DKIM Implementation

MAAWG Training Series

Segment 4 of 4 on DomainKeys Identified Mail

From the onsite training course at the MAAWG 18th General Meeting
San Francisco, February 2010

MAAWG

DKIM Implementation – Video Segments

Segment 1 20 mins.	Segment 2 20 mins.	Segment 3 18 mins.	Segment 4 35 mins.
Theory	Theory	Practical	Practical
• General DKIM Architecture • What DKIM Is and Isn’t	• DKIM Protocol Details • Separate Mail Streams & Signing Practices	• Planning • Keys and Policies	• Signing Software • Verifying Software • Testing, Other Topics • Q&A

Segment 4 Covers

Practical:

- Signing Software
- Verifying Software
- Testing and Other
- Q&A

Murray S. Kucherawy
Principal Engineer
Cloudmark
msk@cloudmark.com

Configuring to Sign Mail

- Consider signing options
  - Set signature expirations?
    - Signature will no longer validate after a specific time has passed
  - Which canonicalizations to use?
    - “relaxed” tolerates minor rewrites such as spacing changes, while “simple” implies maximum strictness
  - Include forensic data?
    - Allows a verifier to see if header fields changed in transit, preventing verification

Configuring to Sign Mail

- Steps specific to *opendkim*
  - Install the filter
  - Select a rendezvous socket
    - Filter will listen for connections from MTAs at the designated socket
    - Security considerations
  - Put private keys someplace safe
    - Filter needs read access to them, but nobody else does
  - Make a list of which keys are used for which users/domains

Configuring to Sign Mail

- Steps specific to *opendkim*
  - Write a configuration file
    - Signing options
    - Domain and key selection rules
    - Auto-restart
    - What socket to use
    - What SMTP clients should have mail signed
  - Start the filter
  - Configure the MTA to connect to the filter and restart it

Configuring to Sign Mail

- Sample `opendkim.conf` contents for signing all of a single domain with one key

```
AlwaysSignHeaders        Subject
AutoRestart              True
Background               True
Canonicalization         relaxed/simple
Diagnostics              Yes
Domain                   example.com
KeyFile                  /var/db/dkim/sign201002.key.pem
InternalHosts            /etc/mail/dkim/internal
LogWhy                   true
Mode                     sign
Selector                 sign201002
SignatureAlgorithm       rsa-sha256
Socket                   inet:8891@localhost
Syslog                   Yes
```

Configuring to Sign Mail

- Sample file contents for signing multiple domains (v2.0.0 or later)
- `/etc/mail/opendkim.conf`:

```
KeyTable                /etc/mail/dkim/keytable
InternalHosts           /etc/mail/dkim/internal
SigningTable            /etc/mail/dkim/signingtable
Socket                  inet:8891@localhost
```

- `/etc/mail/dkim/keytable`

```
opskey  ops.example.com:ops:/etc/mail/dkim/keys/ops
mktgkey mktg.example.com:mktg:/etc/mail/dkim/keys/mktg
execkey exec.example.com:exec:/etc/mail/dkim/keys/exec
preskey exec.example.com:pres:/etc/mail/dkim/keys/president
defkey  example.com:default:/etc/mail/dkim/keys/default
```

- `/etc/mail/dkim/signingtable`

```
ops.example.com            opskey
mktg.example.com           mktgkey
president@exec.example.com preskey
exec.example.com           execkey
.example.com               defkey
```

Configuring to Sign Mail

- A note about OpenDKIM
  - All of the lookup tables referenced in `opendkim.conf` can be:
    - Comma-separated lists
    - Flat files
    - Files matching by regular expressions
    - Sleepycat databases (hash/btree)
    - LDAP directory lookups
    - SQL queries

Configuring to Verify Mail

- Generally you have the following steps:
  - Install your verifying agent (may be an MTA upgrade)
    - Might be the same as the signing agent
  - Tell it which mail to verify
    - Which SMTP clients, which users/domains
    - Might just be “everyone”
  - Select verifying policy options
  - Throw the switch!

Configuring to Verify Mail

- Verification policy options
  - DKIM specifies that an unsigned message and one with a bad signature should be treated the same
  - Any other verification choices are specific to the implementation you use, not to DKIM itself
  - Some common ones are discussed here

Configuring to Verify Mail

- Verification policy options
  - Require certain headers be signed even if absent
    - A favourite is Subject:, since MUAs generally display it
    - Modification or addition both invalidate signatures
  - Require a minimum of additional text when messages are signed with “`l=`“
    - Prevents replay attacks with undesirable appended text

Configuring to Verify Mail

- Verification policy options
  - Do something with "`z=`" (forensics) header fields?
    - Can't do anything other than figure out why a verification failed <u>if</u> it was caused by a header change
  - Authentication-Results: header fields
    - What *authserv-id* to use internally?

Configuring to Verify Mail

- Verification policy options
  - Apply ADSP?
    - Signers might want you to discard mail that's not signed or lacks a valid signature
    - You could end up rejecting/quarantining mail that was accidentally damaged
  - How much clock drift on signatures is allowed?
    - To tolerate misconfigured clocks out there

Configuring to Verify Mail

- Verification policy options
  - Do you want to trust third-party signatures?
    - Again, this is still controversial
    - By default, OpenDKIM only uses author signatures when making final decisions, but you can tell it there are other domains you trust

Configuring to Verify Mail

- Sample `opendkim.conf` contents

```
ClockDrift               300
DiagnosticDirectory      /var/db/dkim/DIAGNOSTICS
DNSTimeout               10
InternalHosts            /etc/mail/dkim/internal
LogWhy                   true
Socket                   inet:8891@localhost
ADSPDiscard              Yes
Syslog                   Yes
Statistics               /var/db/dkim/dkim-stats.db
```

More Complex Policy Options

- Experience has shown that there is a very wide variety of site-specific needs with respect to mail flow and policy enforcement
- Adding features to configuration files to keep up with changing environments is an uphill battle
- As a result, OpenDKIM now (as of v2.0.0) has hooks that allow one to write scripts to enforce policy

More Complex Policy Options

- *Lua* is the scripting language chosen
  - http://www.lua.org
  - Reference books available
- Three scripting entry points
  - *Setup*: Observe properties of message, decide whether to sign (and which key(s) to use), or verify, or both
  - *Screen*: For signed messages, examine the signatures and decide which ones to use and which to discard
  - *Final*: For signed messages, examine the results of processing each signature and decide what to do with the message
- `opendkim` exports message information and a bunch of access and utility functions to the Lua interpreter, then runs your script

More Complex Policy Options

- Sample trivial setup script:

```
-- See if {auth_author} was set
author = odkim.get_mtasymbol(ctx, "{auth_author}")

-- If it's not from an internal source or
-- authenticated, just verify it
if odkim.internal_ip(tx) == 0 and author == nil then
    odkim.verify(ctx)
    return nil
end

-- Since we got this far, we're signing; make a
-- signing request using the key "defkey" from the
-- KeyTable
odkim.sign(ctx, "defkey")

-- That's it!
return nil
```

More Complex Policy Options

- Sample trivial screen script:

```
--  retrieve the count of signatures on the message
nsigs = odkim.get_sigcount(ctx)
if nsigs == nil then
        return nil
end

--  get the From: domain
fdomain = odkim.get_fromdomain(ctx)
if fdomain == nil then
        return nil
end

--  for each signature, ignore it if it's not from the sender's domain
for n = 1, nsigs do
     sig = odkim.get_signhandle(ctx, n)
     sdomain = odkim.sig_getdomain(sig)
     if fdomain ~= sdomain then
         odkim.sig_ignore(sig)
     end
end
```

More Complex Policy Options

- Sample final script

```
 --  retrieve the count of signatures on the message
nsigs = odkim.get_sigcount(ctx)
if nsigs == nil then
        return nil
end

--  If the message had too much stuff added to it (more than 120 bytes)
--  then bounce it
for n = 1, nsigs do
     sig = odkim.get_sighandle(ctx, n)
     bodylen = odkim.sig_bodylength(sig)
     canonlen = odkim.sig_canonlength(sig)
     if bodylen > canonlen + 120 then
         odkim.set_reply(ctx, "554", "",
                         "Too much data after DKIM-protected body")
         odkim.set_result(ctx, SMFIS_REJECT)
     end
end

--  That's it!
return nil
```

Testing Your Setup

- Once configured for signing, send a test message to an autoresponder
  - Check `http://www.dkim.org` for a current list
- Autoresponder will try to validate your message and send it and the results back to you
- The reply will also be signed, so your verifier can take a crack at it
- Of course, if you run two disjoint sites, you can do this yourself

Beyond Basic DKIM

- RFC5451 defines a header field called Authentication-Results that can be used to tell MUAs and other filters what the results of DKIM were
  - Can also be used for SPF, Sender-ID, etc.
- There are some security considerations around using this
  - In particular, dealing with spoofs from outside
  - Read the spec, even if you plan to do this some other way!

Beyond Basic DKIM

- Domain reputation
  - OK, so `example.com`'s signature verified. So what?
    - Spammers can sign their mail just like you can
  - An MUA or filter that considers a verified signature to be ultimate approval is being foolish
    - What if one were to register `marri0tt.com` and send signed phishes? Would the average user be fooled?

Beyond Basic DKIM

- Domain reputation
  - Reputation seeks to associate value with a name
  - In the DKIM world, we would use the domain name found in "`d=`", i.e. the domain that took responsibility for sending the message
  - Likely more value in finding good guys and letting them in than in finding bad guys and keeping them out
    - Bad reputations are very easily shed

Beyond Basic DKIM

- Domain reputation
  - Both commercial and open source efforts are in development
    - OpenDKIM has hooks for one of the open ones, which is still experimental

Beyond Basic DKIM

- Reporting
  - Many sites wish to be advised of unusual activity
    - DKIM failures might indicate phishing or unexpected problems in transit
  - Draft proposal to extend DKIM, ADSP and ARF to publish requests for such advisories
    - Can request reports of incidents such as unsigned messages or failed validations
    - Can request SMTP rejections with specific text, or can request ARF reports

Beyond Basic DKIM

- Doing it on your own
  - The *opendkim* package includes a C library called *libopendkim* that can be used to build your own DKIM-aware applications
  - Includes full HTML documentation

Who’s Doing It Now

- Service providers
  - AOL (verifying)
  - Yahoo! (verifying)
  - Gmail (signing and verifying)
- Popular web sites
  - LinkedIn, Facebook
  - eBay, FTD
- Vendors
  - …just about everyone
  - Several open source implementations

What's Up At MAAWG

- Might want to check out some of the other panels at this conference
  - DKIM and Signing Practices
    - Discusses current DKIM and ADSP-related topics

References

- General Information: `http://www.dkim.org`
- DKIM is defined in RFC4871 (standards track)
  `http://www.ietf.org/rfc/rfc4871.txt`
- Author Domain Signing Practices
  `http://www.ietf.org/rfc/rfc5617.txt`
- Authentication-Results is defined in RFC5451 (standards track)
  `http://www.ietf.org/rfc/rfc5451.txt`
- OpenDKIM
  `http://www.opendkim.org`
- DKIM reporting is currently an IETF individual submission draft
  `http://www.ietf.org/ID.html`
  draft-kucherawy-dkim-reporting
- ARF is now being advanced by the MARF working group
  `http://www.ietf.org/dyn/wg/charter/marf-charter.html`

Questions & Answers

- Now’s the time!