WASP

SoftwareArt Corporation

WASP Mail Server

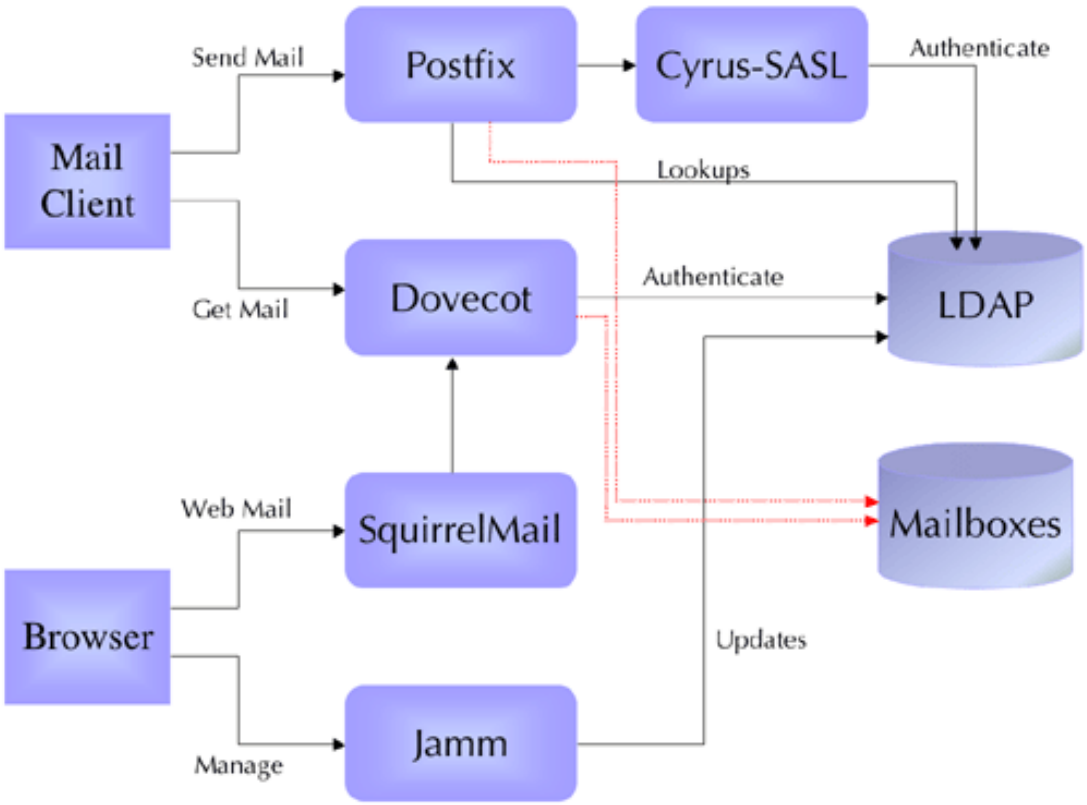
Introduction:

WASP Mail Server is a complete email messaging solution designed for small to mid-sized businesses and is backed by the industry's best technical support team.

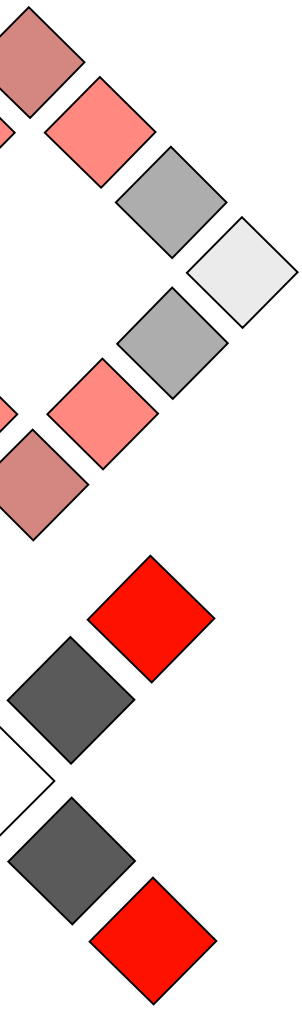
WASP can run on UNIX-like systems including AIX, BSD, HP-UX, Linux, MacOS X, Solaris, and more.

WASP Mail Server is a complete email server with SMTP, POP, IMAP, LDAP, and List Server. WASP users can send and receive email using any standards-based client, including Microsoft Outlook®, Outlook Express®, or Eudora®. Or, users can access email from anywhere via WASP's customizable Web messaging.

WASP is designed to be an easy-to-administer tool while providing a secure alternative to the widely-used Sendmail MTA.



WASP is a very powerful tool yet scalable enough to allow the flexibility to customize the tool to deliver mail locally for one local machine or setup as a full Corporate mail server for a corporation with more than 100 employees or it can be set up as Full Mail Cluster with the help of clustering software. There is no reason/need for any individual or small organization to be stuck with the options their local ISP provides. With WASP can setup your mail the way you want to, abiding by rules defined by you. It provides features to users to block SPAM from a particular geographical area or country. If there is a need to accept mails from trusted associates, then tool provides that option as well. What about setting up an email address for all your friends and family? No problem.



Mail transfer agent [MTA] that routes and delivers electronic mail, intended as an alternative to the widely used Sendmail MTA.

Typical deployment:

As an SMTP server, WASP implements and provides the first layer of defense against spambots and malware. WASP is often combined with other software that provides spam/virus filtering (e.g., Amavisd-new), message store access (e.g., Dovecot), or complex SMTP-level access policies (e.g., postfwd, policyd-weight or greylisting).

As an SMTP client, WASP implements a high-performance parallelized mail delivery engine. Here, WASP is often combined with mailing list software (e.g., Mailman).

Features:

WASP implements a number of features in the MTA.

Main built-in features Includes (some are listed below):

- Standards- Provides compliant support for LMTP, STARTTLS encryption, SASL authentication, MIME encapsulation and transformation, DSN delivery status notifications, IPv4, and IPv6
- Configurable SMTP-level access policy that automatically adapts to overload
- Option to create "Virtual" domains with distinct address-namespaces
- UNIX-system customization interfaces for command-line submission, delivery to command, and direct delivery to message stores in mbox and maildir format
- Light-weight content inspection based on regular expressions
- Large number of database lookup mechanisms including Berkeley DB, CDB, Open LDAP LMDB, Memcached, LDAP and multiple SQL database implementations
- A sophisticated scheduler that implements parallel deliveries, with configurable concurrency and back-off strategies
- A scalable zombie blocker that reduces SMTP server load due to botnet spam

Typical WASP extension features:

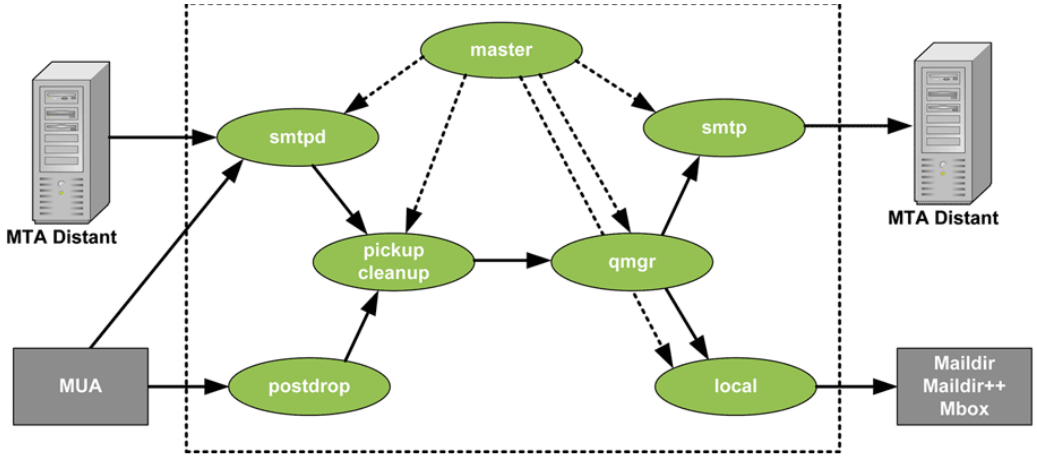
WASP extensions use the SMTP or Milter (Sendmail mail filter) protocols which provides full control over the message content and envelope as well as a simple text-based protocol that enables complex SMTP-level access control policies.

- Deep content inspection before or after a message is accepted into the mail queue
- Mail authentication with DKIM, SPF and other protocols
- SMTP-level access policies such as greylisting or rate control.

Operating systems:

WASP runs on AIX, BSD, HP-UX, GNU/Linux, OS X, Solaris and, generally speaking, on every Unix- style operating system that ships with a C compiler and delivers a standard POSIX development environment. It uses the default MTA for the OS X, NetBSD and Ubuntu operating systems.

Architecture:



WASP consists of a combination of server programs that run in the background, and client programs that are invoked by User defined programs or system administrators.

The WASP core consists of several dozen server programs that run in the background, each specializing in handling specific aspect of email delivery. Example includes the SMTP server, the scheduler, the address rewriter, and the local delivery server.

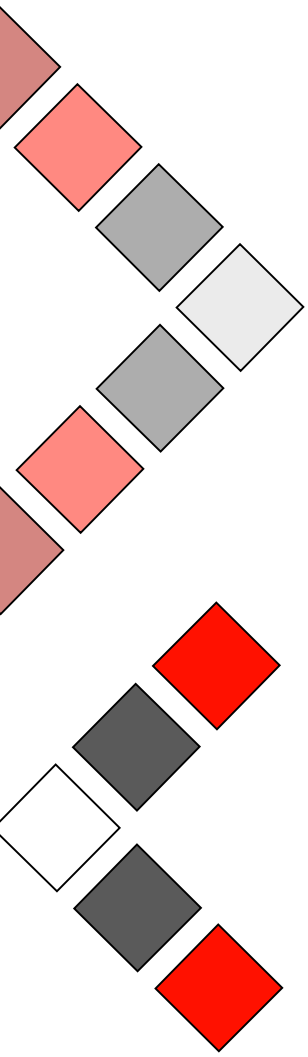
For damage-control purposes, most server programs run with least privileges and terminate voluntarily after processing any number of requests. In order to conserve system resources, most server programs terminate when they become idle.

Client programs run outside the WASP core. They interact with Postfix server programs through mail delivery instructions in the user's ~/.forward file, and through small "gate" programs to submit mail or to request queue status information.

Administrative support tools are included to start or stop Postfix, manipulate the queue, query status information and to examine or update its configuration files.

Implementation:

The WASP implementation uses safe subsets of the C language and API of the POSIX system. These subsets are encapsulated under an abstraction layer that contains almost 50% of all WASP source code. This provides the foundation and Framework on which all WASP programs are built. Example like the "vstring" primitive makes WASP code resistant to buffer overflow attacks, and the "safe open" primitive makes WASP code resistant to race condition attacks on systems that implement the POSIX file system API. This abstraction layer does not affect the attack resistance of non-WASP code, such as code in system libraries or third-party libraries.



Robustness:

Conceptually WASP manages pipelines of processes that handle the responsibility for message delivery and error notification from one process to the other. All message and notification "state" information is persisted in the file system. The processes in a pipeline operate mostly without centralized control; this relative autonomy simplifies error recovery. When a process fails before completing its part of a file or protocol transaction, its predecessor in the pipeline backs off and retries the request later and its successor in the pipeline discards unfinished work. Many WASP daemons can simply "die" when they run into an issue; they automatically restart when the next service request arrives. This approach makes WASP highly resilient, as long as the operating system or hardware does not encounter catastrophic errors or issues.

Performance:

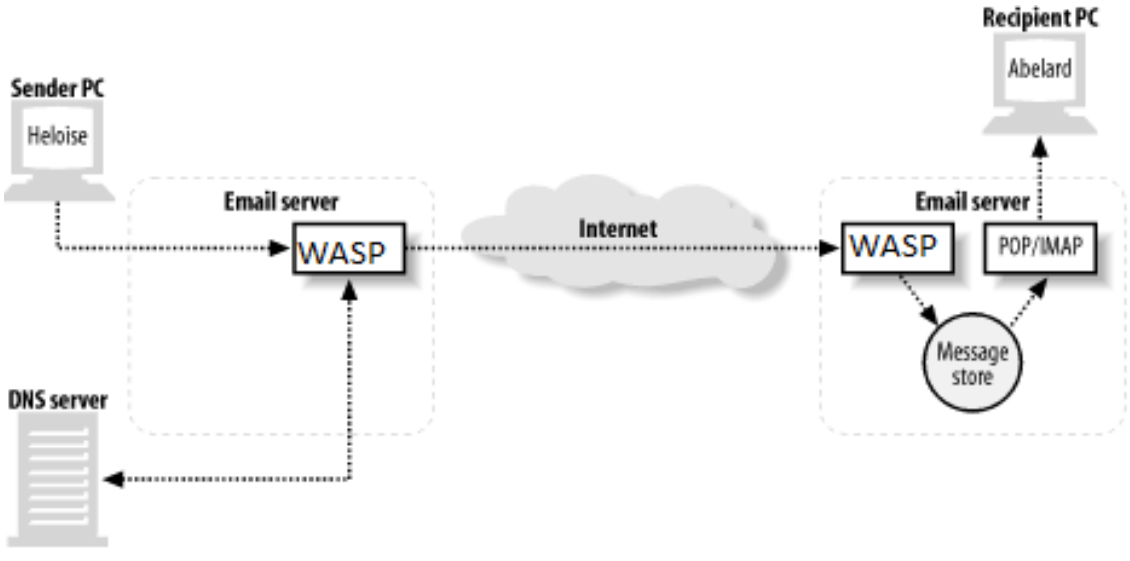
WASP has been clocked at delivering ~300 messages/second across the Internet, running on commodity hardware (a vintage-2003 Dell 1850 system with battery-backed MegaRAID controller and two SCSI disks). This delivery rate is an order of magnitude below the "intrinsic" limit of 2500 message deliveries/second that was achieved with the mail queue on a RAM disk while delivering to the "discard" transport.

Mail systems such as WASP achieve high performance by delivering mail in parallel sessions. With mail systems such as Sendmail and Exim that make one connection at a time, high performance can be achieved by submitting limited batches of mail in parallel, each batch is handled by a different process. WASP require parallel submission into different MTA instances once they reach their intrinsic performance limit or performance limits of the hardware or operating system.

It should be noted that the delivery rates cited above are largely academic. With bulk mail delivery, the true delivery rate is primarily determined by the receiver's mail receiving policies and by the sender's reputation.

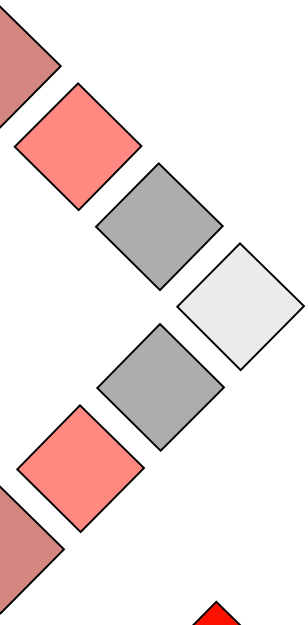
The Role of WASP:

Figure illustrates a simple example of message transmission where WASP handles the responsibilities of the MTA and local delivery. As the MTA, WASP receives and delivers email messages over the network via the SMTP protocol. For local delivery, the WASP local delivery agent can deposit messages directly to a message store or hand off a message to a specialized mail delivery agent.



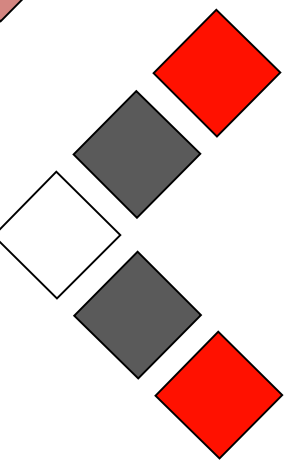
WASP Security:

Email systems are necessarily exposed to possible attacks because their functionality requires accepting data sometimes from untrusted systems. The challenge is to build systems those are resistant to attack and provide good security strategy including multiple layers of protection. This is particularly true for public systems in a potentially hostile environment. WASP takes a proactive and multilayered approach to security. WASP architecture limits the severity of vulnerabilities, even if there are designs or coding errors that might otherwise create major vulnerabilities in a monolithic privileged program.



Modular Design:

The modular architecture of WASP forms the basis for its security. Each WASP process runs with the least amount of privilege necessary to get its particular job done. Many of Sendmail's security problems were exacerbated because Sendmail runs as a privileged process most of the time. WASP operates with the minimum privilege necessary to accomplish a particular task. WASP processes that are not needed on a system can be turned off, making it impossible to exploit them. For example a network firewall system that only relays mail and does not need local delivery can have all the WASP components for local delivery turned off. WASP processes are insulated from each other and depend very little on any inter-process communication.



Shells and Processes:

In most cases, delivery of mail does not require a Unix shell process, but when a configuration does make use of one, WASP sanitizes information before placing it into environment variables. WASP tries to eliminate any harmful characters that might have special meaning to a shell before making any data available to the shell.

Most WASP processes are executed by a trusted master daemon. They do not run as child processes, so they are immune to any of the security problems that rely on parent-child inheritance and communications.

The attacks which use signals, shared memory, open files, and other types of inter-process communication are essentially useless against WASP.

Security by Design:

A buffer overflow is another common type of attack against applications. In this type of attack, crackers cause a program to write to unauthorized memory. Doing so might allow them to change the path of execution in order to take control of the process. As mentioned above that WASP processes run with least privileges as possible making these attacks limited in scope furthermore WASP avoids using fixed-size buffers for dynamic data, making a successful buffer overflow attack highly unlikely.

An important security protection available on Unix systems is the ability to *chroot* applications. A chroot establishes a new root directory for a running application such as /var/spool/mail. When that program runs, its view of the file system is limited to the subtree below /var/spool/mail, and it cannot see anything else above that point. Your critical system directories and any other programs that might be exploited during an attack are not accessible.

WASP Architecture:

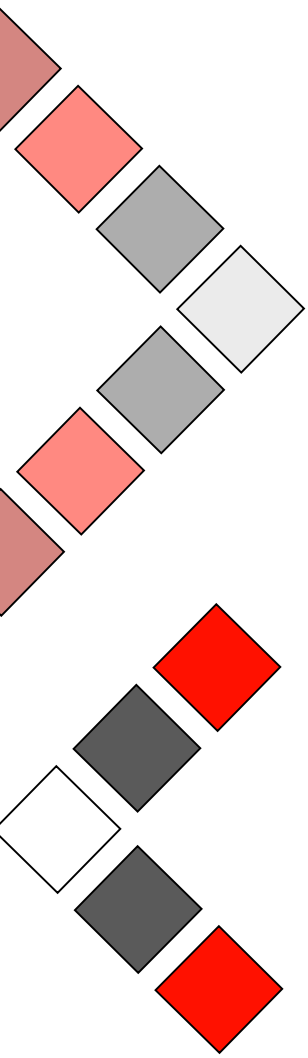
Admin can easily manage and operate WASP with limited understanding of the WASP complete functionality.

WASP consists of a combination of server programs that run in the background, and client programs that are invoked by user programs or by system administrators.

Main WASP built-in features

Standards-compliant support for LMTP, STARTTLS encryption, SASL authentication, MIME encapsulation and transformation, DSN delivery status notifications, IPv4, and IPv6

- Configurable SMTP-level access policy that automatically adapts to overload
- "Virtual" domains with distinct address-namespaces
- UNIX-system interfaces for command-line submission, for delivery to command, and for direct delivery to message stores in mbox and maildir format
- Light-weight content inspection based on regular expressions
- A large number of database lookup mechanisms including Berkeley DB, CDB, OpenLDAP LMDB, Me cached, LDAP and multiple SQL database implementations
- A sophisticated scheduler that implements parallel deliveries, with configurable concurrency and back-off strategies
- A scalable zombie blocker that reduces SMTP to botnet spam server load due



Typical WASP extension features:

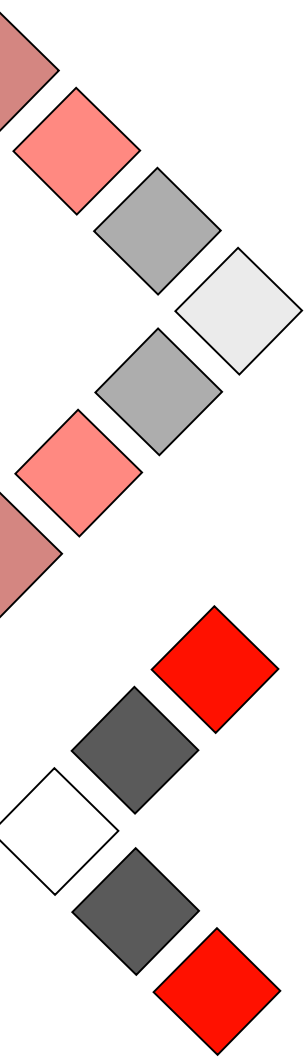
WASP extensions use the SMTP or Milter (Sendmail mail filter) protocols which both give full control over the message envelope and content, or a simple text-based protocol that enables complex SMTP-level access control policies.

- Deep content inspection before or after a message is accepted into the mail queue;
- Mail authentication with DKIM, SPF, or other protocols;
- SMTP-level access policies such as grey listing or rate control.

Technologies Used:

The following are few of the basic technologies.

1. Cyrus IMAP
2. Cyrus SASL
3. Open SSL
4. OpenLDAP
5. IMAP and POP3



Comparison of Mail Servers:

Mail Server	Linux/ Unix	Windows	Mac Os	SMTP	POP3	IMAP	IMAP IDLE	SMTP over TLS	POP over TLS
WASP	Yes	No	Yes	Yes	Dovecot, UW IMAP	Dovecot, UW IMAP	Dovecot,UW IMAP	Yes	No
qmail	Yes	No	Yes	Yes	Yes	Dovecot, UW IMAP	Dovecot,UW IMAP	No	Dovecot,UW IMAP
Sendmail	Yes	No	Yes	Yes	Dovecot, UW IMAP	Dovecot, UW IMAP	Dovecot,UW IMAP	Yes	No
Zimbra	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
IBM Lotus Domino	Yes	Yes	No	Yes	Yes	Yes	?	Yes	Yes
Microsoft Exchange Server	No	Yes	No	Yes	Yes	Yes	Yes[7][8]	Yes	Yes

IPV6	NNTP	SSL	Webmail	Active Sync	Database	File system	Other	License
Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Proprietary
via Qsmtp	?	No	No	No	(cdb)	Yes	No	Public domain
Yes	No	Yes	No	No	?	Yes	?	Sendmail License
No	No	Yes	Yes	Yes	Yes	Yes	No	ZPL and proprietary editions[17]
?	Yes	Yes	Yes	Yes	Yes	No	No	Proprietary
Yes (2007 sp1 onwards)[9]	Yes	Yes	Yes	Yes	ESE only	Yes (up to 2003 only)[10]	Yes	Proprietary