ELECTRONIC MESSAGE AUTHENTICATION - AUTOMATED WHITELIST ENGINE (AWE)

Cross Reference

Incorporated herein by reference is PCT/AU2006/001571 entitled "Electronic message authentication", published as WO2007/045049.

Technical Field

The invention electronic message authentication, such as email messages, to ensure valuable messages are reliably delivered to the recipient, while reducing the delivery of unwanted messages.

Background

The RealMail platform is essentially an Anti-Spam platform, with intelligent heuristics and whitelisting that shifts the Anti-Spam focus *away* from blocking Spam, and into the realms of e-mail protection by ensuring that legitimate messages between known parties are delivered.

Using RealMail in front of another Anti-Spam system enables this system to reap the benefits of the RealMail protection.

Brief Description of the Drawings

Fig. 1 is a schematic diagram of the AWE system.

Fig. 2 is an example flow chart that can be used by the AWE Decision engine.

Fig. 3 is example flow charts showing how the white lists can be built.

Fig. 4 is a schematic diagram of the components of a single node AWE server.

Disclosure of the Invention

AWE provides a product that may be integrated into their existing system:

- By OEM integration

- By installation of the AWE software
- By use of a standalone AWE system sitting alongside

AWE is NOT and Anti-Spam system, but provides the ability to integrate an intelligent Whitelisting system into an Anti-Spam system. This provides a significant reduction in False Positives and confidence that legitimate e-mail messages cannot be blocked by the Anti-Spam system.

The AWE system is designed to sit alongside a conventional Anti-Spam system, and be queried by the Mail Transport Agent (MTA). The AWE system is supplied with enough data to ensure that it can make a decision that is passed back to the MTA.

The MTA can then make use of this decision *in any way it wishes*, although guidelines for correct operation and interpretation of this decision may be provided.

The basic purpose of the AWE system is to provide an additional layer of protection for the AntiSpam system; the provision of an intelligent, automated whitelist engine that has the ability to stop False Positives by correct identification of legitimate senders.

As stated above, the AWE component is expected to be queried by the MTA, which is a core part of most Anti-Spam systems. However, it can be integrated into any part of the system that is able to make use of the AWE API to connect to the AWE system and make use of its responses.

Fig. 1 shows the typical modules that are in use. The AWE system may reside in one of three places:

- On a separate physical machine; queried across the network by the A/S system
- On the same server, installed as a software module
- On the same server, tightly integrated into the A/S system

AWE Decision engine

The AWE system builds local whitelists, based on the data that it is passed by the connected A/S system. Each query can contain a varied amount of data, but a full query will pass the following details for each message that the MTA processes:

- Originator's IP address
- Originator's e-mail address
- Recipient's e-mail address

These details can then be used to check the validity of the message sender, according to the following flow chart of Fig. 2.

Whitelist lookups

A whitelist entry consists of three pieces of information: IP address, e-mail address and domain name.

A lookup on the whitelist may return different levels of matching:

- Exact match: All three details have been found as a WL entry
- Partial match: The e-mail address has been found in an entry, and the IP address matches to the level specified in the configuration (i.e. a certain number of octets are the same)
- Shared match: The e-mail address has been found as a WL entry, but for a different recipient

The lookup to the global whitelist entails encrypting the e-mail address and IP address before carrying out the lookup, as the global list does not contain this information in a raw format (one way encryption techniques are used).

Whitelist building

This is shown in Fig. 3.

The AWE system is able to build whitelists using the data offered by the queries. An important part of this is to correctly identify inbound and outbound messages. An entry can be added to the whitelist when:

- An outbound message is sent. The recipient of the outbound message can be partially added to the white list (WL). This partial record may be enhanced with domain specific information and originating IP addresses, should the domain have an SPF record, or other means of determining their originating IP addresses.
- An inbound message is received from a partial WL entry user: If the WL user that matched does not have an IP address associated with their entry, it can be assumed that this is a response to an outbound message from a local user. The originating address can be added to the WL entry.
- Daily report: The AWE system can generate daily reports that can be sent to individual users or an Administrator. The report contains details of originators that were not found on the WL but which the AWE engine consider should be manually checked. The user/Admin can then choose to have these entries inserted into the WL.

AWE Functional Specification

This section details the functions that the AWE system must have, and the environments it will work within.

Environment

The AWE system can be run on a number of platforms, Linux, Windows, Solaris and AIX platforms.

AWE is software that is installed on a computer system, such as a server.

AWE has the:

- The ability to lookup rDNA data
- The ability to send WL information data to rDNA

- The ability to use extended domain flux and reputation data

Data storage

The AWE system will need to use local data storage for the WL, such as a hard drive. In one embodiment, the data storage used should not enable the casual observer to examine or display the whitelists directly.

Resources

Where caching and memory is used in the AWE implementation, a configuration option is provided to allow the installer to adjust the amount of memory used, such as applicable to the amount of CPU used.

Connectivity

The AWE system is connected with the Internet, in order to carry out its functions. These are expected to be:

- DNS lookups
- rDNA lookups
- rDNA upload of data
- Domain reputation lookups

DNS lookups will be managed by use of the local resolver, other lookups and uploads must be done via port 80/443.

Interfaces

The AWE system has a single internal HTTP API. This may change. However, it is also required that there be other interfaces written that interact with this API:

- A milter module (for postfix and sendmail integration)
- A bizanga integration module
- An exim extension (for re-integration into RealMail v3)
- An HTTP administration console

These interfaces will enable a number of MTAs to utilise the AWE system.

API commands are basically HTTP URLs, with the body of a POST request containing the required parameters.

It should be possible to interact with the API using encrypted request bodies, for when AWE is located across a WAN link.

The commands required of the API are documented in a section below.

Administration

Configuration file

The AWE configuration file contains static information that can be configured at installation time. However, this information should also be changeable via the Admin API. Information held in the configuration file should include (but not be limited to) the following information:

- License details
- Domain groups
- List of internal domains
- List of sending mail servers for each domain
- Reporting options

The configuration file should make it possible to configure all AWE functions so that the AWE system can be used without resort to the Admin API. (i.e. an installation should require a correct configuration file and the AWE environment to operate and no further actions).

The AWE configuration file should be dynamic, in that the AWE daemon will re-read the configuration after a change, rather than requiring a restart.

The API will also include some commands that result in modifications to the configuration file.

The configuration file will be split into sections, according to site code. This enables a single AWE instance to manage queries from different sites.

Admin API

The admin API is an integral part of the AWE HTTP API. It enables the admin user to:

- Configure the necessary start-up parameters for AWE
  - For each domain group:
    - List of local e-mail domains
    - List of local IP addresses that e-mail can originate from
    - Settings for use of shared WL and shared BL
    - Settings for partial IP address matching (no. of octets) for WL
    - Settings for partial IP address matching (no. of octets) for BL
- Manipulate the whitelists and blacklists
  - Add or remove entries
  - List or dump entries
- Enable production of reports
  - For licensing purposes (see above)
  - Resource purposes; data size on disk, memory usage
  - Performance purposes; number of queries handled, query throughput, breakdown of result codes

The Admin API encapsulates all the commands necessary to configure and operate AWE. Third parties can use the API to integrate AWE functions (such as add to whitelist) into their own user and admin interfaces.

However, there may also be a requirement to write an 'example' GUI which allows access to some or all of the API operations for Administrators and end users.

Reporting module

HTML Reporting will again be carried out by using the API functions to determine details on the AWE system.

Logging

AWE will log to two log files, locations specified in the configuration file:

- Request log: All queries made to the API will be logged here, detailing the API call made, the parameters used and the response given
- Error log: A general log file for all other logging information

In one embodiment, the request log can be limited to only log certain queries (simply by listing them in the configuration file) and to have the request log output in either syslog or CSV format.

Message counts

Message originators must be monitored on how many messages they are sending out each day. This requires a daily table that consists of the following fields:

- orig_envelope :The originators envelope address
- count Number: of messages sent today

rDNA updates

Relevant log entries will be pushed to the rdna server to continue building the global rDNA lists. For example, this data can be stored and then integrated into the rDNA data at a later date.

Checking whitelists

AWE will query and build local whitelists. Whitelists may be in the same format as those used by RealMail (whitelist_<domain>, blacklist_<domain>, whitelist_system and blacklist_system).

Alternatively, these whitelists may be held in a very different format (that does not require a database, or uses an internal database format to cut down on dependencies, and to ensure fast performance).

Details of whitelist checking

Whitelists are held in either the system whitelist table, or the domain whitelist table, which has both domain and user entries in it. When checking whitelists, the order is:

- Global whitelist (rdna)
- system whitelist
- system blacklist
- domain whitelist
- domain blacklist
- user whitelist
- user blacklist

As AWE will return a single response code, the lists will still need to be checked in this order. However, AWE should allow the order to be changed - in the configuration file. However, if there is no order specified in the configuration file, the default order will be used. This may be implemented as a hidden option.

Global whitelist

The check for rDNA is basically a DNS lookup, using md5 hashed values for originator and IP. Details can be found in:

/opt/BoxSentry/modules/STIMd/Test/on_rdna.pm

System whitelist

The system whitelist is checked by the following sql:

select originator from whitelist_system where (originator=<orig_address> or originator='*@<orig_domain>'

```
        or originator='*@*.<orig_domain_without_first _word>'
        or originator='*@*.<orig_tld>'
        or originator='*@<orig_domain_without_tld>.*')
        and originator != ""
```

So, for an originator of fred@sales.acme.com, the SQL would be:

```
        select originator from whitelist_system where
(originator='fred@sales.acme.com'
        or originator='*@sales.acme.com'
        or originator='*@*.acme.com'
        or originator='*@*.com'
        or originator='*@sales.acme.*')
        and originator != ""
```

Domain and user whitelist

The domain and user whitelist is checked using the following SQL. However, in AWE, there needs to be a separation of these lists, so an extra column should be used to tag each entry as either domain or user, which will be completed when an entry is added.

Also, AWE must not check the users DOMAIN list, but check the users DOMAIN group list. If the domain is part of a domain group, then the group of domains is effectively sharing whitelist - thus, anything on the whitelist for a user 'steve@acme.com' is effectively on the whitelist for a user 'steve@acme.co.uk', assuming that acme.com and acme.co.uk are part of the same domain group.

Note that BATV addresses need to be normalised, before checking.

```
        select originator from whitelist_<domain>
         where (originator='<orig_address>'
```

or originator='*@<orig_domain>'
or originator='*@*.<orig_domain_without_first _word>'
or originator='*@*.<orig_tld>'
or originator='*@<orig_domain_without_tld>.*')
and (recipient='<recip_address>'
or recipient='*@<recip_domain>')
and (ipaddr='<orig_ip>'
or ipaddr is NULL)
and originator!=''

System blacklist

Uses an identical lookup to whitelist_system, but against the blacklist.

Domain and user blacklist

Domain and user blacklists need to be separated in AWE, as for whitelists. The lookup is the same as for the whitelists, except that blacklists do not contain an IP address.

Exact matching

If an exact match can be found in the database, then this sub code is returned.

However, if an exact match can't be found, other data must be checked to see whether an exact match can still be returned:

SPF records

The originating domains SPF records can be checked to see whether a match is found for the IP address. If so, an exact match can be returned.

In one embodiment, the SPF value passed into the AWE engine as a parameter, to avoid AWE carrying out the check itself.

Partial matching

If a match is not found for the IP address, but there is a match for the originator, then a partial match should be returned when

In order to check for partial matches, the IP address should be checked for the number of octets that match records in the database. In one embodiment this is used in conjunction with the config option, which allows the installer to set this level.

Thus, if the config is set to 2 octet match, the IP address 10.10.23.45 will match 10.10.21.46

Shared matching

If an entry cannot be found for the recipient, then the originator and IP should be checked on other users lists:

```
select from whitelist_<domain> where
        originator = <orig_address>
        and ipaddr = <orig_ip>
```

This is an exact match on both originator and IP address. If this is found then the shared code is returned.

Building whitelists

The AWE engine must build local whitelists based on the queries asked of it.

Outgoing messages

AWE MUST be used on outgoing messages, as well as incoming messages. Every outbound message contains a recipient that should be put on the originators whitelist.

When a query is processed, the message can be determined to be outbound if the originating e-mail address is from one of the local domains, as specified in the config file AND the originating IP address matches those specified for the domain.

A request may also specify directly that the message is outbound, meaning that the above checks are unnecessary.

The outbound message will not contain an originating IP for the new entry however. This is left as NULL unless it can be determined by:

- Checking for other local entries and using the IP addresses present on these.
- Checking the domain SPF records. All IPs present here should be added to the WL entry (or entries)

Incoming messages

Incoming messages can be identified as those messages that are not outbound messages! A request may also specify directly that the message is inbound.

Inbound messages will be checked against all whitelists and may result in new entries being added by:

- Completion of NULL IP address records
- Insertion of new IP records for an existing user by looking up SPF records
- Insertion of new entries as a result of shared or partial matching

User/Admin report

AWE will be capable of sending a daily or weekly report to either users or an administrator.

The report will detail the date, time, originator, recipient and message subject (if available).

The report will enable a user to click through into a GUI that allows them to process these originators to:

- Add to WL
- Add to BL

Configuration file

The configuration file should allow for a single AWE instance to process requests from a number of 'sites'. Each site will have a different set of domains and users.

The configuration file will be in XML format.

The following items should be included in the AWE configuration file, that is read at start-up. There can be a single default section of the configuration file, or multiple sections (named using the site code).

Configuration items are as follows:

General

- license-code = <our license code>
- encryption key
    - Used to de-crypt incoming requests
- cache size
    - Specify cache size used by AWE (if approp)

Logging

- request log location
    - (Must support both windows and Unix path names)
- general log location
- request log style
    - syslog or CSV (d=syslog)
- request log entries
    - A list of API commands. If present, only these actions will be logged

Whitelist checking

- partial-match-level = <num of octets to consider a match>

- list-precedence = global, system_white, system_black, domain_white, domain_black, user_white, user_black

Daemon options

- Listen port
- Bind IP address
- Max threads?

API Specification

External applications and MTAs may integrate with AWE by using the HTTP interface. It will require the requestor to submit the following information:

- originators IP address
- originators email address
- recipients email address
- message direction (inbound or outbound)
- requestors message id (optional)
- message subject (optional)

The interface will respond with:

- The AWE response code
  - One of: USER_WL, DOMAIN_WL, SYSTEM_WL, GLOBAL_WL, USER_BL, DOMAIN_BL, SYSTEM_BL, GLOBAL_BL, UNKNOWN, ERROR
- The AWE sub-code:
  - EXACT (an exact match was found on WL/BL)
  - PARTIAL (a partial match was found for IP)
  - SHARED (a match was found for another domain user)
- The AWE reason text
  - A detailed message explaining why and what the orig/ip matched

The TCP interface may also be extended to allow full Administrative access, such as changing configuration, adding and removing whitelist entries etc.

check_orig

Run an AWE check on the details given as part of the request

list_domaingroups

List domain groups for site

list_domains

List domains for given domain group

list_ips

List IPs for given domain

get_<config item>

The get_ prefix allows the value of any config item to be obtained

set_<config item>

The set_ prefix allows the value of any config item to be set

show_<statistic>

Such as, the show prefix is intended for showing stats values

run_<report>

The run_ prefix is used to run internal reporting operations

wl_<action>

The wl_ prefix is used control whitelist entries. (wl_add/wl_modify/wl_remove/wl_list)

AWE Design notes

AWE is capable of handling a large number of incoming queries, and is able to be split across a number of physical servers.

Fig. 4 shows the basic building blocks of a single node AWE server. In larger deployments, the product can be split across multiple servers, basically down the middle of the diagram, with the HTTP API, FQE, DBM data on one server, and the other components on another server, or more than one server. See later section for more details.

Components

HTTP API

The HTTP API component is responsible for:

- Accepting WL lookup requests from clients
- Accepting WL data that is used to build the WLs
- Accepting and responding to 'admin' requests

In one embodiment, the HTTP API could be split across a number of nodes, so that requestors can make admin requests on one node, and other requests another node.

The HTTP API component controls the other sub components, and returns the responses to the requestor. It runs as a daemon.

A request consists of an HTTP POST, using a URL that contains the request command, with the POST body containing an XML stream containing any parameters required for the request.

Responses are returned as XML in the body of the returned page.

The API controls the calling and responses from the FQE and SQE. It is possible that both the FQE and SQE return multiple values. The API must return a single value to the requestor, so SYSTEM_WL will have the highest precedence.

Data storage

The data for AWE is stored in two places:

- A dbm style database for the FQE
- An sql style database for the SQE

dbm data

The dbm database is a simple key/value pair that contains whitelist and blacklist data. The input files for the creation of the dbm databases consist of keys and values as detailed in the following table:

Type	File	Key	Value
IP WL	ip.dat	IP	IP_WL
System WL	system.dat	orig,recip	SYSTEM_WL
System BL	system.dat	orig,recip	SYSTEM_BL
Domain WL	domain.dat	IP,orig,recip	DOMAIN_WL
Domain BL	domain.dat	IP,orig,recip	DOMAIN_BL
User WL	domain.dat	IP,orig,recip	USER_WL
User BL	domain.dat	IP,orig,recip	USER_BL

Thus, each domain has its own dbm file.

Entries for orig and recip may contain the * character - not as a wildcard, but to denote wildcard type usage:

- Full value: steve@acme.com
- Domain only: *@acme.com

The domain only value arises when a user/admin wishes to accept messages from all users at a domain - this value is entered into the whitelist directly. In order to use this data, it becomes necessary to carry out two searches, steve@acme.com and *@acme.com. If either search fines a matching record, then the WL record matches.

Due to the nature of the application, its possible that there may be many (even hundreds) of domains being handled, and thus hundreds of dbm files.

The FQE must be able to cache the dbm files to ensure that the most frequently used files are held open.

sql data

The SQL database contains data in a similar manner to the dbm files, but is used when more complex queries are carried out, as the full power of a rdb can be used when generating search queries.

The SQL data consists again of one table for the system black/whitelist and one table per domain.

Fast Query Engine (FQE)

The FQE is implemented as a component of the HTTP API.

It is responsible for carrying out fast, exact matches on the static whitelist data, known as 'simple' matching. AWE may be configured to carry out simple matching only in real time, or to carry out both simple and complex matching in real time.

The FQE queries a (semi) static set of whitelist data, held in a dbm format, described above.

The data in these files is basically a key/value pair. The FQE receives the data from the API and carries out up to 6 queries on the underlying data, looking for:

- An exact match for the IP, orig, recip
- A match for a IP match
- IP/orig match
- IP/domain match

So, for a message sent from (1.1.1.1) steve@acme.com → david@bs.com, this data will be processed in order to look up the following keys:

dbm file	lookup key	Expected result
system	1.1.1.1,steve@acme.com	SYSTEM_WL or SYSTEM_BL
system	1.1.1.1,*@acme.com	SYSTEM_WL or SYSTEM_BL
system	*,steve@acme.com	SYSTEM_WL or SYSTEM_BL
system	*,*@steve.com	SYSTEM_WL or SYSTEM_BL
domain	1.1.1.1,steve@acme.com,david@bs.com	DOMAIN_WL/BL or USER_WL/BL
domain	1.1.1.1,*@acme.com,david@bs.com	DOMAIN_WL/BL or USER_WL/BL
domain	1.1.1.1,steve@acme.com,*@bs.com	DOMAIN_WL/BL or USER_WL/BL
domain	1.1.1.1,*@acme.com,*@bs.com	DOMAIN_WL/BL or USER_WL/BL

The result(s) of the lookups are returned to the API for processing and response to the requestor. The lookups may result in more than one result - i.e. a match may be found in the system list as well as the domain list. In which case, the precedence ordering will be decided by the HTTP API component (as this may have waited for a result from the SQE as well.)

The FQE is capable of processing key lookups in parallel. Further, the FQE should handle the absence of a dbm file gracefully - i.e. stop and wait for a period before retrying (this is to cover the time period when a dbm file is being re-written by the updater).

SQE

The Slow Query Engine does 'slow lookups' on the underlying whitelist data. These lookups include:

- Shared list lookups
- Partial IP lookups
- Partial address lookups (including newsletter matching)
- Checks when IPs don't match to use SPF

The SQE is called by the HTTP API process, which can call the SQE in two modes:

- lookup: carry out a lookup and return a response (use data to add to WL if pos.)
- learn: carry out a lookup with the given data; add data to WL if possible. No response is expected.

The SQE carries out a series of lookups and tests according to a rules file. When a match is found, the result is used to add to the WL and is returned to the API component if the request was a lookup.

Message Queue

The SQE operates by monitoring a message queue. The message queue may be spread amongst a number of servers.

Rules file

<TBC>

Updater

The purpose of the updater is to run periodically to extract data from the SQL database and re-create the dbm databases for the FQE. The updater may be on a different server to the FQE.

Request flow

Most requests to the AWE engine will be lookup requests, attempting to find an entry in the WL. A lookup request will contain one or all of:

- Originators IP address
- Originators email address
- Recipients email address

AWE must look up these details in its whitelist tables to determine if it is present, or if a close match exists. The expected program flow will be as follows:

Request flow

1. A request arrives at the HTTP API component. The request will detail whether this is an IP only lookup, or a full lookup, or a learn request (i.e. outbound message)
2. If 'Fast Queries Only' is defined in the configuration, the data will be sent to the FQE and to the SQE as a learn request. Otherwise, the data is sent to both engines.
3. The API waits for responses from the FQE and the SQE, unless Fast Queries Only is set, in which case it only waits for a response from the FQE
4. When the responses are returned, a response is generated for the requestor.

It will be appreciated by persons skilled in the art that numerous variations and/or modifications may be made to the invention as shown in the specific embodiments without departing from the scope of the invention as broadly described. The present embodiments are, therefore, to be considered in all respects as illustrative and not restrictive.

CLAIMS:

1. The steps, features, integers, compositions and/or compounds disclosed herein or indicated in the specification of this application individually or collectively, and any and all combinations of two or more of said steps or features.