perdition: Mail Retrieval Proxy

Horms (Simon Horman)
Verge Networks – horms@verge.net.au
VA Linux Systems Japan, K.K. – horms@valinux.co.jp

January 2003

http://verge.net.au/linux/perdition/

Introduction and Overview

Perdition is a fully featured POP3[6] and IMAP4[1] proxy server. It is able to handle Plain-Text, SSL and TLS connections and connect end-users to a real-server based on a database lookup. Perdition supports modular based database access. The API for modules is open, allowing arbitrary modules to be written to allow access to any data store.

Perdition has many uses. Including, creating large mail systems where an end-user's mailbox may be stored on one of several hosts, integrating different mail systems together, migrating between different email infrastructures, and bridging plain-text, SSL and TLS services. It can also be used as part of a firewall.

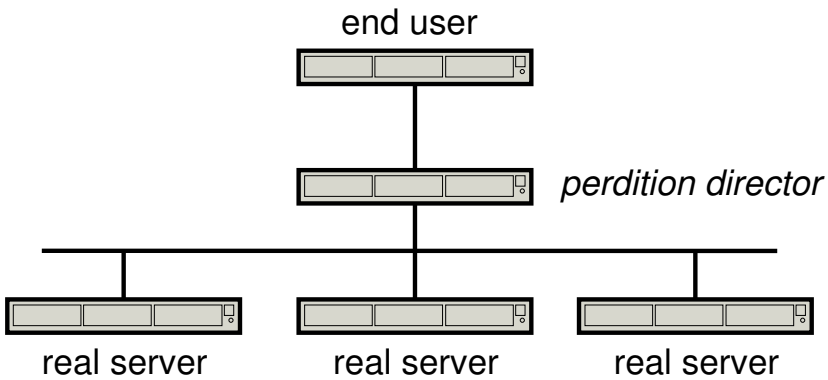

Figure 1: Basic Architecture

Perdition handles the authentication phase of end-users' connections. More specifically, when a connection is made to perdition in POP3 mode, it reads the USER and PASS commands and then checks its popmap to find where the user's connection should be forwarded. A connection to the real-server is made and perdition enters the USER and PASS commands. If authentication is successful then perdition pipes data between the client and the real-server. If authentication fails then the real-server server connection is closed and an error is returned

to the end-user. Similarly in IMAP4 mode, perdition accepts the LOGIN command and passes the username and password supplied onto the real-server specified in the popmap.

When perdition is piping information between the end-user and real-server it does not interpret the data, it merely reads bytes from the end-user and sends then to the real-server and vice versa. This means that perdition only understands the POP3 and IMAP4 commands that are used in the authentication phase. It does not interpret the communication between the end-user and the real-server. This greatly reduces the amount of the POP3 and IMAP4 protocol that perdition needs to understand, reducing the complexity of the code.

Plain-Text, SSL and TLS

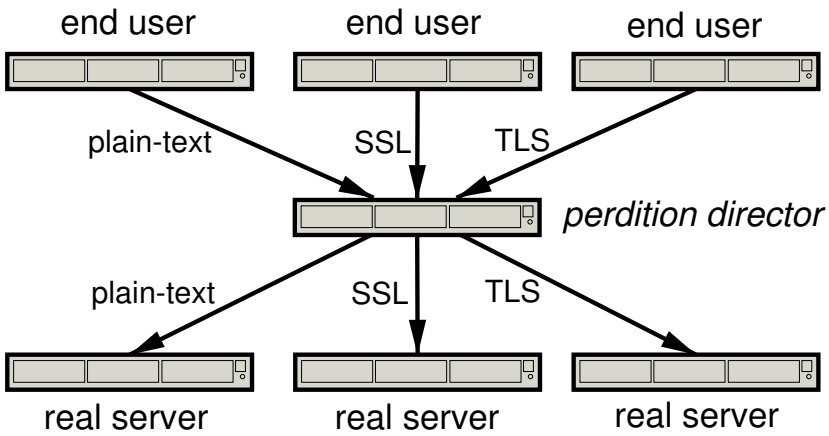

Figure 2: Plain-Text, SSL and TLS Modes

Perdition can independently accept plain-text, SSL and TLS[7] connections from end-users and make plain-text, SSL or TLS connections to real-servers. In this way perdition can be used to bridge between plain-text, SSL and TLS services. For example, suppose a system is running a POP daemon that does not support SSL, but the end user is downloading mail from an SSL enabled client. By running perdition, listening for SSL connections, on the same or another box, and connecting to the plain-text only daemon an SSL POP service can be provided to end-users.

Popmap

The database that perdition uses to determine which server an end-user's connection should be forwarded to is called the *popmap*. The lookup functions for the popmap are provided by a a map-library whose symbols are loaded at run-time. By creating different map-libraries it is possible for perdition to lookup user information using any database source. The current revision provides map-libraries to resolve lookups using LDAP, ODBC, MySQL, PostgreSQL, GDBM, Berkeley DB, POSIX Regular Expression and NIS.

Input

The key for a popmap lookup is the username provided by the end-user. The username supplied by the user is referred to as the long username. The portion of the long username before the run-time configurable domain delimiter, an at-symbol ('@') by default, is referred to as the short username. The portion of the long username after the domain delimiter is referred to as the domain. Thus, if the domain delimiter is omitted, the short username is the same as the long username and there is no domain.

Long Username	Short Username	Domain
mary@verge.net.au	mary	vergenet.net
bob	bob	-

Figure 3: Username Components

Query Key

Perdition is able to build up a query key using the components provided in the long username by the end-user and the source and destination IP address of the connection. The query key is built up by using a format-string which contains a combination of string-literals and escape sequences.

Escape Sequence	Entity
\U	long username
\u	short username
\D	domain delimiter
\d	domain
\i	source IP address
\I	destination IP address
\p	source port
\P	destination port
\\	Literal \

Figure 4: Query Key Escape Sequences

Multiple keys may be provided. A comma (',') is used to delimit keys. Each key is queried in order, and the first valid result retrieved is used. For example, search for the short username, domain delimiter and destination IP address in the database. If this fails search for the short username, domain delimiter and the string "default".

```
\u\D\I,\u\Ddefault
```

Thus, perdition is able to search the database in a very flexible manner. This flexibility allows perdition to be deployed for a wide range of applications. As discussed in the applications section.

Result

The result of the query is the real-server to which a connection should be made. Optionally, the server-name may be suffixed with a colon (':') and the port to connect to. The server may also be prepended with a username and the domain delimiter.

Result	Username	Server	Port
pop0.verge.net.au	-	pop0.verge.net.au	-
pop0.verge.net.au:110	-	pop0.verge.net.au	110
mary@pop0.verge.net.au	mary	pop0.verge.net.au	-
mary@pop0.verge.net.au:110	mary	pop0.verge.net.au	110

Figure 5: Result Components

If a username is provided, then it may be used as the username to log into the real-server. In this way usernames may be translated. This is useful for avoiding user-space collisions when integrating different mail systems together. If a port is provided then it will override the default port for real-server connections that is set at run-time. This allows for situations where different real-servers may be listening on different ports for some reason. A server must always be supplied as part of the result.

No Result

If no result is returned from the database all is not lost. A comma delimited list of default servers may be supplied. Each server in the list has the name of the server, optionally followed by a colon (:) and the port to connect to. For example:

pop0.vergenet.net:110,pop1.vergenet.net:110

A username cannot be provided as part of the default server, thus username translation cannot take place when the default server is used. These servers will be selected in a round-robin fashion.

It may be desirable for all connections to be forwarded to the default servers. In this case a database lookup is not necessary. Rather than make queries to an empty database, by specifying the empty string ("") as the map-library to use, no database lookup will be attempted. By providing a single default server and no map-library it is possible to forward all connections to the same real-server. This is useful for providing SSL access to an otherwise plain-text only POP or IMAP daemon. It is also useful for proxying connections as part of a firewall.

POP/IMAP Before SMTP

Generally speaking it is undesirable to allow hosts outside of the local network to use a mail server as a mail relay. Doing so provides an open relay that can be used by third parties to send unsolicited email. For this reason many mail servers are configured not to relay mail, other than from a very select list of IP addresses, often the local network and nothing else. However,

in some situations it is desirable to allow end-users outside of the local network to relay email through a server. For example, an employee who is on the road using dial-up connections. Probably the best solution to this problem is to have clients authenticate themselves when sending mail using SMTP AUTH[5]. If this is not possible, POP/IMAP before SMTP may be used.

POP/IMAP before SMTP works by recording the IP address of end-users that are authenticated for POP or IMAP access. These IP addresses are added to a list of IP addresses that are allowed to relay mail. Typically IP addresses on this list are valid for a window of time and are removed from the list once this window expires.

Perdition logs connections in three phases to allow it to work in conjunction with POP/IMAP before SMTP software which monitors logs and maintains a list of IP addresses that may be relayed.

When a user connects:

Connect: <source_ip_address>[inetd_pid=<pid>]

When a user is authenticated:

Auth: <source_ip_address> user="<username>"
server="<servername"> port="<port>" status=failed|ok

When a user disconnects:

Close: <source_ip_address> user="<username>" received=<bytes> sent=<bytes>

Given that IMAP connections may last for a very long time, many mail readers keep an IMAP connection open until the user quits the mail reader. Thus the connection may be open for much longer than the user's IP address will last in the POP/IMAP before SMTP relay list. For this reason perdition can be configured to periodically reissue the "Auth" log.

Perdition also provides a POP/IMAP before SMTP daemon, perdition-pbs. This works by monitoring the system logs for the "Auth" line. It adds the IP addresses to a Berkeley database. Periodically entries are expired from this database.

Perdition-PBS can be used in conjunction with sendmail by adding a rule to sendmail.cf that is able to use this database as a list of hosts to relay mail from. It can also be used in conjunction with qmail by using a wrapper for qmail-smtp which sets the RELAYCLIENT environment variable if a connection is received from an IP address in the database.

It is thought that by using TDBrepl[8], a simple database replication system built on top of TDB[2], instead of Berkeley DB for the back-end database it is possible to use perdition-pbs in a distributed environment.

Applications

The original design and implementation of perdition was intended to allow a mail service to grow beyond a single machine. However, it soon became apparent that it could be used to

solve a variety of problems. Today its features support many different applications.

Scaling

As the number of end-users accessing an email system, and the amount of email that they are sending and receiving increases it is often necessary to grow an email system beyond a single machine. There are many different ways of achieving this. One is to divide the end-users between different mail servers[4].

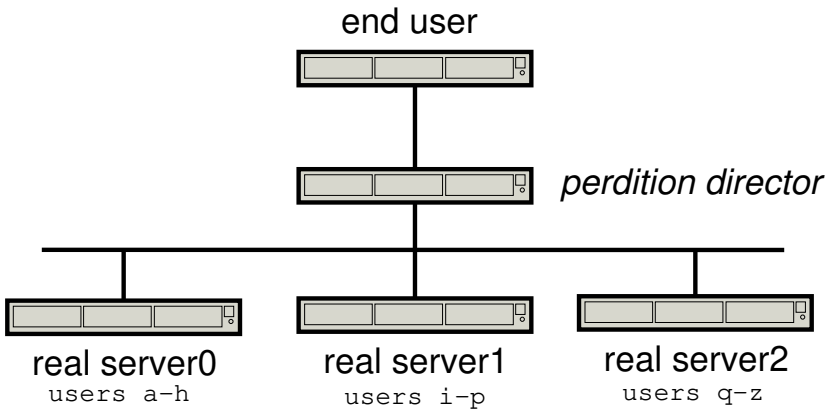

Figure 6: Using Perdition to Scale a Mail Service

For example, suppose that there are three mail servers. And that end-users are split between the mail servers based on a simple match on the first letter of their login-name. end-users a-h are allocated to real-server0, i-p to real-server1 and q-z to real-server2. To retrieve their email, end-users may connect to the perdition director which proxies their connection to the real server that their mail is on. This can be done simply by using a regular expression popmap on the perdition-director.

```
^[a-h]: real-server0
^[i-p]: real-server1
^[q-z]: real-server2
```

This method allows additional real-servers to be added to the system without end-users needing to change their settings as they always connect to the perdition-director. If additional directing capacity is required, then more perdition-directors may be added and load balanced using Layer 4 Switching or Round-Robin DNS.

Integration

When organisations merge it may be desirable to consolidate email serving infrastructure. Unfortunately there is often username clashes when doing this. Perdition is able to resolve this problem by distinguishing between users based on either a domain-name supplied by an end-user or the IP address that the end-user connects to.

User-Supplied Domain Name

Users may supply a domain name when connecting to the perdition-director. This is done by following their username with a domain delimiter and the domain name. Perdition can use this domain to distinguish between two end-users from different domains with the same username and can map the username such that on the underlying real-server the users have different usernames.

For example, bob@foo.com becomes bob1 and bob@bar.com becomes bob2. Perdition then opens up a connection to the real-server, which could be the same host as the perdition-director and accesses the mapped user account. This system works particularly well in an environment where end-users already have their mail-clients configured to include the domain name as part of their login name.

Perdition-Director with Multiple IP Addresses

Another approach to identifying end-users based on their domain is to configure the perdition-director with multiple IP addresses. Thus, in DNS a different IP address can be used for each domain. The IP address that an end-user connects to can be used as part of the query key for the database lookup. Thus is bob@foo.com connects to pop3.foo.com, then the database lookup could be bob@10.0.0.1. Similarly the lookup for bob@bar.com could be bob@10.0.0.2. Again, this can be used to map the two Bob's to a unique username and connect to the corresponding real-server.

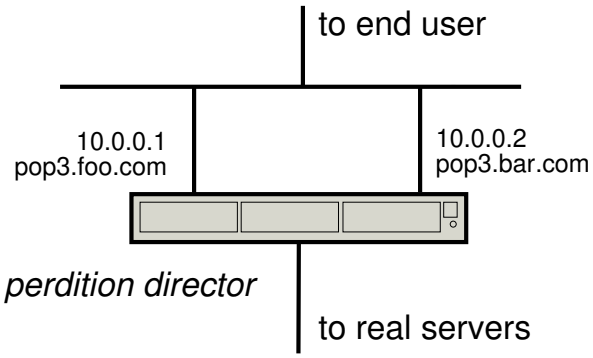

Figure 7: Perdition-Director with Multiple IP Addresses

This approach is particularly useful in an environment where end-users have not already set up their mail-clients to supply their domain as part of their login information. End-users continue to access their email using the same POP or IMAP server as before. Perdition is able to direct their request to the correct mailbox based on information about the connection itself. The disadvantage of this approach is that it requires each domain to have a separate IP address for its POP and IMAP servers.

Essentially, having the user supply a domain name is equivalent to an HTTP name-based virtual host[3]. While configuring perdition with multiple IP addresses is equivalent to an IP-based virtual-host. Both techniques may be used simultaneously on the same perdition-director for different domains.

Migration

Sometimes when new mail infrastructure is deployed it is useful to migrate end-users gradually from the existing system to the new system. Using perdition this can be done without end-users needing to change their settings.

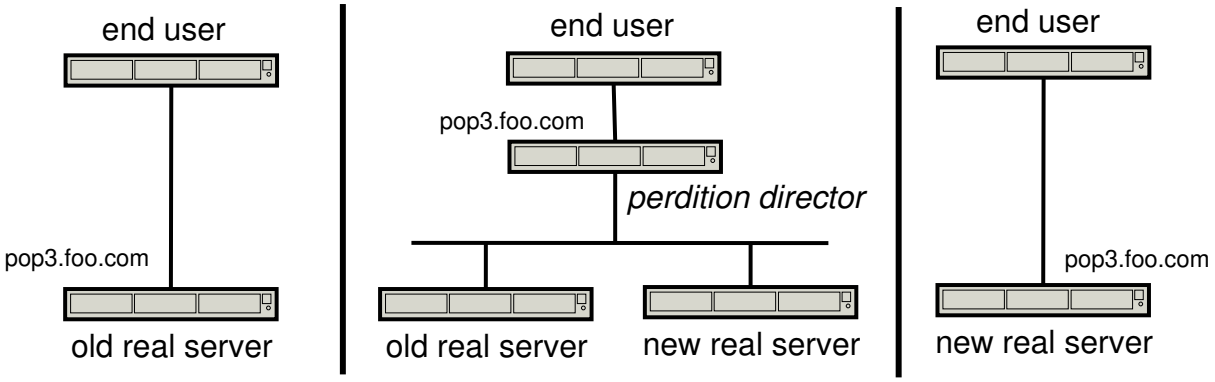

Figure 8: Using Perdition to Migrate a Mail Service

Suppose that an organisation has a POP server, pop3.foo.com and want to gradually move end-users over to a new server. This can be done by having perdition set up with the IP address of pop3.foo.com and the new and old servers set up as real servers. The popmap on the perdition director can be used to determine which end-users have their connections forwarded to the old server and which end-users have their connections forwarded to the new server. Once all the end-users have been migrated to the new server, the perdition director can be removed.

Firewall

Perdition can be used as part of a firewall to proxy outgoing POP and IMAP requests. End-users' connections can be forwarded to an external real-server according to a popmap. Alternatively, perdition can be configured to use the domain supplied by the end-user's mail-client as the real server to connect to. For example, if an end-user connects to the perdition director and logs-in as bob@pop3.foo.com, then perdition will connect the user to pop3.foo.com.

Perdition can also be configured to authenticate the user, thus limiting access. This is, however, extremely limited as the same username and password is used to authenticate with both perdition and the real server.

It is important to note, that once the user is authenticated with the real-server, perdition does not interpret the POP or IMAP commands sent by the end-user's mail-client or the responses sent by the real-server. Thus, it cannot be used to protect real-servers from malicious end-users or vice versa.

Availability

Perdition is implemented in C. It is available under the terms of the GNU General Public Licence from `http://www.vergenet.net/linux/perdition/`. It is also distributed as part of Debian GNU/Linux. The primary development platform for perdition is Linux, although it is known to work well on other Unixes including Sun Solaris and FreeBSD. Contributions and bug reports are always more than welcome.

Conclusion

Perdition provides a flexible way to proxy POP and IMAP connections from end-users to one or more real-servers. Pipex, an ISP in the United Kingdom, WorldCom Europe, Belgacom Skynet, the largest ISP in Belgium, Ohio State University, SoVerNet, and ISP in Vermont, USA, Fastmail.FM, and NetCologe, and ISP in Colonge, Germany are using Perdition as a front-end to multiple real-servers. GPS-Technik in Switzerland and Osprey Network Technologies in Kansas, USA are using perdition to translate account names for end-users when integrating mail services together. These systems use a range of database-backends, including LDAP, MySQL and Berkely DB. The size of these deployments range from 2,500 to in excess of 250,000 accounts with up to 8 real-servers.

References

[1] M. Crispin. Rfc 1730: Internet message access protocol – version 4, December 1994. http://www.ietf.org/rfc/rfc1730.txt.

[2] Andrew Tridgel et al. Tdb – trivial database. http://sourceforge.net/projects/tdb/.

[3] The Apache Software Foundation. Apache virtual host documentation. http://httpd.apache.org/docs/vhosts/.

[4] Simon Horman. High capacity email, November 1999. http://vergenet.net/linux/mail_farm/.

[5] J. Myers. Rfc 2554: Smtp service extension for authentication, March 1999. http://www.ietf.org/rfc/rfc2554.txt.

[6] J. Myers and M. Rose. Rfc 1939: Post office protocol – version 3, May 1996. http://www.ietf.org/rfc/rfc1939.txt.

[7] C. Newman. Rfc 2595: Using tls with imap, pop3 and acap, June 1999. http://www.ietf.org/rfc/rfc2559.txt.

[8] Liam Widdowson. Tdbrepl – the trivial database replication system. http://tdbrepl.inodes.org/.

Special thanks to Kfish, Raster, Alex, Jake, John Ferlito and Nerdy Amanda Lin.