Enquanto uns reclamam e choram, outros analisam firmwares...

Agenda

- whoami.
- bla bla bla.
- fim.

REGRA GERAL:

Pouca segurança ou nenhuma (quase sempre).

Para nossa alegria...

<3!

Pré requisitos

Pré requisitos

Praticar:

- Compilar o código na plataforma embarcada e analisar o binário.

Pré requisitos

- Conhecimento sobre arquiteturas de sistemas embarcados.
- Bom conhecimento em linguagens de programação.
- Não acreditar em decompiladores.

Embarcados presentes na nossa vida:

- SOHO.
- Entreprise.
- Estruturas CRÍTICAS.

Pouca segurança ou nenhuma (quase sempre).

Alguns fabricantes de hardware no mundo (tendência de aumentar muito mais...).

Poucos fabricantes nacionais (ou mal e porcamente fabricados), quase sempre tudo comprado aqui é fabricado lá fora.

- O que são embarcados?
- O que são firmwares?
- Quais os desafios?
- Como conquistar os desafios.
- Formatos tipícos de firmware e conteudo.
- Firmware packing.
- Como desempacotar.
- Processo típico de analise.

Firmware

Firmware

O termo firmware foi criado por Ascher Opler em 9167.

Definido por ele como:

“as a glue microcode layer between the CPU instruction set
and the actual hardware”

Firmware

Mas o IEEE Standard Glossary of Software Engineering Terminology, Std 610.12-1990 diz:

The combination of a hardware device and computer instructions and data that reside as read-only software on that device.

Firmware

Notes: (1) This term is sometimes used to refer only to the hardware device or only to the computer instructions or data, but these meanings are deprecated.
Notes: (2) The confusion surrounding this term has led some to suggest that it be avoided altogether.

Classes de dispositivos

Classes de dispositivos

Firmwares são encontrados em todos os lugares hoje em dia.

Carros são controlados por centenas de microcontroladores, máquinas de lavar são programáveis e é claro, controles industriais são automatizados e controlados por uma console central.

Classes de dispositivos

Alguns exemplos de dispositivos e suas classes...

Classes de dispositivos

Networking: Routers, Switches, NAS, VoIP phones.
Alarmes: Alarmes, Cameras, CCTV, DVRs, NVRs.

Classes de dispositivos

Automação industrial: PLCs, Usinas de energia, processos de monitoramento e automação industrial.
Automação residencial: Sensores, Smart Homes, Z-Waves.

Classes de dispositivos

Eletro domésticos: Máquina de lavar, geladeira, fogão.
Entretenimento: TV, DVRs, Receiver, Stereo, Video Games, MP3 Player, Camera, Mobiles.

Classes de dispositivos

Outros: Hds, impressoras, carros, equipamentos médicos e etc...

Arquiteturas

Arquiteturas

Temos dois tipos de embarcados:

Arquiteturas de processadores.
On board buses.

Arquiteturas

Arquiteturas de processadores.

Arquiteturas

Contrariando a relativa uniformidade do mercado de Pcs, dispositivos embarcados são muito diversificados.

Arquiteturas

No segmento do meio para o segmento do alto mercado temos processadores oferecendo opções como virtualizações e altos rates de clocks.

Arquiteturas

Arms são bastante espalhados e a Intel está tentando alcançar com a linha ATOM. Processadores MIPS podem ser encontrados também.

Arquiteturas

No mercado da classe baixa, temos arquiteturas de processadores como Atmel AVR, Intel 8051 e microcontroladores Motorola 6800/6800 com pequenas memórias e frequências de clocks baixas.

Arquiteturas

E algumas arquiteturas mais exóticas como Ambarella, Axis, CRIS e outras...

Arquiteturas

On board buses

Arquiteturas

Os cores dos processadores comunicam-se com outros blocos ou chips em volta dele através de uma variedade de interfaces:

Quase sempre, SPI (Serial Peripheral Interface),
I2C (Inter-IC),
Dallas 1-Wire e UART serial buses.

Arquiteturas

Podem ser encontrados, mas sistemas mais complexos com barramentos mais comuns em arquiteturas de PCs como PCI and PCI Express.

Arquiteturas

E cores ARM pode ser conectados através da interface AMBA
(Advanced Microcontroller Bus Architecture).

Linhas de comunicação comuns

Linhas de comunicação comuns

- Ethernet - RJ 45.
- RS485.
- CAN/FlexRay.
- Bluetooth.
- WIFI.
- Infrared.
- Zigbee.
- GPRS/UMTS.
- USB.

Memórias

Memórias

Diferentes tipos de memórias são usadas para mapear diretamente no espaço de endereços do sistema embarcado.

Temos:

- DRAM.
- SRAM.
- ROM.
- NOR Flash.

DRAM

DRAM

Memória volátil que pode ser acessada para ler/escrever.

Bem veloz, alguns ciclos do processador são necessários para acessar o conteúdo.
O controlador dessa memória, precisa saber o timing da memória antes desse tipo de memória ser usado, o que acontece logo nos primeiros estágios do bootloader.

SRAM

SRAM

Memória volátil que pode ser acessada para ler/escrever.

Muito rápida e pode ler ou escrever sem muito delay que a DRAM, porém é mais cara. Encontrada em pequenas quantidades (tipicamente 1mb) nos dispositivos embarcados.

ROM

ROM

Memória não volátil que pode ser lida.

Tipicamente programada na fábrica e que contém o código inicial, como o bootloader.

NOR Flash

NOR Flash

Memória não volátil que pode ser lida/escrita.

Contrariando as memórias passadas, a leitura pode ser em qualquer offset, mas a escrita tem que tomar local num block, o que é comumente usado para guardar o bootcode.

Storage comuns

Storage comuns

Outras opções para armazenar dados:

- NAND Flash.
- SD Card.
- HD.

NAND Flash

NAND Flash

Tipicamente conectada num barramento como SPI para a CPU e comportamento similar ao NOR Flash.

NAND Flash

Qualquer offset pode ser lido, mas escrita e deleção precisa usar o bloco todo para isso.

Desde que cada bloco pode ser escrito, é comum dar problemas principalmente em sistemas que tentam equilibrar os dados entre as células.

SD Card

SD Card

SD Card ou MMC podem serem usados como block device no Linux.

A conexão para o controlador do sistema principal pode ser feito via USB, SPI etc...

HD

HD

HD pode ser conectado via SATA, SCSI ou PATA.

Como os SD Cards, a conexão atual com a controladora pode ser feitas usando outros barramentos.

Sistemas Operacionais

Sistemas Operacionais

Sistemas embarcados são portados com firmwares de complexidades variada.

Mais complexo é quando possuem sistemas operacionais como o Linux.

Sistemas Operacionais

Dispositivos menos complexos usam sistemas como o VxWorks.

Veremos uma lista dos mais comuns:

Sistemas Operacionais

Linux, o mais comum.
VxWorks (popular sistema RTOS).
Cisco IOS.
Windows CE/NT.
L4.
eCos.
DOS.
Symbian.
JunOS.
Ambarella.
Etc...

Bootloaders.

Bootloaders.

A primeira parte do software que é executado depois do ROM bootloader.

Seu propósito é carregar partes do sistema operacional na memória e trazer o sistema num estado definido para o kernel.

Bootloaders.

Pode ser organizado em um ou dois estágios.

Numa configuração com o segundo estágio, o primeiro estágio apenas sabe como carregar o segundo estágio, enquanto o segundo estágio prove suporte para os arquivos de sistema.

Bootloaders.

Exemplos:

- U-Boot (mais popular).
- RedBoot
- BareBox
- Ubicom.

Bibliotecas e ambientes

Bibliotecas e ambientes

Temos diversos toolchains pré pacotados.

Eles possuem ferramentas de build específicas para o processador (compilador, assembler e etc...).

Bibliotecas e ambientes

Na maioria dos casos, também temos a biblioteca padrão para o nosso alvo e para alguns pacotes open sources, como penembedded.

Bibliotecas e ambientes

Uma lista dos exemplos mais comuns:

busybox + uClibc (provavelmente a combinação mais usada).
buildroot.
openembedded.
crosstool.
crossdev.

cross-channel scripting (XCS)

cross-channel scripting (XCS)

Com o tempo, novas classes de vulnerabilidades foram descobertas e nomeadas de cross-channel scripting (XCS).

Vulnerabilidade não somente restrita a embarcados mas uma vulnerabilidade muito comum em equipamentos.

Ferramentas

Ferramentas

Existem muitos esforços da comunidade de segurança dedicada a engenharia reversa de firmwares e serviços embarcados.

Cada esforço desse tem um objetivo específico.

Ferramentas

Uma lista de exemplos de ferramentas e veremos profundamente suas capacidades:

- binwalk.
- firmware-mod-kit.
- FRAK: Firmware Reverse Analysis Konsole.
- ERESI framework.
- signsrch.
- offzip.
- TrID.
- gpltool/bat.

Ferramentas

- PFS.
- CNU_fpu.
- adrone-tool.
- unYAFFS.
- squash-tools.
- UbiFS.

Ferramentas/binwalk

Ferramenta de analise de firmware desenhada para ajudar na analise, extração e engenharia reversa das imagens dos firmwares e outros binários.

Ferramentas/binwalk

Simples de usar, totalmente scriptável e pode ser extendida via assinaturas custom, regras de extração e módulos de plugins.

Ferramentas/firmware-mod-kit

Coleção de scripts e utilitários para extrair e reconstruir imagens de firmware baseados no Linux.

O kit permite descontrução e reconstrução fácil de imagens de firmware

Ferramentas/FRAK: Firmware Reverse Analysis Konsole

Sem notícias desde BlackHat 12 US.

FRAK é open source mas sem muita informação recente.

Ferramentas/ERESI framework

Framework multi arquitetura de analise de binário com uma linguagem específica preparada para engenharia reversa e manipulação de programas.

Ferramentas/signsrch

Ferramenta para pesquisar sobre assinaturas dentro de arquivos, extremamente importante na tarefa de engenharia reversa.

Boa para ter uma idéia inicial sobre qual algoritmo de criptografia/compressão é usado pelo protocolo proprietário ou arquivo.

Ferramentas/signsrch

Pode reconhecer diversas compressões, multimídias e algoritmos de criptografia e muitas strings e código anti debugging, que pode ser adicionado devido a ele ser todo baseado em assinaturas em texto lidos na runtime e fáceis de modificar.

Ferramentas/offzip

Ferramenta muito boa para descompactar dados de zips (zlib, gzip,deflate, etc...) contendo qualquer tipo de arquivo inclusive raw, packets, arquivos zips, executáveis e tudo mais...

Ferramentas/offzip

É necessário explicar o offset onde o zip começa ou usando -S para procurar o bloco de zip no arquivo passado.
Também tem opção para extração de todos os zips dos blocos que foram encontrados ou dumpando eles todos pra forma original.

Ferramentas/TrID

Usada para identificar tipos de arquivos das assinaturas dos binários.

TrID não tem regras fixas, é flexível e pode ser treinado para reconhecer novos formatos de forma simples e rápida.

Ferramentas/gpltool/bat

O BAT (Binary Analisys Tool) previamente gpltool, é uma ferramenta fácil e simples para se olhar por dentro dos binários, encontrar dados e etc..

Ferramentas/PFS

PFS é usado em routers como Benq ESG 103, NDC NWH8018 e provavelmente outros dispositivos.

Ferramentas/CNU fpu

Ferramenta de pack/unpack para Cisco IP Phones firmware (7941, 7961, 7911-12 e outros baseados no CNU File Archive 3.0 format).

Ferramentas/ardrone-tool

Ferramentas de desenvolvimento para AR Drone, permitindo criar e customizar kernels do Linux.

Ferramentas/UnYAFFS

UnYAFFS é usado para extrair arquivos de uma imagem de sistema de arquivos yaffs.

Ferramentas/squash-tools

Ferramentas para lidar com file system.

Formatos de firmwares

Formatos de firmwares

Dependendo da complexidade do firmware, podemos encontrar niveis diferentes de packing e objetos diferentes dentro dos arquivos.

Formatos de firmwares

- Completo (full-OS/kernel + bootloader + libs + apps).
- Integrado (apps + OS-as-a-lib).
- Update parcial (apps ou libs ou fontes ou suporte).

Formatos de firmwares\Completo

- Completo (full-OS/kernel + bootloader + libs + apps).

Tipicamente um firmware que usa Linux ou Windows e carrega o sistema completo.
As aplicações vão rodar em User Mode, através de módulos e drivers do kernel customizados.

Formatos de firmwares\Integrado

- Integrado (apps + OS-as-a-lib).

Firmware com pequeno sistema operacional proprietário ou nenhum.
A aplicação vai rodar no mesmo privilégio que o kernel.

Formatos de firmwares\Update parcial

- Update parcial (apps ou libs ou fontes ou suporte).

A imagem do firmware não vai conter todos os arquivos de um sistema completo, mas apenas nas atualizações dos arquivos.

Formatos de firmwares

Nos firmwares, vamos encontrar os seguintes objetos típicos enquanto descompactamos os firmwares:

Formatos de firmwares

- Bootloader (1st/2nd stage).
- Kernel.
- Imagens de sistema de arquivos.
- Binários user land.
- Fontes e arquivos de suporte.
- Web-server/web-interface.

Formatos de firmwares

Esses objetos podem ser agrupados e armazenados nos seguintes sistemas de arquivos ou imagens:

Formatos de firmwares

- Arquivos (CPIO/Ar/Tar/GZip/BZip/LZxxx/RPM).
- Sistemas de arquivos (YAFFS, JFFS2, extNfs).
- Formatos de binários (SREC, iHEX, ELF).
- Híbridos.

Ferramentas e formatos

Ferramentas e formatos

- Ar (ar x arquivo, para extrair).
- YAFFS2 (ferramentas do projeto yaffs2utils).
- JFFS2 (BAT usa um python wrapper sobre o utilitario js2dump que faz parte do mtdtools.

Ferramentas e formatos

- CramFS (firmware-mod-kit tem uma ferramenta chamada uncramfs para extrair).
- ROMFS (romfschk).
- xFAT (montado como dispositivo no Linux).
- NTFS (montado como dispositivo no Linux).

Ferramentas e formatos

- ext2fs/ext3fs/ext4fs (montado como dispositivo no Linux).

Ferramentas e formatos

iHEX – converta assim:

objcopy -I ihex -O elf32-little <input> <output>
objcopy -I ihex-O binary <input> <output>

Ferramentas e formatos

- SREC/S19 – converta assim:

objcopy -I srec -O elf32-little <input> <output>
objcopy -I srec-O binary <input> <output>

Ferramentas e formatos

- CPIO/Ar/Tar/GZip/BZip/LZxxx/RPM:

Sua distribuição deve ter ferramentas para lidar com esses tipos de arquivos.

Quais os desafios?

Quais os desafios?

- Formatos non-standard.
- Pedaços encriptados.

- Canais de updates non-standard:
Firmwares vem e vão, vendedores mudam rapidamente de sites/ftp de suporte.

- Procedimentos de atualizações non-stantard:
Impressoras com updates via hacks de PJL (printer job language) específicos.

Atualizando o firmware

Atualizando o firmware

Firmware update built-in:
- Web-based upload
- Socket-based upload
- USB-based upload

Atualizando o firmware

- Função de update no bootloader.
- USB-boot recovery.

- Resgate de partição:

Novo firmware é escrito em espaco livre e a checagem de integridade antes de ser ativado. Velho firmware não é sobrescrito se o novo não estiver ativo.

Atualizando o firmware

- JTAG/ISP/Parallel programing.

Armadilhas de atualização

Armadilhas de atualização

- Protocolos de update com autenticação não mútua.
- Pacotes não assinados.
- Assinaturas não verificadas.
- Verificação das assinaturas incorretas/inconsistentes;
- Vazamento de chaves de assinaturas.

Por que os firmwares estão desatualizados?

Visão do vendedor

Visão do vendedor

- Lucro e rápida colocação no mercado primeiro.

Segurança? Nah!

QUE SE FODA!

Visão do vendedor

- Diversas plataformas geram custos de compilação e manutenção.
- Segurança leva muito tempo e esforço (aparelhos médicos dependem de muito tempo para serem aprovados...).

Visão do comprador

Visão do comprador

- “Se funciona, não toque!”.
- Baixo nível técnico.
- Grande esforço para instalar firmwares novos.

=> PROBABILIDADE GIGANTE DE DAR MERDA NA ATUALIZAÇÃO DO FIRMWARE NA MÃO DO NOOBA! <=

Visão do comprador

“Onde eu coloco esse CD de upgrade na impressora?

Ela não tem leitor e nem nada....”

Iniciando a análise

Iniciando a análise

- Pegue o firmware.
- Reconhecimento.
- Unpacking.
- Engenharia (veja code.google.com e sourceforge e etc...).
- Localize o alvo da análise.

Decompile/compile/analise/fuzz/pentesting/pwnz/0dayz!

Pegue o firmware

Pegue o firmware

Não é fácil as vezes (quase sempre...).
As vezes é uma merda mesmo pra achar mal e porcamente o firmware.

Pegue o firmware

- Pode estar em CD/DVD.
- Download de algum site HTTP/FTP.
- Download Bootdisk / USB / CD boot images, extraia usando Winrar (Windows) ou mount (Linux).

Pegue o firmware

- Registro no site necessário (As vezes Fabricantes Satanás passam dos limites...).
- Google Dorks.
- FTP index sites.
- Traços do wireshark (sniffing do update via port mirror).
- Dump da memória do dispositivo.

Pegue o firmware

- .bin / .hex / .s19 / .mot / .rom / .raw
- Non bin (hex2bin).

Reconhecimento

Reconhecimento

- Strings no firmware.
- Fuzzy match.
- Encontre e leia os specs e datasheets dos dispositivos.

Ninguém prestou atenção no último slide!

RISOS!

Unpacking

Unpacking

Binwalk:

- Fácil de criar / modificar assinaturas.
- Detecção de falso positivo.
- Automático, extração recursiva.
- Analise de entropia/heurística
- Rápido.

Emulação de Firmware

Emulação de Firmware

- Imagem do kernel com um set de módulos do kernel.
- QEMU compilado na arquitetura do dispositivo embarcado (ARM, MIPS, MSP430, MN103 e etc...).
- Firmware – splitado em partes pequenas para não quebrar o QEMU.

Debugando

Debugando

- JTAG.
- Software debugger (gdb ou ARM Angel Debug Monitor).
- OS debug (KDB/KGDB).

Desenvolvimento

Desenvolvimento

- GCC/Binutils toolchain.
- Cross-compilers.
- Compilador proprietário.
- Construa a imagem.

E VOCÊ?

E VOCÊ?

TEM RECLAMADO BASTANTE?

E VOCÊ?

TEM CHORADO BASTANTE?

E VOCÊ?

...

VOCÊ É UM...

VOCÊ É UM...

MERDA!

VENHA SER PARTES DOS OUTROS!

Os outros

VENHA SER PARTE DA GALERINHA DO MAL!

Os outros

MAS NÃO ROUBE BUGS DO SEU AMIGUINHO
PRA DAR TALK SOBRE ISSO LÁ FORA...

Os outros

PARE DE RECLAMAR E CHORAR E VAMOS DESCOBRIR 0DAYZ ANALISANDO FIRMWARES!

Os outros

COMO FAS?

DEMO

TP LINK 3020.

TP LINK 3020


TP-LINK
150Mbps TL-MR3020
WPS/RESET
3G/4G
WISP
AP
WAN/LAN
POWER
3G/4G USB

TP LINK 3020

Download:

TL-MR3020_V1_150921.zip

TP LINK 3020

Unzip:

unzip TL-MR3020_V1_150921

TP LINK 3020

mr3020nv1_en_3_17_2_up_boot(150921).bin

TP LINK 3020

Binwalk:

```
# binwalk -e
mr3020nv1_en_3_17_2_up_boot(150921).bin
```

TP LINK 3020

Navegar:

cd _mr3020nv1_en_3_17_2_up_boot(150921).bin.extracted

TP LINK 3020

```
# cd squashfs-root/
```

TP LINK 3020

```
# cd /etc/rc.d/
```

TP LINK 3020

cat rcS

TP LINK 3020

```
!/bin/sh

# This script runs when init it run during the boot process.
# Mounts everything in the fstab

mount -a
#mount -o remount +w /

#
# Mount the RAM filesystem to /tmp
#

mount -t ramfs -n none /tmp
mount -t ramfs -n none /var
mount -t usbfs none /proc/bus/usb

export PATH=$PATH:/etc/ath

#insmod /lib/modules/2.6.15/net/ag7100_mod.ko
#insmod /lib/modules/2.6.15/net/ag7240_mod.ko
```

TP LINK 3020

```
#
# Set lo eth1 up
#
ifconfig lo 127.0.0.1 up
#ifconfig eth1 up

#
# insert netfilter/iptables modules
#
```

/etc/rc.d/rc.modules

TP LINK 3020

```
#
# Start Our Router Program
#

/usr/bin/httpd &

# [wukan start] In BETA version,we start telnetd for debugging.
if [ -x /usr/sbin/telnetd ]; then
/usr/sbin/telnetd &
fi
# [wukan end]

echo 524288 > /proc/sys/net/ipv4/ipfrag_high_thresh
```

TP LINK 3020

Olha só isso...

TP LINK 3020

=> /usr/bin/httpd &

TP LINK 3020

Ele só sobe o daemon httpd....

Mas vamos lá verificar!

TP LINK 3020

```
# file httpd
httpd: ELF 32-bit MSB executable, MIPS, MIPS32
rel2 version 1 (SYSV), dynamically linked,
interpreter /lib/ld-uClibc.so.0, stripped
```

TP LINK 3020

```
# readelf -h httpd  ou mips-linux-gnu-readelf -h httpd
ELF Header:
  Magic:   7f 45 4c 46 01 02 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, big endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           MIPS R3000
  Version:                           0x1
  Entry point address:               0x41b890
  Start of program headers:          52 (bytes into file)
  Start of section headers:          1558496 (bytes into file)
```

TP LINK 3020

Number of program headers: 9
Size of section headers: 40 (bytes)
Number of section headers: 31
Section header string table index: 30

TP LINK 3020

```
# mips-linux-gnu-objdump -x httpd | grep lib
  NEEDED               libpthread.so.0
  NEEDED               libc.so.0
  NEEDED               librt.so.0
  NEEDED               libmsglog.so
  NEEDED               libutil.so.0
  NEEDED               libwpa_ctrl.so
  NEEDED               libgcc_s.so.1
  required from libgcc_s.so.1:
```

TP LINK 3020

```
# mipsel-linux-gnu-objdump -f httpd

httpd:     file format elf32-tradbigmips
architecture: mips:isa32r2, flags 0x00000112:
EXEC_P, HAS_SYMS, D_PAGED
start address 0x0041b890
```

TP LINK 3020

```
# arm-linux-gnueabi-objdump -d -m arm httpd
```

TP LINK 3020

httpd: file format elf32-big

Disassembly of section .init:

0041b808 <_init>:

```
 41b808:  3c1c0015  ldccc   0, cr0, [ip], {21}
 41b80c:  279c7308  ldrcs   r7, [ip, r8, lsl #6]
 41b810:  0399e021  orrseq  lr, r9, #33   ; 0x21
 41b814:  27bdffe0  ldrcs   pc, [sp, r0, ror #31]!
 41b818:  afbc0010  svcge   0x00bc0010
 41b81c:  afbf001c  svcge   0x00bf001c
 41b820:  afbc0018  svcge   0x00bc0018
```

Etc...

TP LINK 3020

Vamos descobrir a backdoor?

TP LINK 3020

```
# strings httpd | grep "Authorization"
```

TP LINK 3020

httpAuthorizationGet
httpMimeAuthorization
<html><body><script type="text/javascript">window.parent.document.cookie="Authorization=;path=/";window.parent.document.cookie="TPLoginTimes=0;path=/";window.parent.location.href="/";</script>
Authorization: Basic
Authorization: Basic YWRtaW46aWxvdmV0cGxpbms=
Authorization: Basic YWRtaW46aWxvdmV0cGxpbms=
Authorization: Basic YWRtaW46aWxvdmV0cGxpbms=
Authorization

TP LINK 3020

Authorization: Basic YWRtaW46aWxvdmV0cGxpbms=

????

TP LINK 3020

```
# echo "YWRtaW46aWxvdmV0cGxpbms=" |
base64 -d
```

Resultado:

admin:ilovetplink

TP LINK 3020

admin:ilovetplink

<3

<3<3

Yealink SIP-T22P

Yealink SIP-T22P

Yealink SIP-T22P

Download:

7.72.0.80.zip

Yealink SIP-T22P

Unzip:

unzip 7.72.0.80.zip

Yealink SIP-T22P

Binwalk:

```
# binwalk -e 7.72.0.80.rom
```

Yealink SIP-T22P

Navegar:

cd _7.72.0.80.rom.extracted/

Yealink SIP-T22P

```
# cd squashfs-root/
```

Yealink SIP-T22P

cd /etc/

Yealink SIP-T22P

```
# cat passwd
root:x:0:0:Root,,,:/:/bin/sh
toor:x:0:0:Root,,,:/:/bin/sh
```

Yealink SIP-T22P

cat shadow

root:
1.jKlhz0B$/Nmgj0kIrsZk0nYc1BLUR/:11876:0:99999:7:::
toor:
$1$6sa7xxqo$eV3t7Nb1tPqjOWT1s3/ks1:11876:0:99999:7:::

Yealink SIP-T22P

root:
1.jKlhz0B$/Nmgj0kIrsZk0nYc1BLUR/:11876:0:99999:7:::

Yealink SIP-T22P

toor:
$1$6sa7xxqo$eV3t7Nb1tPqjOWT1s3/ks1:11876:
0:99999:7:::

Pararam de chorar e reclamar?

:)

Agora volte pra sua casa e saia debugando tudo,
até mesmo seu cachorro!

Oscar Marques
@f117usbr
oscarbm@gmail.com