FIRMWARE CREDENTIAL EXTRACTION

HENDRICK STRÖNGREEN
@strongreen

O QUE VOCÊ VAI APRENDER HOJE?

- Firmware
- Ferramentas
- Como utilizá-la
- Tipos de sistemas de arquivos
- Buscar credenciais
- Encontrando resultados
- Analisando arquivos
- Automatizando o processo (parte 1)
- Modificar um firmware
- Montar um firmware modificado
- Emular firmware com o QEMU
- Próximos passos
- Como poderia ser mitigado

Rio Maker Space

FIRMWARE

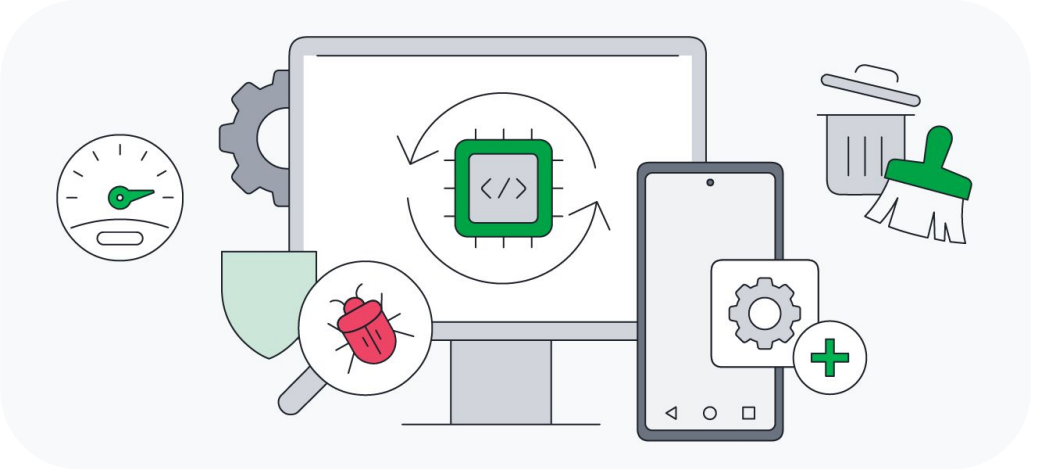
Rio Maker Space

O que é um Firmware?

É um software que fica diretamente no hardware e controla seus funcionamentos básicos. Basicamente ele fica entre o hardware e o software de alto nível.

Exemplos:

- Roteador
- Smart TV
- Periféricos do computador
- Dispositivos IoT

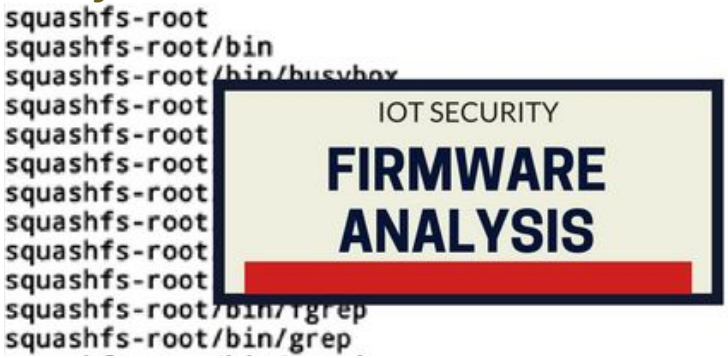


FIRMWARE

Rio Maker Space

Por que analisar Firmwares?

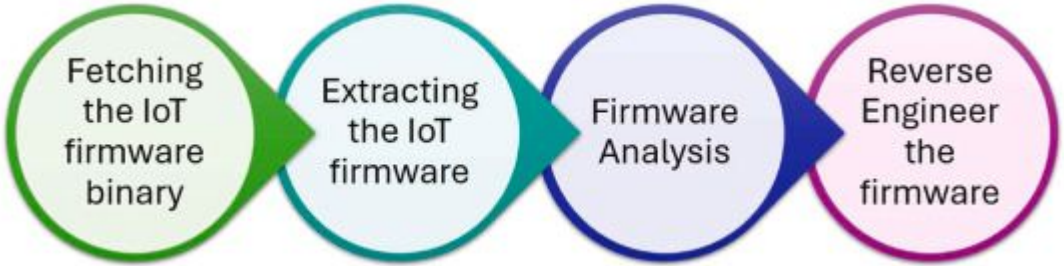
- Firmwares controlam dispositivos essenciais
- Fabricantes armazenam as credenciais *hardcoded*, ou seja, diretamente no código
- Engenharia reversa ajuda a identificar vulnerabilidades



FERRAMENTAS UTILIZADAS PARA ISSO

As ferramentas mais conhecidas usadas para analisar Firmwares são:

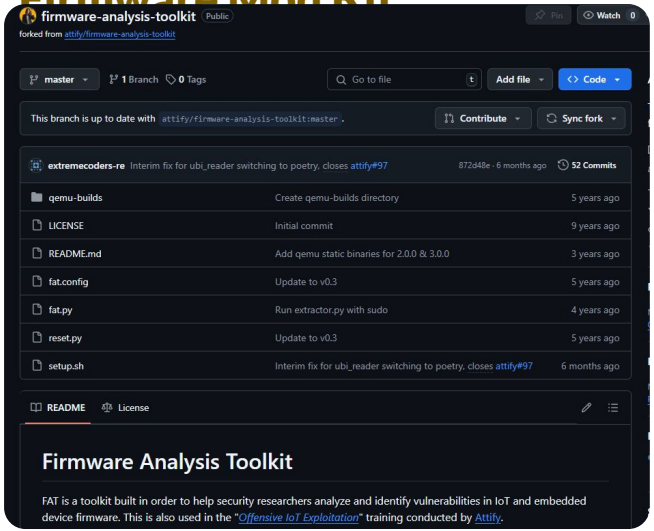
- **Binwalk**
  - **Analisar e extrair Firmwares**
- **Strings**
  - **Extração de textos legíveis do binário**

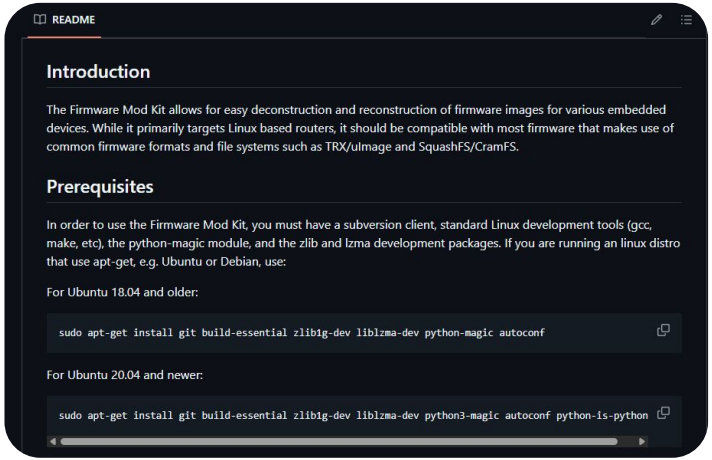


COMO INSTALAR AS FERRAMENTAS

Faça um git clone desses dois repositórios e seguir o passo a passo do Readme:

- **Firmware Analysis toolkit**
- **Firmware Mod Kit**





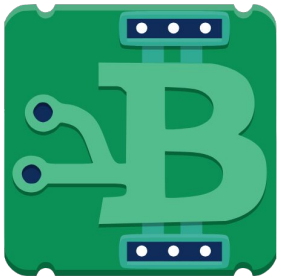
Tem o fork do projeto no meu github: strongreen

BINWALK

Rio Maker Space

É uma ferramenta de código aberto usada para análise, engenharia reversa e extração de imagens de firmware.

Ele identifica assinaturas de arquivos em binários, extrai sistemas de arquivos embutidos, descomprime dados compactados e recupera dados perdidos ou fragmentados — útil até para partes corrompidas ou escondidas.



BINWALK

Rio Maker Space

Parâmetros utilizado e o que ele faz?

-M (merge) : Analisa não só o arquivo principal, mas também os arquivos extraídos.

-e (extract) : Extrai todos os dados que ele identificar.

Como fazer o comando e a estrutura após a extração:

```
┌──(strongreen㉿kali)-[~]
└─$ binwalk -Me firmware.bin
```

```
firmware.bin
└── _firmware.bin.extracted
    ├── squashfs-root
    │   ├── etc
    │   │   ├── passwd
    │   │   └── shadow
    │   ├── bin
    │   │   └── busybox
    │   └── www
    │       └── config.html
    └── 1E7200
        └── zlib compressed data
```

EXTRAIR OS ARQUIVOS DO SQUASHFS

Rio Maker Space

Suporta diferentes métodos de compressão como: gzip, lzma, xz | Foco no FS |
Usado quando não é possível realizar a extração com o binwalk

```
┌──(strongreen㉿kali)-[~]
└─$ /firmware-mod-kit/unsquashfs_all.sh new_firmware
```

Se tudo ocorrer bem você verá uma pasta squashfs-root/ com os arquivos do firmware como as pastas: bin, etc, lib

Rio Maker Space

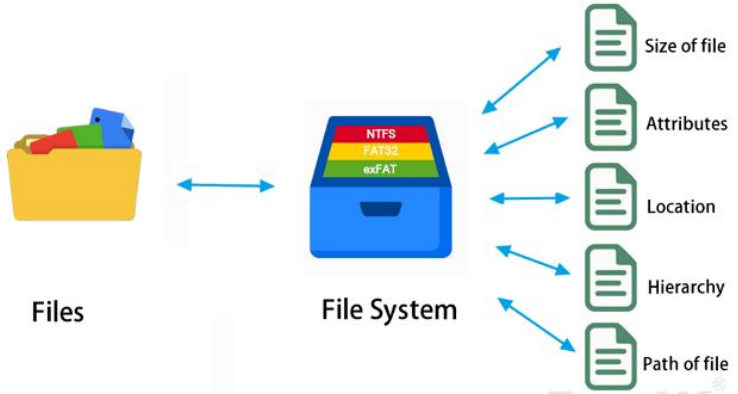
TIPOS DE SISTEMAS DE ARQUIVOS

Em dispositivos embarcados, o sistema de arquivos (FS) é projetado para operar com espaço limitado, memória não volátil e inicialização rápida.

Eles costumam ter compactação de dados, tolerância a falhas e otimização para leitura/escrita sequencial, garantindo eficiência e estabilidade.

Exemplos comuns incluem:

- **SquashFS**
- **CramFS**
- **JFFS2**
- **YAFFS2 e ext2**



SQUASHFS

Rio Maker Space

É um sistema de arquivos compacto somente de leitura, usado para criar imagens comprimidas de sistema de arquivos. Ele usa técnicas de compressão (como gzip ou LZMA).

É muito usado em distribuições de Linux embarcado, como em roteadores, onde a integridade e a eficiência do armazenamento são importantes.

```
00000000  68 73 71 73 ...
```

68 73 71 73 traduzido pra texto ASCII dá hsqs, mas dependendo da codificação, pode aparecer invertido como "shsq".

O QUE É INTERESSANTE PROCURAR?

- /etc/passwd e /etc/shadow (Linux)
- Arquivos de configuração: config.xml, settings.json
- Chaves SSH (id_rsa)
- Scripts com credenciais ou endpoints
- Backdoors e serviços ocultos

Rio Maker Space

STRINGS

Extrai texto legível de binários para encontrar senhas, URLs e configs.

Exemplo:

```
┌──(strongreen㉿kali)-[~]
└─$ strings firmware.bin | grep -iRn 'password' .
```

-i → Desabilita key sensitive (password, Password, PASSWORD, etc.)

-R → Busca recursiva em todos os arquivos e pastas

-n → Mostra o número da linha onde encontrou a palavra

. → O ponto representa "a pasta atual", ou seja, busca em todos os arquivos dentro da pasta extraída

Rio Maker Space

ANALISANDO CREDENCIAIS

Procurando por palavras-chave comuns:

-E → Expressão regular estendida (Extended Regex): Permite usar o "OR" (|) para buscar várias palavras ao mesmo tempo.

Exemplo:

```
┌──(strongreen㉿kali)-[~]
└─$ strings firmware.bin | grep -riE "password|admin|root|key|token" extracted_firmware/
```

Outras dicas:

- **Procurar por base64 ou hex codificado**
- **Checar scripts como o init.d**

SCRIPT PARA AUTOMAÇÃO COM O BINWALK

- Analisa o firmware
- Extrai recursivamente
- Procura por palavras-chave:
  - password
  - admin
  - root
  - etc
- Gera um log bonitinho no terminal

```python
import os
import subprocess
from rich import print

def run_command(command):
    result = subprocess.run(command, shell=True, capture_output=True, text=True)
    return result.stdout

import sys
if len(sys.argv) < 2:
    print("[red][!] Usar o comando: python biwalk-script.py <arquivo_do_firmware>[/red]")
    sys.exit(1)

firmware = sys.argv[1]
output_dir = "extracted_firmware"

os.makedirs(output_dir, exist_ok=True)

print("[cyan][+] Analisando o firmware com binwalk...[/cyan]")
print(run_command(f"binwalk {firmware}"))

print("[yellow][+] Extraindo o conteúdo do firmware...[/yellow]")
print(run_command(f"binwalk -Me {firmware} -C {output_dir}"))

print("[green][+] Procurando por credenciais e palavras-chave...[/green]")
credentials_found = run_command(f"grep -riE 'password|admin|root|key|token' {output_dir}")

with open("credentials_found.txt", "w") as file:
    file.write(credentials_found)

if credentials_found.strip():
    print("[red][!] Credenciais encontradas! Confira o arquivo credentials_found.txt[/red]")
else:
    print("[yellow][-] Nenhuma credencial óbvia encontrada. Tente analisar manualmente.[/yellow]")

print("[cyan][+] Análise concluída! Confira a pasta:[/cyan]", output_dir)
```

ANALISANDO E EXTRAINDO O FIRMWARE

Rio Maker Space

Identificar offset (unsquashfs) :

```
┌──(strongreen㉿kali)-[~]
└─$ strings firmware.bin
```

```
┌──(strongreen㉿kali)-[~]
└─$ hexdump -C firmware.bin | less | grep -iR "shsq|hsqs"
```

Extrair os arquivos para analisar:

```
┌──(strongreen㉿kali)-[~]
└─$ binwalk -Me firmware.bin
```

```
firmware.bin
└── _firmware.bin.extracted
    ├── squashfs-root
    │   ├── etc
    │   │   ├── passwd
    │   │   └── shadow
    │   ├── bin
    │   │   └── busybox
    │   └── www
    │       └── config.html
    └── 1E7200
        └── zlib compressed data
```

DEMONSTRAÇÃO PRÁTICA

Rio Maker Space

Roteiro:

1. Baixar
2. Analisar
3. Extrair
4. Buscar credenciais
    a. telnet, passwd, shadow
    b. pegar o usuário e senha
5. Mostrar os resultados

AGORA VAMOS PARA A PRÁTICA.

RESULTADOS

```
┌──(strongreen㉿kali)-[~/…/TL-WR841N(EU)_V13_171019 :: /_TL-WR841.bin.extracted/
squashfs-root/etc]
└─$ cat passwd.bak
admin:$1$$iC.dUsGpxNNJGeOm1dFio/:0:0:root:/:/bin/sh
dropbear:x:500:500:dropbear:/var/dropbear:/bin/sh
nobody:*:0:0:nobody:/:/bin/sh
```

```
Dictionary cache built:
* Filename..: ../../../rockyou.txt
* Passwords.: 14344391
* Bytes.....: 139921497
* Keyspace..: 14344384
* Runtime...: 2 secs

$1$$iC.dUsGpxNNJGeOm1dFio/:1234

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 500 (md5crypt, MD5 (Unix), Cisco-IOS $1$ (MD5))
Hash.Target......: $1$$iC.dUsGpxNNJGeOm1dFio/
Time.Started.....: Fri Mar 28 01:42:45 2025 (1 sec)
Time.Estimated...: Fri Mar 28 01:42:46 2025 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (../../../rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:     2420 H/s (10.59ms) @ Accel:32 Loops:1000 Thr:1 Vec:16
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 1280/14344384 (0.01%)
Rejected.........: 0/1280 (0.00%)
Restore.Point....: 1024/14344384 (0.01%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1000
Candidate.Engine.: Device Generator
Candidates.#1....: kucing -> poohbear1
Hardware.Mon.#1..: Util: 22%
```

1. Descobrir a senha do admin
2. Verificar SSH
3. UID 0 - Backdoor? Verificar se tem realmente acesso ou está desativado, testar na emulação

Rio Maker Space

PREPARANDO PARA MONTAR

Rio Maker Space

Remover lixo inicial (Headers) ou extrair uma partição específica para montar o firmware corretamente (VALOR DO DECIMAL: binwalk):

```
┌──(strongreen㉿kali)-[~]
└─$ dd if=firmware.bin skip=(valor do decimal) bs=1 of=new_firmware.bin
```

Validar se o firmware foi extraído corretamente:

```
┌──(strongreen㉿kali)-[~]
└─$ file new_firmware.bin
```

Squashfs filesystem ou data

MODIFICAR

- Editar a página de login (HTML/CSS/JS)
- Modificar senhas (/etc/passwd ou /etc/shadow)
- Alterar scripts de inicialização (/etc/init.d/)
- Inserir um backdoor ou customizar um script

RECRIAR O SQUAHFS

```
┌──(strongreen㉿kali)-[~]
└─$ mksquashfs squashfs-root/ new-rootfs.squashfs -comp xz -b 131072
```

mksquashfs → Criar um sistema de arquivos SquashFS.

squashfs-root/ → A pasta que contém o FS extraído e modificado.

new-rootfs.squashfs → Nome do novo FS.

-comp xz → Define o método de compressão como XZ (é mais eficiente e gera arquivos menores).

-b 131072 → Define o tamanho do bloco de compressão para **131072** bytes (128 KB).

Tamanhos maiores podem melhorar a compressão, mas exigem mais RAM ao descompactar.

MONTAR NOVO FIRMWARE

```
┌──(strongreen㉿kali)-[~]
└─$ cat bootloader.bin kernel.img new-rootfs.squashfs > firmware_modificado.bin
```

cat → Concatena arquivos

bootloader.bin → Contém o bootloader, que inicializa o roteador (U-Boot)

kernel.img → O kernel do sistema Linux embarcado no roteador (LZMA)

new-rootfs.squashfs → O FS modificado (slide anterior)

> → Redireciona para novo arquivo.

firmware_modificado.bin → O novo firmware pronto para ser testado.

Esse comando deixa os arquivos no formato esperado pelo roteador

[BOOTLOADER] + [KERNEL] + [ROOTFS]

Rio Maker Space

EMULAR COM O QEMU

```
┌──(strongreen㉿kali)-[~]
└─$ binwalk -A firmware.bin
```

MIPS:

qemu-system-mips | qemu-system-arm → **Chama o QEMU para emular um sistema com arquitetura MIPS | ARM**

```
┌──(strongreen㉿kali)-[~]
└─$ qemu-system-mips -M malta -kernel vmlinux -hda firmware_modificado.bin -append "console=ttyS0"
```

-M malta → **Define o modelo da placa de hardware a ser emulado (malta é um tipo de plataforma MIPS usada para desenvolvimento)**

-M vexpress-a9 → **Usa a plataforma ARM Cortex-A9 (ARM)**

ARM:

```
┌──(strongreen㉿kali)-[~]
└─$ qemu-system-arm -M vexpress-a9 -kernel vmlinux -m 256M -drive file=firmware_modificado.bin,format=raw -append "console=ttyAMA0"
```

Rio Maker Space

EMULAR COM O QEMU

-kernel vmlinux → Especifica o kernel Linux que será usado no emulador (Se o kernel estiver embutido no firmware, não é necessário passar isso)

-hda firmware_modificado.bin → Usa o firmware modificado como disco principal (hda = Hard Disk A)

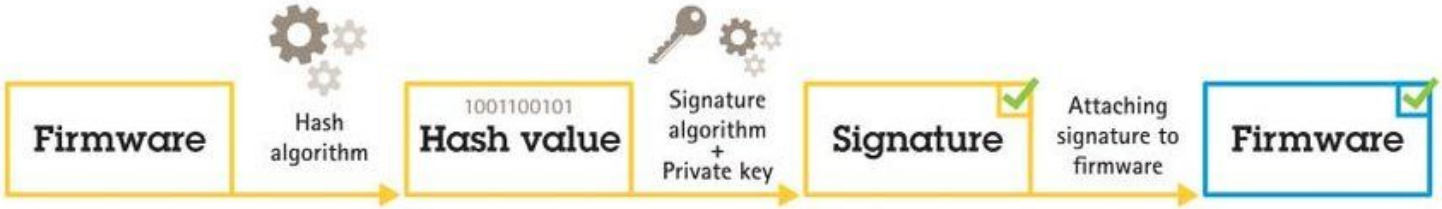
-append "console=ttyS0" → Passa parâmetros para o kernel durante a inicialização "console=ttyS0" | "console=ttyAMA0" (ARM) → Diz ao kernel para enviar a saída para ttyS0, porta serial virtual do emulador | permitindo ver os logs do sistema no terminal do QEMU

-drive file=firmware_modificado.bin, format=raw → Usa o firmware como disco (ARM)

Rio Maker Space

COMO PODERIA SER EVITADO?

- Hardcoded de senhas
- Dados criptografados
- Testes de segurança antes do lançamento
- Atualização de Firmwares



PRÓXIMOS PASSOS

- Analisar partições para emular o firmware com o QEMU, executando comandos, buscando mais informações e entendendo o comportamento.
- Modificar o firmware e após isso recompilá-lo e testar emulado.
- Aprender, analisar e extrair partições com outro sistema de arquivo, como o unyaffs yaffs.img
- Explorar às vulnerabilidades apontadas e buscar firmwares na internet, criando relatórios consistentes e enviando para a fabricante para pedir boughts.

Rio Maker Space

DEMONSTRAÇÃO PRÁTICA EM CASA

Rio Maker Space

Exemplos para testar em casa:

D-Link DIR-822 (Roteador) — Tem versões antigas vulneráveis, com credenciais hardcoded.

TP-Link WR841N — Tem versões antigas vulneráveis (tipo backdoor telnet).

Câmeras IP — Muitos firmwares de câmeras, Hikvision, costumam ter senhas e serviços ocultos.

AGORA VOCÊ PRATICA!

PERGUNTAS???

Obrigado, por prestarem atenção no meu humano, assim ele compra mais petiscos para mim! Ass: Zephyr

@strongreen
https://www.strongreen.com
contato@strongreen.com
(31) 93618-0448

APOIADORES:

CARIOCAS
COMUNIDADE DE STARTUPS RJ

konzup

Lymatek

GFRJ | UERJ SATS

DFK DIGITAL SOLUTIONS

A Melo
IT Governance & Security Consulting

WOMCY
LATAM Women in Cybersecurity

Código Carioca

Cyber Security Girls

FIAP