PENETRATION TEST REPORT

for

ASL19

V1.0
Amsterdam
June 6th, 2018

Document Properties

Client	ASL19
Title	PENETRATION TEST REPORT
Targets	ASL19 Twitter Proxy ASL19 Telegram Proxy ASL19 HAProxy
Version	[redacted]
Pentesters	
Author	
Reviewed by	
Approved by	

Version control

Version	Date	Author	Description
0.1	April 20th, 2018	[redacted]	Initial draft
0.2	June 5th, 2018		Main report
1.0	June 6th, 2018		Final version

Contact

For more information about this Document and its contents please contact Radically Open Security B.V.

Name	[redacted]
Address	
Phone	
Email	

Table of Contents

1 Executive Summary

1.1 Introduction

Between April 25, 2018 and May 25, 2018, Radically Open Security B.V. carried out a penetration test for ASL19

This report contains our findings as well as detailed explanations of exactly how ROS performed the penetration test.

1.2 Scope of work

The scope of the penetration test was limited to the following target:

- ASL19 Twitter Proxy
- ASL19 Telegram Proxy
- ASL19 HAProxy

1.3 Project objectives

Evaluate and test the configuration of HAProxy, Dante, and Squid proxy services (used for proxying Telegram, Twitter and TCP connections) for security, privacy, and practical concerns.

1.4 Timeline

The Security Audit took place between April 25th and May 25th, 2018.

1.5 Results In A Nutshell

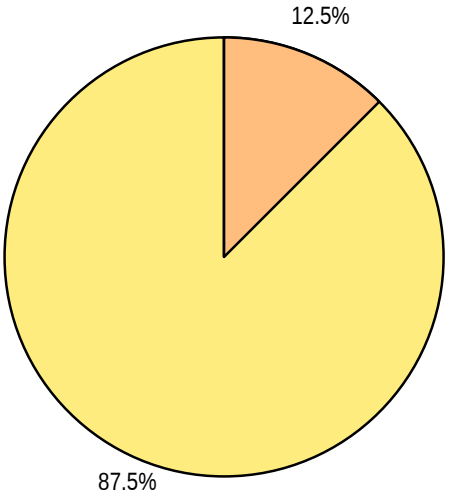
A number of configuration problems were found, raising security, privacy, and practical issues.

None of the findings were especially severe. ASL19 was very responsive and addressed all of these concerns immediately, and these changes were verified in a retest.

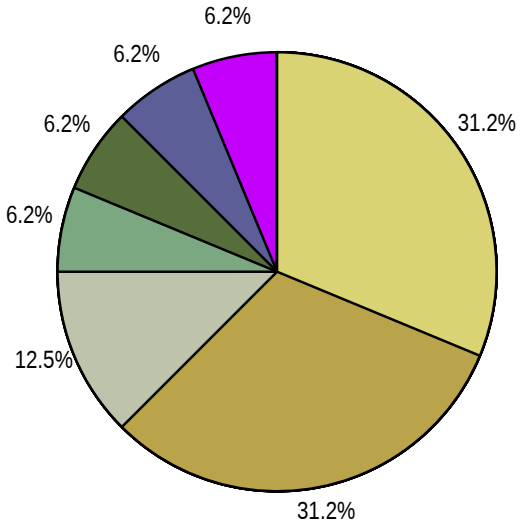
1.6 Summary of Findings

ID	Type	Description	Threat level
ASL-003	Missing encryption	HAProxy has an HTTP statistics UI; This interface is not configured to use TLS encryption.	Moderate
ASL-009	Open ports	There are a number of unnecessary firewall rules, and unneeded open ports.	Moderate
ASL-001	Outdated software	There are three supported major releases of HAProxy; The server is using the oldest of them.	Low
ASL-002	Outdated software	The latest release of HAProxy 1.6 is 1.6.14; the server is running 1.6.3, from 2015.	Low
ASL-004	Configuration	Connection count limits do not allow for the connections needed by the stats service.	Low
ASL-005	Configuration	Connection count limits are set to unrealistically high values.	Low
ASL-006	Configuration	The high connection count limits set in HAProxy will not be obtainable if the overall system's limits are not set to similarly high values.	Low
ASL-007	Incorrect permissions	Permissions on the chroot that HAProxy runs in are not set up as recommended in HAProxy docs	Low
ASL-008	Configuration	The stats service is set to run in 'admin' mode, which is excessive for typical monitoring purposes.	Low
ASL-010	Information disclosure	The Dante server is configured to log iooperation. When this setting is enabled both source and destination addresses are written to the log, which could violate the users privacy.	Low
ASL-011	Exposed credentials	The file /etc/squid/squid.user contains an unused encrypted user/password combination for the user paskoocheh.	Low
ASL-012	Information disclosure	Squid is configured to cache data if possible. Payloads from unencrypted client requests might be persisted to disk that could contain private user information.	Low
ASL-013	Configuration	The Ubuntu server does not install security updates automatically.	Low
ASL-014	Information disclosure	Connecting client IP addresses appear in the Dante log file /var/log/danted.log.	Low
ASL-015	Information disclosure	Proxy client IP addresses appear in the Squid server log file /var/log/squid/access.log.	Low
ASL-016	Information disclosure	Proxy client IP addresses appear in the HAProxy server log file /var/log/haproxy.log.	Low

1.6.1 Findings by Threat Level



1.6.2 Findings by Type



1.7 Summary of Recommendations

ID	Type	Recommendation
ASL-001	Outdated software	• Update to latest version of HAProxy 1.8
ASL-002	Outdated software	• Update to latest version of HAProxy 1.6
ASL-003	Missing encryption	• Either enable encryption on the stats interface, or disable it.

ASL-004	Configuration	• Leave some breathing room in the connection count limits to ensure that the stats service is always available.
ASL-005	Configuration	• Reduce connection count limits to realistic expected values.
ASL-006	Configuration	• Set sysctl values high enough accommodate the values HAProxy is asking for.
ASL-007	Incorrect permissions	• Set permissions and ownership as recommended.
ASL-008	Configuration	• Configure the stats service with minimum necessary permissions.
ASL-009	Open ports	• Remove unneeded rules • Close unused ports
ASL-010	Information disclosure	• Disable the `iooperation` option.
ASL-011	Exposed credentials	• Remove the password file entry
ASL-012	Information disclosure	• Disable caching
ASL-013	Configuration	• Install the `unattended-upgrades` package and configure it to install security patches automatically.
ASL-014	Information disclosure	• Confirm whether client IPs are being logged.
ASL-015	Information disclosure	• Disable logging of client IP addresses.
ASL-016	Information disclosure	• Disable logging of client IP addresses in the HAProxy log. • Configure frequent clearing/rotation of error logs.

2 Methodology

2.1 Planning

Our general approach during this penetration test was as follows:

1. **Reconnaissance**
   We attempted to gather as much information as possible about the target. Reconnaissance can take two forms: active and passive. A passive attack is always the best starting point as this would normally defeat intrusion detection systems and other forms of protection, etc., afforded to the network. This would usually involve trying to discover publicly available information by utilizing a web browser and visiting newsgroups etc. An active form would be more intrusive and may show up in audit logs and may take the form of a social engineering type of attack.
2. **Enumeration**
   We used varied operating system fingerprinting tools to determine what hosts are alive on the network and more importantly what services and operating systems they are running. Research into these services would be carried out to tailor the test to the discovered services.
3. **Scanning**
   Through the use of vulnerability scanners, all discovered hosts would be tested for vulnerabilities. The result would be analyzed to determine if there are any vulnerabilities that could be exploited to gain access to a target host on a network.
4. **Obtaining Access**
   Through the use of published exploits or weaknesses found in applications, operating system and services access would then be attempted. This may be done surreptitiously or by more brute force methods.

2.2 Risk Classification

Throughout the document, vulnerabilities or risks are labeled and categorized as:

- **Extreme**
  Extreme risk of security controls being compromised with the possibility of catastrophic financial/ reputational losses occurring as a result.
- **High**
  High risk of security controls being compromised with the potential for significant financial/ reputational losses occurring as a result.
- **Elevated**
  Elevated risk of security controls being compromised with the potential for material financial/ reputational losses occurring as a result.
- **Moderate**

Moderate risk of security controls being compromised with the potential for limited financial/ reputational losses occurring as a result.

- **Low**
  Low risk of security controls being compromised with measurable negative impacts as a result.

Please note that this risk rating system was taken from the Penetration Testing Execution Standard (PTES). For more information, see: http://www.pentest-standard.org/index.php/Reporting.

3 Reconnaissance and Fingerprinting

Through automated scans we were able to gain the following information about the software and infrastructure. Detailed scan output can be found in the sections below.

3.1 Automated Scans

As part of our active reconnaissance we used the following automated scans:

- nmap – http://nmap.org

4 Pentest Technical Summary

4.1 Findings

We identified the following issues, most of which were fixed more or less immediately by ASL19 during the test.

4.1.1 ASL-001 — Older HAProxy major version in use

Vulnerability ID: ASL-001

Vulnerability type: Outdated software

Threat level: Low

Description:

There are three supported major releases of HAProxy; The server is using the oldest of them.

Technical description:

HAProxy maintains several concurrently supported versions: 1.6, 1.7, and 1.8; the server is running 1.6. There are no recorded vulnerabilities in intervening versions, but there are numerous bug fixes and enhancements in later versions.

This finding has been addressed by updating the server to the latest version of HAProxy 1.8.

Impact:

None known

Recommendation:

- Update to latest version of HAProxy 1.8

4.1.2 ASL-002 — Outdated Version of Haproxy Installed

Vulnerability ID: ASL-002

Vulnerability type: Outdated software

Threat level: Low

Description:

The latest release of HAProxy 1.6 is 1.6.14; the server is running 1.6.3, from 2015.

Technical description:

HAProxy maintains several concurrently supported versions; ASL is running 1.6, the oldest of the supported versions, 1.6 (see http://git.haproxy.org/?p=haproxy-1.6.git). There are no recorded vulnerabilities in intervening versions, but there are numerous bug fixes.

This finding has been addressed by fixing ASL-001 (page 9).

Impact:

None known

Recommendation:

- Update to latest version of HAProxy 1.6

4.1.3 ASL-003 — HAProxy Stats Interface Does Not Use Encryption

Vulnerability ID: ASL-003

Vulnerability type: Missing encryption

Threat level: Moderate

Description:

HAProxy has an HTTP statistics UI; This interface is not configured to use TLS encryption.

Technical description:

The stats interface is configured to be available at http://159.65.201.126:9999/stats. This interface is not configured to use TLS encryption and thus may leak information, or expose remote admin functions to anyone sniffing traffic on that port.

That said, the login details configured in the config file don't work for me, so it may be broken. Nonetheless, it should not be available over HTTP anyway.

This has been addressed by disabling the stats interface altogether.

Impact:

The stats traffic is exposed, which may allow attackers to submit requests and intercept data that may expose information useful for further attacks.

Recommendation:

- Either enable encryption on the stats interface, or disable it.

4.1.4 ASL-004 — It's Possible for the Stats Service to Be Starved of Connections

Vulnerability ID: ASL-004

Vulnerability type: Configuration

Threat level: Low

Description:

Connection count limits do not allow for the connections needed by the stats service.

Technical description:

HAProxy's global maxconn is set to 2000000, and so is the squid-proxy listener. Because these are the same, it's possible (though only in an academic sense!) for the stats listener to be starved of connections. The squid-proxy maxconn value should be decreased slightly, or the global maxconn should be increased slightly (the stats service is not expected to serve large amounts) so that there is space in the connection counts for both listeners.

The connection counts have been reduced, and the stats service is no longer running, so this finding has been resolved.

Impact:

Under heavy load, the HAProxy stats service may become unavailable.

Recommendation:

- Leave some breathing room in the connection count limits to ensure that the stats service is always available.

4.1.5 ASL-005 — Use Realistic Maxconn Values

Vulnerability ID: ASL-005

Vulnerability type: Configuration

Threat level: Low

Description:

Connection count limits are set to unrealistically high values.

Technical description:

It's reasonable for the HAProxy maxconn values (both global and per listener) to be (much) larger than the maxconn for the back-end server, as it is very efficient at queuing connections. Maxconn for each back-end should be set to a realistic value for the traffic it's likely to be able to cope with.

The maxconn of 2000000 (used for both global and squid-server) is a lot of simultaneous connections to support, and is most likely unrealistic for the intended use; reduce it to a more reasonable figure. If a 1-way connection consumes 4k of memory, 2000000 2-way connections will require 16G - four times what's on this test server.

This is slightly academic - If the load balancer is serving traffic volume well below its capabilities, these limits will never come into play and so it doesn't matter if they are set too high - but should load ever increase to a point where back-ends are having trouble, the values become critical, so it's best to use realistic estimates rather than just setting them very high.

Confirmed that the connection limits have been lowered to more realistic levels.

Impact:

If traffic increases enormously, connection count limits that are set too high will cause the server to run out of resources (effectively a DoS) instead of degrading gracefully.

Recommendation:

- Reduce connection count limits to realistic expected values.

4.1.6 ASL-006 — Amend Sysctl Values to Support Limits Set in HAProxy Config

Vulnerability ID: ASL-006

Vulnerability type: Configuration

Threat level: Low

Description:

The high connection count limits set in HAProxy will not be obtainable if the overall system's limits are not set to similarly high values.

Technical description:

HAProxy has some very high connection count limits set, however, they are subject to whatever resources and limits are made available by the kernel, which is controlled from sysctl. For example, `net.core.somaxconn` is at its default value of 128, means that the attainable connection count is well below what HAProxy is configured for. Tuning sysctl parameters is a very complex area, but there are numerous articles that may help, for example https://klaver.it/linux/sysctl.conf and https://stackoverflow.com/questions/410616/increasing-the-maximum-number-of-tcp-ip-connections-in-linux.

Aside from the HAProxy limits sounding too high (ASL-005 (page 12)), `max_syn_backlog` should be set to about twice `somaxconn`, which should also be the same as `netdev_max_backlog`, but overall,

they need to match the connection limits HAProxy asks for. There's another nice discussion here: http://engineering.chartbeat.com/2014/01/02/part-1-lessons-learned-tuning-tcp-and-nginx-in-ec2/ .

I can't give accurate suggestions as it depends on the traffic you're expecting, and getting it right will involve longer term monitoring of what's actually happening.

Confirmed that sysctl values have been set appropriately.

Impact:

HAProxy connection count limits will never be reached if the kernel settings are smaller, resulting in connections timing out under high load.

Recommendation:

- Set sysctl values high enough accommodate the values HAProxy is asking for.

4.1.7 ASL-007 — Chroot Permissions Not Set as Recommended

Vulnerability ID: ASL-007

Vulnerability type: Incorrect permissions

Threat level: Low

Description:

Permissions on the chroot that HAProxy runs in are not set up as recommended in HAProxy docs

Technical description:

Permissions on the chroot that haproxy runs in are not set up as recommended in haproxy docs, which say:

A safe configuration will have :

- a chroot statement pointing to an empty location without any access permissions. This can be prepared this way on the UNIX command line :

  ```
  # mkdir /var/empty && chmod 0 /var/empty || echo "Failed"
  ```

  and referenced like this in the HAProxy configuration's global section :

  ```
  chroot /var/empty
  ```

- both a uid/user and gid/group statements in the global section :

  ```
  user haproxy
  group haproxy
  ```

- a stats socket whose mode, uid and gid are set to match the user and/or

group allowed to access the CLI so that nobody may access it :

```
stats socket /var/run/haproxy.stat uid hatop gid hatop mode 600
```

and also:

It is important to ensure that `<jail_dir>` is both and unwritable to anyone.

The current configuration uses a chroot in `/var/lib/haproxy`, and its permissions are set to `0755`, which is considerably looser than recommended. I experimented with configuring it as recommended and it still runs as it should.

Confirmed that these changes have been made.

Impact:

There is a distant possibility that incorrect permissions on a chroot folder could allow the chrooted process to escape into the global file system.

Recommendation:

- Set permissions and ownership as recommended.

4.1.8 ASL-008 — Stats Service Has Unnecessary Privileges

Vulnerability ID: ASL-008

Vulnerability type: Configuration

Threat level: Low

Description:

The stats service is set to run in 'admin' mode, which is excessive for typical monitoring purposes.

Technical description:

The HAProxy stats socket (not the listener) is configured like this:

```
stats socket /run/haproxy/admin.sock mode 660 level admin
```

`level admin` (see docs) is unnecessary for simply monitoring the status of HAProxy (e.g. from munin). `level operator` (the default) is probably sufficient, and the lowest `user` level may be enough.

Ownership of the socket is not set explicitly, so it will inherit the `haproxy:haproxy` of the daemon itself. If possible, run it as a different unprivileged user with the `user`/`uid` and `group`/`gid` directives. Permissions are set to `0660`, looser than the default `0600` - this should be set appropriately, depending on which user and group need access to it.

Confirmed that this configuration has been hardened, though it is no longer needed since the Stats service has been disabled.

Impact:

Excessive privileges might allow a process to access or affect data or other processes that it should not be able to.

Recommendation:

- Configure the stats service with minimum necessary permissions.

4.1.9 ASL-009 — Excess Firewall Rules, Too Many Open Ports

Vulnerability ID: ASL-009

Vulnerability type: Open ports

Threat level: Moderate

Description:

There are a number of unnecessary firewall rules, and unneeded open ports.

Technical description:

This is somewhat off-topic, but it's worth mentioning.

There are a set of firewall rules that look like something out of a WAF, and labelled "RH-Firewall-1", which is a rule set from RedHat linux, which this server is not running (though they will still work). They are mostly rules that check for the presence of certain strings in the traffic - however, they would only apply if we were handling unencrypted traffic, which is not the case.

```
# iptables -S
-P INPUT DROP
-P FORWARD DROP
-P OUTPUT ACCEPT
-N f2b-ssh
-N f2b-sshd
-A INPUT -p tcp -m multiport --dports 22 -j f2b-ssh
-A INPUT -p tcp -m multiport --dports 22 -j f2b-sshd
-A INPUT -m set --match-set blacklist src -j DROP
-A INPUT -i lo -j ACCEPT
-A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -m tcp --dport 22 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 13133 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 443 -j ACCEPT
-A INPUT -s 188.166.12.61/32 -p tcp -m tcp --dport 59490 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 9999 -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -m limit --limit 3/min -j LOG --log-prefix "iptables_INPUT_denied: "
```

```
-A INPUT -j REJECT --reject-with icmp-port-unreachable
-A FORWARD -m limit --limit 3/min -j LOG --log-prefix "iptables_FORWARD_denied: "
-A OUTPUT -p tcp -m tcp --dport 22 -j ACCEPT
-A OUTPUT -p tcp -m tcp --dport 53 -j ACCEPT
-A OUTPUT -p udp -m udp --dport 53 -j ACCEPT
-A OUTPUT -p tcp -m tcp --sport 13133 -j ACCEPT
-A OUTPUT -m limit --limit 3/min -j LOG --log-prefix "iptables_OUTPUT_denied: "
```

The firewall also allows traffic on ports 25 and 587 - but the server is not running an inbound mail server, and also allows port 53 TCP and UDP access for DNS queries, which doesn't seem appropriate either. It also allows ports 80 and 443, though the server does not provide http or https services. There is some other traffic allowed globally on UDP ports 33434:33523 - I don't know what that might be for - keepalived?

I suspect most of these rules should be removed; they will probably be spotted during the pentest anyway. Since it's running only a proxy service, the only open ports I can see being appropriate are 22 (ssh), 13133 (the proxy) and 9999 (haproxy stats).

Ports 443 (https) and 9999 (haproxy stats, disabled in #3) are still open. I don't know what that special case for port 59490 is for.

iptable's config file was cleaned and replaced.

Extra SSH rules are applied via Cloud provider's FW feature.

Confirmed that the firewall rule set has been cleaned up, and only necessary ports are open.

Impact:

Unnecessary open ports increase attack surface, and may give an attacker more opportunities for compromise.

Recommendation:

- Remove unneeded rules
- Close unused ports

4.1.10 ASL-010 — Dante - Destination Address Logging

Vulnerability ID: ASL-010

Vulnerability type: Information disclosure

Threat level: Low

Description:

The Dante server is configured to log `iooperation`. When this setting is enabled both source and destination addresses are written to the log, which could violate the users privacy.

Technical description:

Dante does not provide a configuration option to log only source addresses (which have no effect on the users privacy, as described in #17), so disabling this setting entirely is recommended.

An excerpt from the Dante source code `dante/lib/log.c`:

```
case OPERATION_IO:
if (rule->log.data || rule->log.iooperation) {
  if (rule->log.data && datalen != 0) {
     /* ... Disabled in the configuration provided by ASL19 */
  }
  else {
     DO_BUILD(srcdst_str, dologdstinfo);
     snprintf(buf, buflen,
              "-: %s (%lu)", srcdst_str, (unsigned long)datalen);
  }
}

...

#define DO_BUILD(srcdst_str, dst_too)
do {
   /* ... Only source address related */
   if (dst_too) {
      char dststr[MAX_IOLOGADDR];

      build_addrstr_dst(dst != NULL && dst->local_isset ?
                           &dst->local : NULL,
                        todst_proxy != NULL && todst_proxy->local_isset ?
                           &todst_proxy->local : NULL,
                        todst_proxy != NULL && todst_proxy->peer_isset  ?
                           &todst_proxy->peer  : NULL,
                        dst != NULL && dst->peer_isset ?
                           &dst->peer : NULL,
                        dst != NULL && dst->auth_isset ?
                           &dst->auth : NULL,
                        todst_proxy != NULL && todst_proxy->auth_isset ?
                           &todst_proxy->auth  : NULL,
                        GET_HOSTIDV(dst),
                        GET_HOSTIDC(dst),
                        dststr,
                        sizeof(dststr));

      snprintf(srcdst_str, sizeof(srcdst_str), "%s -> %s", srcstr, dststr);
   }
} while (/* CONSTCOND */ 0)
```

In this code snippet we can see that the destination address is included in the log.

Confirmed that this has no privacy impact; all the destination addresses belong to Telegram, not end-users.

Impact:

Excessive logging of user data increases the opportunity for data loss or exposure.

Recommendation:

- Disable the `iooperation` option.

4.1.11 ASL-011 — Squid - User Password

Vulnerability ID: ASL-011

Vulnerability type: Exposed credentials

Threat level: Low

Description:

The file `/etc/squid/squid.user` contains an unused encrypted user/password combination for the user `paskoocheh`.

Technical description:

The file `/etc/squid/squid.user` contains an encrypted user/password combination for the user `paskoocheh`.

```
paskoocheh:$apr1$pFaD5saL$FBDyLlzz7BLzbehGqueFz.
```

Authentication was commented out in the main configuration file of the Squid service, so credentials are not required to log in to the server.

The password prefix `$apr1$` defines the hashing schema to use a modified (hashed and iterated) version of MD5, which has been considered broken since 2006, and is relatively easy to reverse.

Confirmed that the squid password file is now empty.

Impact:

Even though it's not in use, attackers could attempt to find the cleartext password for this user that could potentially grant access to other systems.

Recommendation:

- Remove the password file entry

4.1.12 ASL-012 — Squid - Caching Enabled

Vulnerability ID: ASL-012

Vulnerability type: Information disclosure

Threat level: Low

Description:

Squid is configured to cache data if possible. Payloads from unencrypted client requests might be persisted to disk that could contain private user information.

Technical description:

Confirmed that the cache has been disabled by uncommenting the following line in squid.conf:

```
cache deny all
```

Impact:

Unencrypted personal data might be stored unintentionally in locations that are easier for an attacker to access.

Recommendation:

- Disable caching

4.1.13 ASL-013 — Dante - Ubuntu Has No Auto-updates Enabled

Vulnerability ID: ASL-013

Vulnerability type: Configuration

Threat level: Low

Description:

The Ubuntu server does not install security updates automatically.

Technical description:

Although there is no direct risk connected to this, it can help to get updates rolled out without manual involvement.

https://help.ubuntu.com/lts/serverguide/automatic-updates.html

Confirmed that unattended upgrades have been enabled.

Impact:

The server may remain susceptible to vulnerabilities even after patches are made available.

Recommendation:

- Install the `unattended-upgrades` package and configure it to install security patches automatically.

4.1.14 ASL-014 — Dante - IP Logging Enabled

Vulnerability ID: ASL-014

Vulnerability type: Information disclosure

Threat level: Low

Description:

Connecting client IP addresses appear in the Dante log file `/var/log/danted.log`.

Technical description:

It was not possible to prevent client IP addresses from being recorded in the server logs, however this may not matter because the Dante server only ever sees load balancer IP addresses, so no further action is required.

Impact:

IP addresses are considered personal data, so logging them should be avoided where possible.

Recommendation:

- Confirm whether client IPs are being logged.

4.1.15 ASL-015 — Squid - IP Logging Enabled

Vulnerability ID: ASL-015

Vulnerability type: Information disclosure

Threat level: Low

Description:

Proxy client IP addresses appear in the Squid server log file `/var/log/squid/access.log`.

Technical description:

The log format is defined in `/etc/squid/squid.conf`:

```
logformat squid %tl %6tr %>a %Ss/%03Hs %<st %rm %ru %un %Sh/%<A %mt
```

To respect user privacy the line should be changed to:

```
logformat squid %tl %6tr %Ss/%03Hs %<st %rm %ru %un %Sh/%<A %mt
```

The Squid configuration documentation describes client privacy related log fields:

```
Connection related format codes:

>a  Client source IP address
>A  Client FQDN
>p  Client source port
>eui    Client source EUI (MAC address, EUI-48 or EUI-64 identifier)
>la Local IP address the client connected to
>lp Local port number the client connected to
>qos    Client connection TOS/DSCP value set by Squid
>nfmark Client connection netfilter mark set by Squid

la  Local listening IP address the client connection was connected to.
lp  Local listening port number the client connection was connected to.

<a  Server IP address of the last server or peer connection
<A  Server FQDN or peer name
<p  Server port number of the last server or peer connection
<la Local IP address of the last server or peer connection
<lp     Local port number of the last server or peer connection
<qos    Server connection TOS/DSCP value set by Squid
<nfmark Server connection netfilter mark set by Squid
```

Confirmed that the log format string has been altered as recommended.

Impact:

IP addresses are considered personal data, so logging them should be avoided where possible.

Recommendation:

- Disable logging of client IP addresses.

4.1.16 ASL-016 — HAProxy - IP Logging Enabled

Vulnerability ID: ASL-016

Vulnerability type: Information disclosure

Threat level: Low

Description:

Proxy client IP addresses appear in the HAProxy server log file `/var/log/haproxy.log`.

Technical description:

HAProxy's default logging format (for TCP proxies) includes the client IP address. The default TCP log format is defined as this:

```
"%ci:%cp [%t] %ft %b/%s %Tw/%Tc/%Tt %B %ts %ac/%fc/%bc/%sc/%rc %sq/%bq"
```

The `%ci` and `%cp` elements correspond to client ip and port. To remove client IP and port, this should be changed in `/etc/haproxy/haproxy.conf` to:

```
log-format "[%t] %ft %b/%s %Tw/%Tc/%Tt %B %ts %ac/%fc/%bc/%sc/%rc %sq/%bq"
```

The HAProxy configuration documentation (http://cbonte.github.io/haproxy-dconv/1.8/configuration.html#8.2.4)describes the available log fields.

HAProxy also logs client IP in the error log file, as described in their docs here: http://cbonte.github.io/haproxy-dconv/1.8/configuration.html#8.2.5. It does not seem possible to alter this format or disable error logging altogether, so the next best approach is to rotate and clear logs frequently.

Confirmed that the log format has been altered as recommended.

Impact:

IP addresses are considered personal data, so logging them should be avoided where possible.

Recommendation:

- Disable logging of client IP addresses in the HAProxy log.
- Configure frequent clearing/rotation of error logs.

4.2 Non-Findings

In this section we list some of the things that were examined but turned out to be dead ends or not to matter.

4.2.1 NF-001 — Duplicate fail2ban ssh config

Not a finding, but a minor configuration bug. fail2ban has two jails that are checking the same thing. The default `/etc/fail2ban/jail.conf` includes an `[sshd]` jail that checks `auth.log`; this is duplicated in `/etc/fail2ban/jail.local` under the name `[ssh]`.

Confirmed that this has been resolved.

4.2.2 NF-002 — HTTPS Config Not Hardened

The HTTPS config within haproxy is not hardened, and includes things like outdated 3DES ciphers, however this doesn't matter because haproxy is only serving TCP proxies, not HTTPS. I can see there is a revised cipher list in haproxy.cfg, but it's commented out, and the old one is still active.

Confirmed that the revised cipher list has been enabled but it has no overall impact.

4.2.3 NF-003 — Unclear why the HAProxy stats interface is not global

The haproxy docs suggest that the stats endpoint be configured in the global section rather than as a listener. The stats socket config directive supports all the same syntax that bind does, so I'm unclear why the stats socket has been defined as a separate listener.

Confirmed that the config was moved to the global section, but it doesn't matter anyway because the stats interface is disabled altogether.

4.2.4 NF-004 — Dante - Debugging Disabled

Dante's debug logging can possibly leak data, but it was disabled and consequently not a risk; only more serious "global" errors are logged.

4.2.5 NF-005 — Dante - Connection Logging

Although the Dante server is configure with connection logging enabled, user privacy is not harmed because only internal connections from HAProxy instances are accepted and logged.

The configuration lines from `/etc/danted.conf` defining the logging behavior are applied equally in all sections:

```
log: connect disconnect error ioop
```

5 Future Work

- **Regular retests**
  Given the sensitive nature of ASL19's work, we recommend regular retesting of infrastructure and configuration to ensure that client privacy and security is conserved.

6 Conclusion

A number of configuration problems were found, raising security, privacy, and practical issues. None of the findings were especially severe in security terms, but privacy concerns are especially important in this case.

The HAProxy connection count limits were set unrealistically high, which would be likely to adversely affect the server's ability to be able to cope at high traffic levels, and related kernel options were not configured to match.

HAProxy was an outdated version, and its configuration had some security issues, mainly relating to the stats monitoring service.

HAProxy, Squid, and Dante were all storing excessive information via logging, caching, and core dump options, especially client IP addresses. An unused (on this server) encrypted password was using a weak hash, possibly exposing other servers to attack.

ASL19 was very responsive and addressed all of the findings immediately, and these changes were verified in a retest.

Finally we want to emphasize that security is a continuous process; this penetration test is just a one-time snapshot. Security posture must be continuously evaluated and improved. Regular audits and ongoing improvements are essential in order to maintain control of your corporate information security. We hope that this pentest report (and the detailed explanations of our findings) will contribute meaningfully towards that end. Do not hesitate to let us know if you have any further questions or need further clarification of anything in this report.