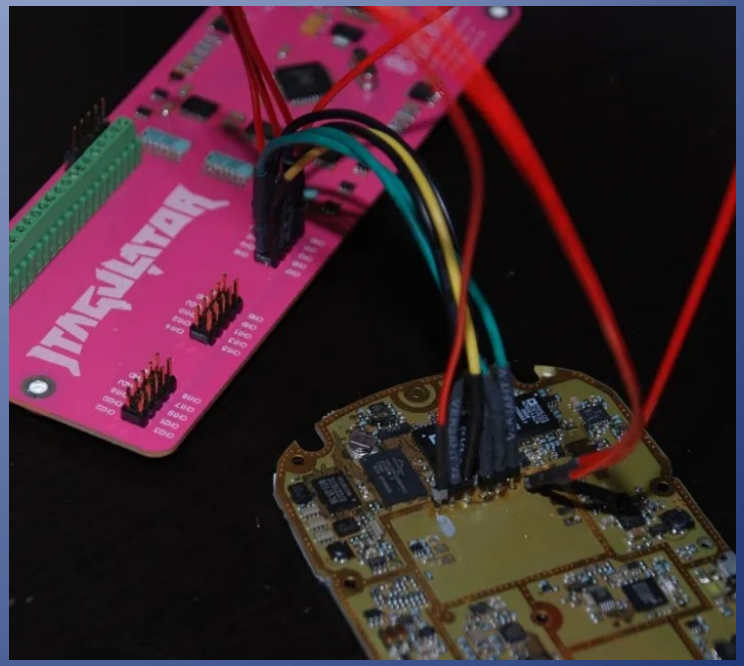
Le interfacce UART sono una cosa comune nei dispositivi IoT, spesso lasciate accessibili per la diagnostica. Toccando queste porte UART, a volte puoi ottenere l'accesso alla console (e sto parlando di shell root/bootloader). La chiave sta nell'identificare i pin TX, RX e GND, e quindi usare un convertitore seriale-USB e un emulatore di terminale per comunicare con il dispositivo. Questo metodo può essere un modo semplice ed efficace per iniziare a svelare il funzionamento interno di un dispositivo IoT.



Alcuni link sull'argomento:
https://voidstarsec.com/blog/uart-uboot-and-usb
https://dfresh.ninja/index.php/2021/01/16/exploring-uart-protocol-and-dumping-firmware-on-the-tenda-300-wireless-router/

Interfacce JTAG/SWD

Progettate per il test e il debug a livello di chip, queste interfacce offrono un accesso di basso livello alla CPU di un dispositivo. Individuare i pin JTAG o SWD su un PCB imballato è il primo passo (supponendo che siano esposti), seguito dall'interfacciamento con un debugger come J-Link o ST-Link. Una volta connesse, queste interfacce consentono di interrompere l'esecuzione del firmware, ispezionare la memoria e così via. Alcuni produttori lo sanno e cercano di disabilitare o nascondere queste interfacce, proteggerle con password, ecc. Ma, ovviamente, alcuni non lo fanno.



Questi post potrebbero aiutarti a capire JTAG:
https://sergioprado.blog/2020-02-20-extracting-firmware-from-devices-using-jtag/
https://wrongbaud.github.io/posts/jtag-hdd/

Fun with Flash Chips

Questo firmware che stiamo cercando deve essere archiviato da qualche parte. Per la maggior parte dei dispositivi intelligenti in circolazione, questo da qualche parte sarà un chip flash. Se è archiviato lì, può essere letto da lì. Infatti, questo è ciò che fa la CPU del tuo target quando si avvia! Ma non abbiamo accesso alla CPU, quindi dobbiamo trovare un altro modo, che consiste nell'interfacciare direttamente il chip o dissaldare il chip flash dal PCB e leggerlo utilizzando un dispositivo speciale.

Flash Sniffing/Controlling Through SPI/I2C/ecc.

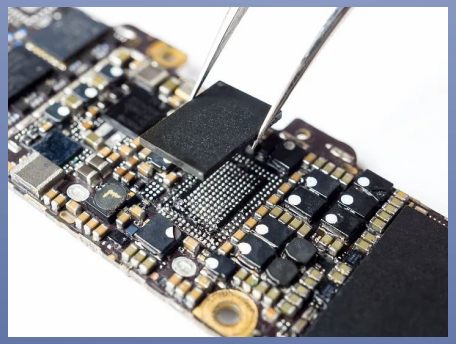
I chip flash comunicano con CPU, MCU e SoC tramite protocolli come SPI. Questi protocolli sono standard, quindi possiamo usarli anche noi. Quando un dispositivo si accende, la sua CPU legge la memoria flash, il momento perfetto per sniffare i dati tramite un analizzatore logico. Ma ecco il trucco: a seconda del dispositivo, può leggere solo una parte del firmware. Quindi potresti aver bisogno di un controllo completo del chip per leggerlo.

Ecco una selezione di post che approfondiscono l'argomento:
https://ivanorsolic.github.io/post/hardwarehacking1/
https://securityboulevard.com/2023/02/exploiting-embedded-apis-by-dumping-firmware/
https://jcjc-dev.com/2016/06/08/reversing-huawei-4-dumping-flash/
https://www.nsideattacklogic.de/en/dumping-spi-flash-memory-of-embedded-devices-2/

Di origine, un chip flash ha bisogno di alimentazione per funzionare. Senza dissaldare e alimentare una gamba del chip, potrebbe riattivare l'intero PCB, causando un tira e molla sulla flash tra te e la CPU. Una soluzione alternativa? Dissaldare la gamba VCC: meno lavoro, ma più possibilità di danneggiare il chip. Un altro approccio è dissaldare l'intero chip; renderà anche più comodo utilizzarlo in un programmatore flash.

Flash Chips in QFP, TSOP and Other Non-BGA Packages

Per un flash in un QFP (Quad Flat Package), il processo prevede di riscaldare attentamente la saldatura per rimuovere il chip senza danneggiare il PCB o il chip stesso. Una volta rimosso il chip, puoi leggerne il contenuto utilizzando un programmatore flash.
Uno dei trucchi è quello di utilizzare una lega con un basso punto di fusione.



Questa tecnica prevede l'applicazione di una lega speciale alle gambe del chip. Questa lega si mescola con la saldatura standard, riducendone il punto di fusione complessivo.
Consente di sollevare il chip dalla scheda senza danneggiare l'intera area per eccessivo riscaldamento.
Alcune letture sullo prelievo del firmware con la dissaldatura:

https://astrid.tech/2022/07/13/0/blink-mini-dumping/
https://fastcall.medium.com/dumping-firmware-from-a-router-5d7e819199fd

eMMC/SPI Flash in BGA Package

La flash eMMC nei package BGA (Ball Grid Array) presenta una serie di sfide. La dissaldatura di un package BGA richiede precisione e abilità, poiché le sfere di saldatura si trovano sotto il chip, rendendo difficile riscaldarle in modo uniforme senza danneggiare il chip o la scheda. Dopo la rimozione, viene utilizzato un adattatore socket BGA o un metodo di saldatura diretta su un lettore flash per interfacciarsi con il chip. Questo approccio richiede attrezzature specializzate e una mano ferma, ma è un modo affidabile per accedere direttamente al firmware. Ulteriori letture se sei abbastanza coraggioso da affrontare il BGA:

https://blog.quarkslab.com/flash-dumping-part-i.html
https://riverloopsecurity.com/blog/2020/03/hw-101-emmc/

UFS Flash in BGA Package

Questo è ancora più difficile del precedente. Ma se il caso precedente è stato difficile per le tue mani, questo lo sarà anche per il tuo portafoglio. Il fatto è che UFS è un nuovo protocollo molto più veloce per la memoria flash. E questo rende i programmatori flash UFS molto più rari e molto più costosi. Non sarei sorpreso se qualcuno si costruisse da solo un programmatore UFS a basso costo nel proprio garage, ma per ora, abbiamo quello che abbiamo. O forse c'è un modo per riutilizzare una chiavetta USB con chip flash UFS a bordo in un lettore flash improvvisato?
Non ho esperienza personale con lo scarico di UFS né sono riuscito a trovare una fonte decente per questo, ma ecco un link a un programmatore flash che afferma di supportare UFS:
https://evision-webshop.eu/dediprog/nuprog-e2-engineering-universal-programmer

Edge Cases

In alcuni scenari, i produttori implementano misure di sicurezza aggiuntive come la protezione read-out, la crittografia del firmware a riposo o meccanismi di protezione unici. Queste misure richiedono un approccio più esoterico. Ognuno di questi casi limite richiede un approccio unico, che spesso comporta una combinazione di competenze di reverse engineering hardware e software per aggirare queste protezioni.

Link utili:

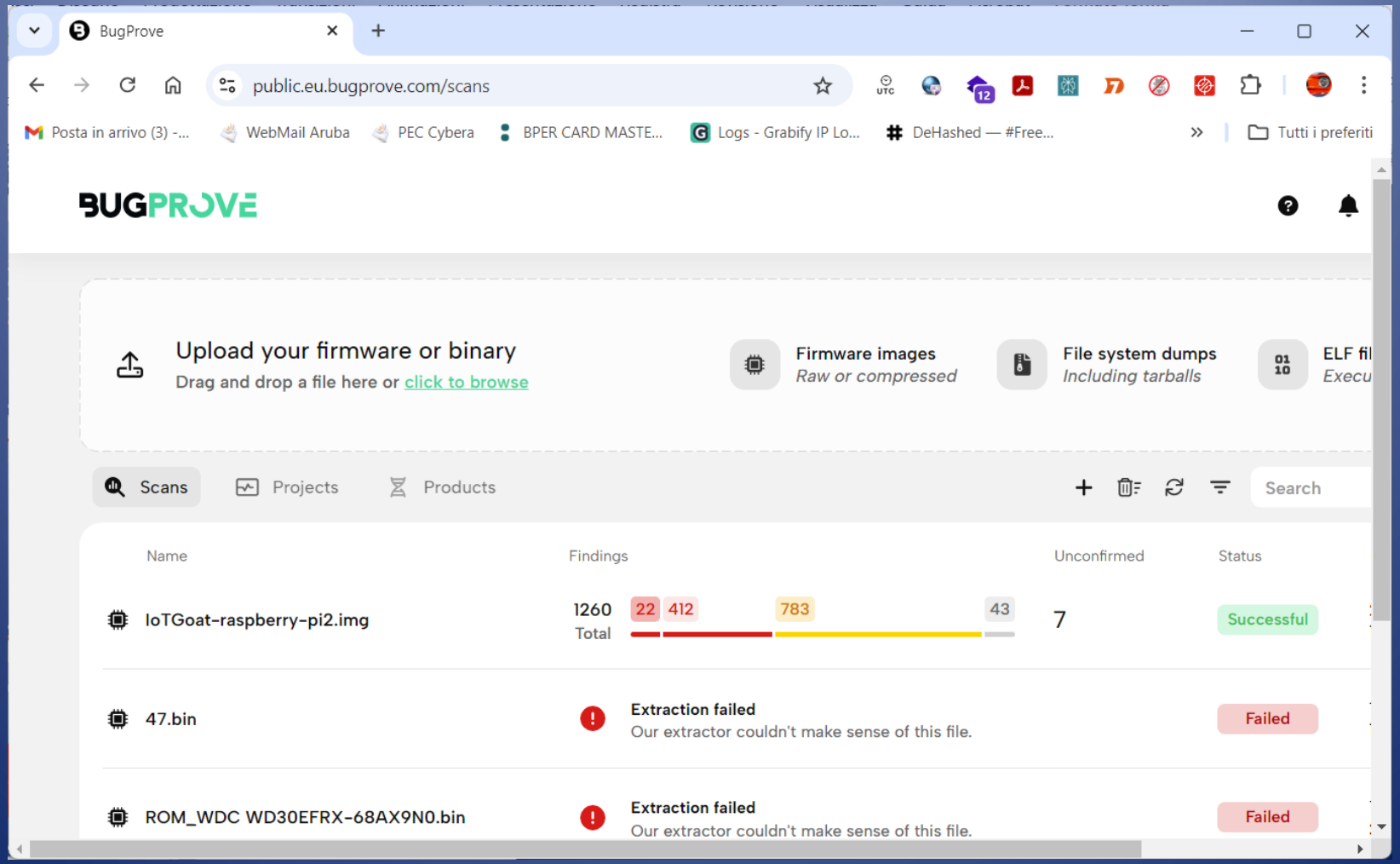
https://ethicalhacs.com/dvwa-command-injection/
https://www.darknet.org.uk/2016/03/bettercap-modular-portable-mitm-framework/
https://www.sentryair.com/blog/industry-applications/electronics-technology/the-hazards-of-solder-fumes/
https://www.rapid7.com/blog/post/2022/04/07/lessons-in-iot-hacking-how-to-dead-bug-a-bga-f lash-memory-chip/
https://mcuoneclipse.com/2014/07/21/terminal-connection-to-the-riot-board/
https://hackaday.com/2016/12/15/the-many-faces-of-jtag/
https://jcjc-dev.com/2016/05/23/reversing-huawei-3-sniffing/
https://www.tarlogic.com/blog/hardware-hacking-chip-off-for-beginners/
https://www.eetimes.com/jedec-adds-ufs-card-spec/

Riepilogo

1. Ricercare informazioni sul firmware (OSINT e Intelligence)
2. Ottenere informazioni su hardware ed eventuali vulnerabilità già conosciute
3. Ottenere il firmware
4. Capire il formato del firmware, compresso vs cifrato, ecc.
5. Ottenere il file system del firmware
6. Analizzare i singoli elementi, fare reverse engineering e code review
7. Trovare i difetti o le vulnerabilità
8. Testare le vulnerabilità su un emulatore o sul device stesso
9. Scrivere un report dettagliato che documenta il lavoro svolto, foto e screenshot saranno utili per documentare il lavoro svolto.

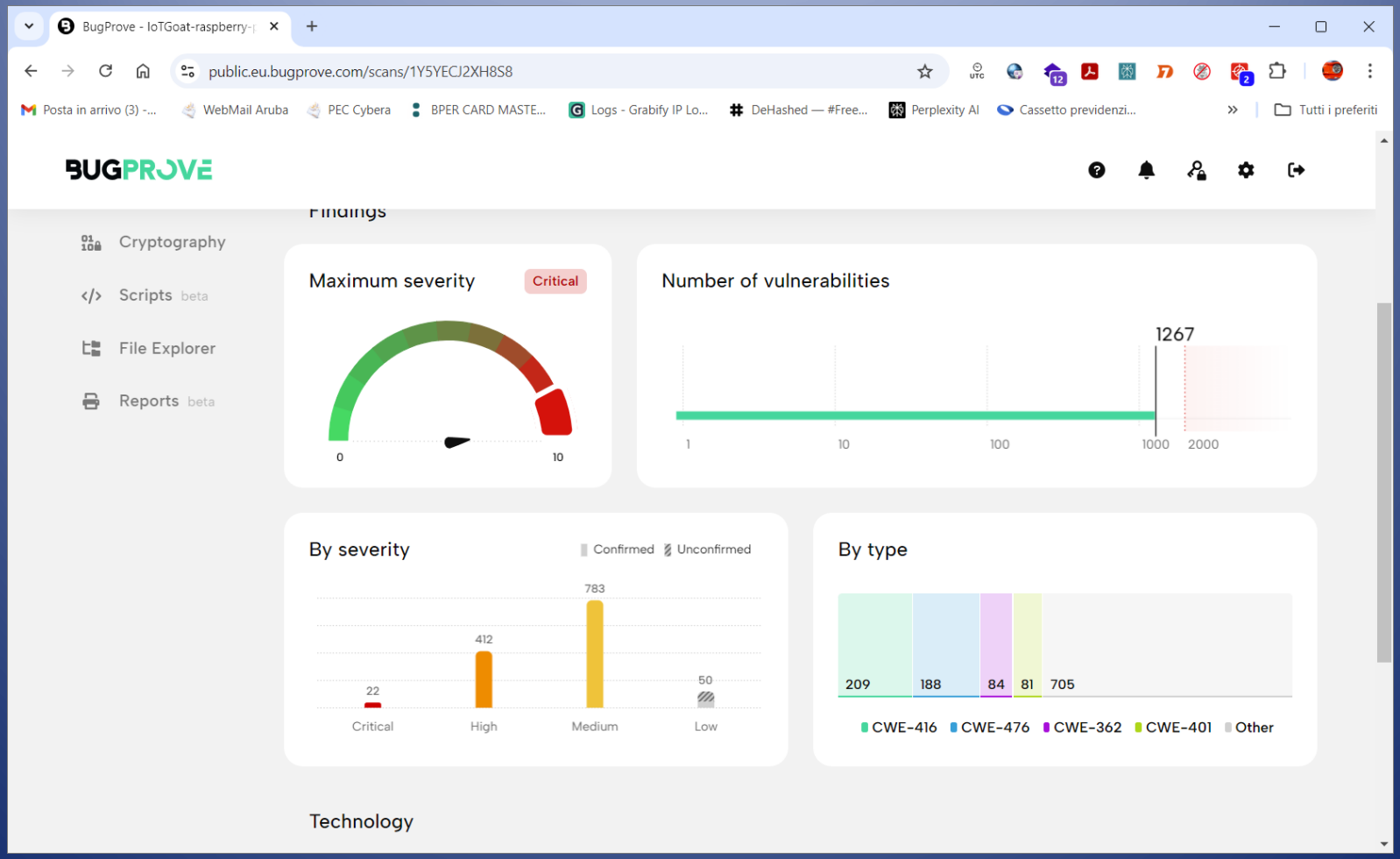
Hack the Firmware

tecniche e strumenti di analisi firmware



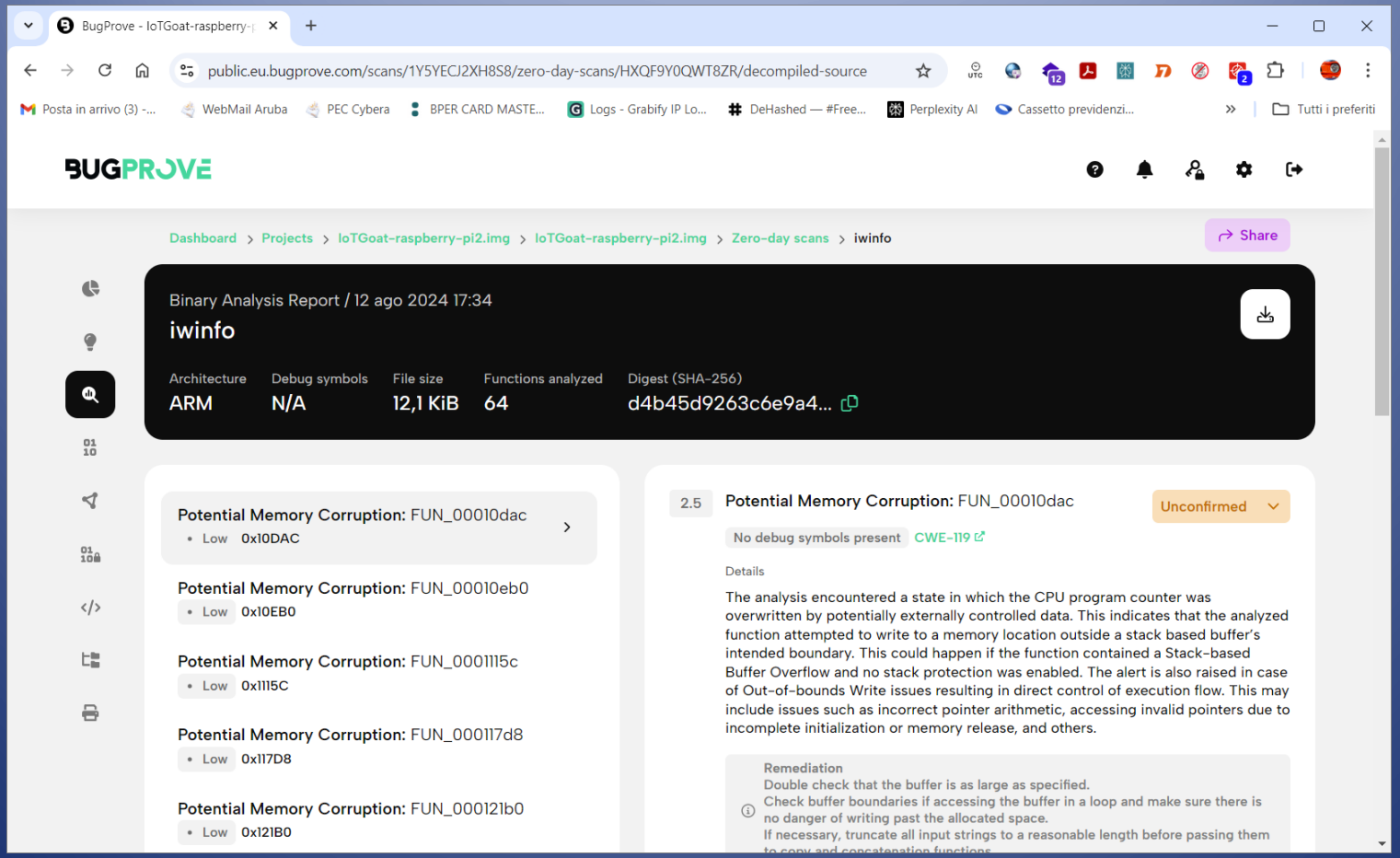
Hack the Firmware

tecniche e strumenti di analisi firmware



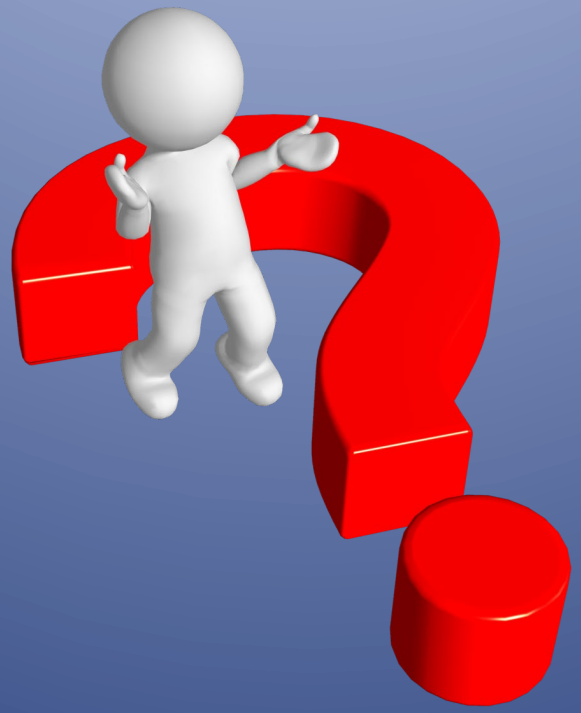
Hack the Firmware

tecniche e strumenti di analisi firmware



Hack the Firmware

tecniche e strumenti di analisi firmware



Hack the Firmware

tecniche e strumenti di analisi firmware

Link utili:
https://github.com/OWASP/IoTGoat
https://mindstormsecurity.com/product/offensive-hardware-hacking-training/
https://www.youtube.com/watch?v=zbUuBZJIHkE
https://www.whid.ninja
https://owasp.org/www-project-iot-security-testing-guide/
https://owasp.org/owasp-istg/03_test_cases/firmware/index.html
https://owasp.org/www-project-web-security-testing-guide/
https://github.com/scriptingxss/owasp-fstm
https://owasp.org/www-project-mobile-app-security/
https://www.iotpentestingguide.com/
https://link.springer.com/book/10.1007/978-1-4842-4300-8
https://www.packtpub.com/en-us/product/iot-penetration-testing-cookbook-9781787280571
https://nostarch.com/practical-iot-hacking
https://github.com/attify/firmware-analysis-toolkit

Hack the Firmware

tecniche e strumenti di analisi firmware

GRAZIE
per avermi ospitato

Chi desidera rimanere in contatto: massimiliano.graziani@cybera.it

Sito https://www.cybera.it

Public Profile https://it.linkedin.com/in/mgraziani