NIST Technical Note 1945

Email Authentication Mechanisms: DMARC, SPF and DKIM

Stephen Nightingale

This publication is available free of charge from:
https://doi.org/10.6028/NIST.TN.1945

National Institute of Standards and Technology
U.S. Department of Commerce

NIST Technical Note 1945

Email Authentication Mechanisms: DMARC, SPF and DKIM

Stephen Nightingale
High Assurance Domains Project
Advanced Network Technology Division
Information Technology Laboratory

This publication is available free of charge from:
https://doi.org/10.6028/NIST.TN.1945

February 2017

National Institute of Standards and Technology
Kent Rochford, Acting NIST Director and Under Secretary of Commerce for Standards and Technology

National Institute of Standards and Technology Technical Note 1945
Natl. Inst. Stand. Technol. Tech. Note 1945, 43 pages (February 2017)
CODEN: NTNOEF

This publication is available free of charge from:
https://doi.org/10.6028/NIST.TN.1945

Abstract

In recent years the Internet Engineering Task Force (IETF) has been making a range of efforts to secure the email infrastructure and its use. Infrastructure protection includes source authentication by RFC 7208 Sender Policy Framework (SPF), message integrity authentication by RFC 6376 Domain Keys Identified Mail (DKIM), and domain owner feedback on the effectiveness of these methods by RFC 7489 Domain-based Message Authentication, Reporting and Conformance (DMARC).

The High Assurance Domains (HAD) secure email project at NIST has been supporting the development of these initiatives by developing and deploying test infrastructure. This report describes our cumulative experiences with a test system for DMARC and its related protocols.

Table of Contents

Tables and Figures

1 Introduction

1.1 Overall Context for this Document

The Information Technology Laboratory at NIST has a long history of research and development in networked email, more recently with a focus on securing email. This includes the development of guidelines, procedural documents, standards documents, infrastructure development, protocol and test implementations, and research results. A recent security guideline is NIST SP 800-45, Version 2 of February 2007, *Guidelines on Electronic Mail Security [SP800-45]*, whose purpose is to recommend security practices for designing, implementing and operating email systems on public and private networks.

The High Assurance Domains project within the Information Technology Laboratory is working to develop, test and help to deploy new network security technologies to aid in building trust in network communications. Some of the specific security technologies the HAD project is working with include email, the Domain Name System [RFC1034][RFC1035] (DNS) and web browsing. Using DNSSEC [RFC4033][RFC4034][RFC4035] to provide integrity and authentication for DNS information, the DNS has an expanded range of uses. This includes using it to store policy information and keying material, to facilitate authentication mechanisms for email.

A recent document that highlights the use of these technologies is NIST SP 800-177 *Trustworthy Email* published in October 2016 [SP800-177]. Recognizing that there is no one single 'big bang' solution for securing email, this document introduces a range of protocol enhancements to SMTP [RFC5321][RFC5322] for authenticating sending domains[1] and protecting email confidentiality. The roots of security for the methods chosen are in securing the DNS, using DNSSEC and public keys stored as PKIX certificates [X.509] for message authentication and end-to-end message encryption. Confidentiality is protected by encryption methods, including Transport Layer Security (TLS) [RFC5246] for encrypting the channel, and either S/MIME [RFC5750][RFC5751] or OpenPGP [RFC4880] for encrypting message content. Authentication of the sending domain occurs through a trilogy of protocols: Sender Policy Framework (SPF) [RFC7208], Domain Keys Identified Mail (DKIM) [RFC6376] and Domain-based Message Authentication, Reporting and Conformance (DMARC) [RFC7489]. NIST SP 800-177 includes recommendations for deploying the above technologies. These

[1] Throughout this document we make a careful distinction between *senders* and *sending domains*. In a secure and authenticated network these would always be expected to be the same. However, one circumstance we are defending against with the technology deployments explained here is that of address spoofing: where a malicious actor at *aggressor@wespoofyou.gud* tries to appear as *victim@example.com*, the actual sender is not the same as the apparent sending domain. In these cases, where *example.com* deploys a DMARC record soliciting aggregate feedback to *feedback@example.com*, actual feedback goes to *example.com* that includes records of messages apparently from *example.com* but actually from *wespoofyou.gud*. In the test system described here, we also try to avoid sending replies to senders who are not authentically representative of the purported sending domain.

recommendations have been developed through practical deployment experience of DMARC, DKIM and SPF, which is the subject of this document.

1.2 What this Document Covers

When the Internet Engineering Task Force (IETF) develops technical responses to problems that have emerged in the Internet, it is informed by analyses of problems that occur, and experimental solutions developed, deployed and critiqued by the internetworking community. This is the essence of the "rough consensus and running code" ethos of the Internet community. Spamming, spoofing and phishing of email have had a long development within the (mal-)practice of internet messaging. Solutions have been also long maturing. Sender Policy Framework (SPF) is one such protocol, intended to allow mail recipients to associate particular sending IP addresses with particular domains. As the protocol matured implementations in various popular languages were also developed. The test system discussed in the balance of this document is written in Python. A Python implementation of SPF was developed by Terrence Way and this has been publicly available for download and experimentation[2]. This is one of the modules employed in the test system developed here.
Domain Keys Identified Mail (DKIM) is intended to be complementary to SPF. Using DKIM with the associated private key, the sender computes a signature over the message and applies it as an SMTP header. The recipient gets the public key from the DNS and verifies the signature, demonstrating that the message has not been modified in transit. Greg Hewgill developed an early Python module dkim.py. William Grant later modified and updated it[3]. This is the DKIM implementation we use in the Pythentic test system.

At the onset of developing this test system, DMARC was a new initiative, aimed at providing aggregate and forensic feedback to sending domains on the total effectiveness of their email authentication strategies. As it was new, we developed a Python module to take in the results from SPF and DKIM processing and compute a deliverability result, to then indicate to Sendmail whether the message must be delivered, discarded or rejected. These SPF, DKIM and DMARC modules have been working in the Pythentic[4] test system since November 2012, approaching 4 years as of the date of publication of this document. We have been accumulating statistics on the disposition of every message received since that time. Here, we explain the test system created and the statistics accumulated in operating the compendium of authentication strategies.

In addition to email message authentication there are other techniques for securing email, which broadly include channel encryption techniques such as Transport Layer Security (TLS) [RFC5246] and message encryption techniques such as S/MIME [RFC5751] and OpenPGP [RFC4880]. These encryption mechanisms are not the subject of this document.

[2] https://pypi.python.org/pypi/pyspf/
[3] https://launchpad.net/dkimpy
[4] Pythentic = Python + authentic. It's the sort of portmanteau word widely used in Python naming conventions.

1.3 Document Structure

Section 2: Email and its Defects: An introduction to the Simple Mail Transfer Protocol as the Internet's email solution, and problems that arise due to its lack of built-in security.

Section 3: DMARC, DKIM and SPF as Remedies: A discussion of the authentication protocols developed to help stem the tide of Spam, Spoofing and Phishing.

Section 4: The Test System and its Components: An architectural description of the actors, systems and modules involved in effecting DMARC, DKIM and SPF testing.

Section 5: The Mechanics of a Typical Test: The exchange of email messages between a correspondent and the Pythentic test system, annotated. A comparison with other message reflectors that perform DMARC authentication.

Section 6: Experience of Use: Analysis of the database records accumulated over 3+ years of test system use.

2 Email and its Defects

The Simple Mail Transfer Protocol (SMTP) [RFC821] [RFC5321] was originally specified in 1982 as a store and forward protocol, where the sending client originates a message, transmits it to a message transfer relay, and forwards it through a series of zero or more further relays, to be delivered by the receiving client. Connections between hops are established by a text based protocol over Telnet [RFC854], which connects using TCP [RFC793].

When originally developed for a relatively small scale academic network, there was no thought given to security of message traffic, either through channel encryption, message encryption, or secure authentication of sender and receiver. Three exploits in particular are very easy to effect under this original scheme:

1. **From Address Spoofing**: Since email headers and content are text lines sent over TCP, it is trivial for the message originator to 'spoof' the from address and fool the recipient into thinking the message came from some domain other than the one properly associated with the source IP address.

2. **Phishing**: With the advent of the World Wide Web it became possible to embed hyperlinks into email body content to direct the activator to phishing sites dedicated to separating you from your money or your sensitive personal information. This is usually done in association with the above mentioned spoofed address. An attacker may spoof his site as AcmeBank.com and fool some recipients into thinking the source of the email and the destination of the embedded link are genuine.

3. **Man in the Middle Message Modification**: When a 'man-in-the-middle' is able to re-route mail in the absence of authentication and encryption, he can freely modify the content of messages transmitted even from bona-fide senders.

While the network remained small scale and collegial, the honor system more or less worked, and nothing large was riding on the outcomes of most messages. With expansion into the commercial domain and multiple countries from the late 1980s, email came to be viewed as mission critical for corporate, government and commercial business. Going into the 1990s, identifiable companies began sending out unsolicited mass email for marketing purposes, and thereby kick-started the spam industries. From this time also, malign actors began sending out phishing emails soliciting for unsuspecting users' personal and financial details. Protective Internet responses were becoming necessary.

What counters are possible for the above three exploits?

1. There needs to be a way for receivers to independently verify the combination of **`From`** address and IP address.

2. Identifiability of the source reduces incentive for phishers to publish malignant links. There is also a continuing campaign to warn users against indiscriminately clicking on links in email messages from unknown or dubious sources.
3. There needs to be a way for receivers to confirm whether email messages received have been modified in transit.[5]

With the development of DNSSEC and other DNS based initiatives, researchers began to look to the Domain Name System itself as a reliable medium for authenticating other domain derived information. Essentially, if a sending domain can post authentication information about itself, then the solutions to the above problems can embrace the DNS. The DMARC, DKIM and SPF protocols deployed in what follows are solutions in that vein.

From the early 2000s SPF was developed in the IETF. Originally designated as 'Sender Permitted From' as a protocol to identify permitted senders from specific domains. The name later settled on Sender Policy Framework [RFC7208]. SPF associates a domain with one of more approved mail senders, and so allows a mail receiver to authenticate the sender. Domain Keys Identified Mail (DKIM) [RFC6376] was developed independently of SPF as a way to allow the sender to sign designated header and body elements of a message, and to associate the message with a specific domain, as identified in the signature, and again through the Domain Name System to enable receivers to retrieve the public key, and authenticate the message on receipt.

While authentication is a good first step, it doesn't prevent a message being altered in transit. Nor does it provide a method of feedback to the originator on the effects of their policies. In the next section, DMARC, DKIM and SPF are considered together as a system for protecting email receivers and sender domains, and returning feedback to sender domains on the effectiveness of their policies.

The list of email defects enumerated here is small, and targeted. A broader picture of the email defects typology is painted in the Trustworthy Email document [SP800-177].

[5] This test system does not test for legitimately forwarded messages.

3 DMARC, DKIM and SPF as Remedies

RFC 5322 defines the Internet Message Format for delivery over the Simple Mail Transfer Protocol (SMTP), but in its original state any sender can write any envelope-From: (sometimes referred to as the "return path") address in the header. This envelope-From: address can however be overridden by malicious senders or enterprise mail administrators at source or en route, who may have organizational reasons to rewrite the header. As a result, both [RFC 5321] and [RFC 5322] defined From: addresses can be aligned to some arbitrary form not intrinsically associated with the originating IP address. In addition, any man in the middle attack can modify header or data content. Beginning in the early 2000s, new protocols have been developed to detect these envelope-From: and message-From: address spoofing or modifications.

The Sender Policy Framework protocol (SPF) [RFC7208] uses the Domain Name System (DNS) to allow domain owners to create records that associate the RFC5321 envelope-From address domain name with one or more IP address blocks used by authorized Mail Sending Agents (MSAs). An example SPF TXT record associated used by the Pythentic protocol tester domain is:

"v=spf1 ip4:129.6.100.200 ip6:2610:20:6005:100::20 -all"

- The first mechanism, **v=spf1** identifies this as an SPF record.
- The second mechanism **ip4:129.6.100.200** says messages originating from the given IPv4 address should be considered valid.
- The third mechanism **ip6:2610:20:6005:100::200** says messages originating from the given IPv6 address should be considered valid.
- The fourth mechanism **-all** says no other address is approved for messages claiming to originate at the given domain.

If a correspondent receives a message from **pythentic@had-pilot.biz**, from IP address 129.6.100.200, the SPF record at had-pilot.biz is extracted and the SPF module compares each mechanism in sequence until a match is found. In this case the mechanism **ip4:129.6.100.200** matches and the message is authenticated. If the message were received from had-pilot.biz with address, say, 1.2.3.4, when each mechanism is tried in turn the **-all** mechanism is the one that matches, and the '**-**' sign says to yield a fail. Some spammers are catching on to the trick of creating SPF record terminated with **+all**. This indicates that every address matches and yields a pass. The observed use of +all and its effects are discussed in the Experience of Use section, later.

The DomainKeys Identified Mail (DKIM) [RFC6376] protocol allows a sending MTA to digitally sign selected headers and the body of a message with a RSA signature and include the signature in a DKIM header attached to the message prior to transmission. The DKIM signature header field includes a selector, which the receiver can use to retrieve the public key from a record in the DNS. This public key is then used to validate the DKIM signature over the message. So, validating the signature assures the receiver that the message has not been modified in transit – other than additional headers added by

MTAs en-route which are ignored during the validation and validates the message sender to the domain publishing the public key. The DKIM record associated with a domain example.com with the selector "mailkey" is stored at **`mailkey._domainkey.example.com`**. The **`mailkey`** label is extracted by the correspondent from the DKIM Signature header on the message. The record is:

`"v=DKIM1; p=<encoded public key>"`

- The first mechanism identifies this as a DKIM record.
- The second mechanism includes the public key that will authenticate signatures originating from our domain.

If a correspondent receives a DKIM signed message purporting to be from example.com, the DKIM Signature is extracted, the `mailkey` selector and DKIM domain are combined, and the DKIM TXT record is read. The DKIM module verifies the signature using the public key extracted, and continues to deliver the message.

Deploying SPF and DKIM may curb illicit activity against a sending domain, but the sender gets no indication of the extent of the beneficial (or otherwise) effects of these policies. Sending domain owners may choose to construct pairwise agreements with selected recipients to manually gather feedback, but this is not a scalable solution. The Domain-based Message Authentication, Reporting and Conformance protocol (DMARC) [RFC7489] institutes such a feedback mechanism, to let sending domain owners know the proportionate effectiveness of their SPF and DKIM policies, and to signal to receivers what action should be taken in attack scenarios. After setting a policy to advise receivers to deliver, quarantine or reject messages that fail SPF and/or DKIM, Email receivers then return DMARC aggregate and/or failure reports of email dispositions to the domain owner, who can review the results and potentially refine the policy.

A sample DMARC TXT record associated with example.com would be published at **`_dmarc.example.com`** in the DNS. The record is:

`"v=DMARC1; adkim=r; aspf=s; p=none; pct=100; rf=afrf; ri=86400; ruf=mailto:forensics.example.com;"`

- The first mechanism identifies this TXT RR as a DMARC record.
- **`adkim=r`** says that the dkim domain is to be evaluated on a relaxed basis, meaning the organizational domain in the record must match that of the RFC5322From domain.
- **`aspf=s`** says the spf domain is to be evaluated on a strict basis, which means that the fully qualified domain name in the record must be an exact match for the RFC5322From domain.
- **`p=none`** says do not apply DMARC policy to messages that fail. These default to local policy, which is in any case identical with DMARC policy.
- **`pct=100`** says apply policy to 100 % of messages received from our domain.
- **`rf=afrf`** says individual forensic reports will be in the AFRF format.

- **`ri=86400`** value is used to requests summary delivery reports on a once daily interval (86,400 seconds = 1 day) from receivers.
- **`ruf=mailto:forensics.example.com`** indicates the address to which forensic reports (i.e. messages that failed DMARC validation) should be sent.

DMARC actually performs up to three complementary tasks:

1. Authenticating messages.
2. Providing aggregate information about messages received from particular domains.
3. Providing instant feedback to sending domains on messages that misuse their domain names.

The pythentic tester does (1) and (3) of these. We do not do (2), as we are only a casual recipient of mail, and not a high volume domain.

While DMARC can do a lot to curb spoofing and phishing, it does need careful configuration. Intermediaries that forward mail have many legitimate reasons to rewrite headers, usually related to legitimate activities such as operating mailing lists, mail groups, and end-user mail forwarding. It should be noted that mail server forwarding changes the source IP address, and without rewriting the envelope-From: field, this can make SPF checks fail. On the other hand, header rewriting, or adding a footer to mail content, may cause the DKIM signature to fail. Both of these interventions can cause problems for DKIM validation and for message delivery.

The challenge is how to test the authentication system built around SMTP and an assemblage of authentication protocols. Since Sendmail[6] is a mature, widely used implementation of SMTP, development of SPF and DKIM checking must be contained separately, but in coordination. The milter protocol[7] and a closely associated milter module hold the key to:

- Deployment of DKIM signing of outgoing messages, and
- SPF, DKIM and DMARC checking of incoming messages.

The test system developed to realize these protocol implementations is described in the following section.

[6] Open Source www.sendmail.com.
[7] See Sendmail documentation.

4 The Test System and its Components

The HAD DMARC test system gives feedback for the SPF, DKIM and DMARC authentication protocols for email messages originating from remote correspondents. The mechanism starts when the correspondent sends a message to **`pythentic@had-pilot.biz`**, the test system retrieves the correspondent's DNS records to help guide policy responses, and replies to the correspondent with an evaluation of the authentication characteristics of the message. To get full use out of it, correspondent domain owners must first configure TXT records for SPF, DKIM and DMARC in the DNS. Messages originated by the test system can have positive or negative anticipated validation results applied. Positive validation results to messages sent to correspondents for delivery include computing a DKIM signature and adding it as an SMTP header. Negative anticipated validation results include corrupting the DKIM signature with a bad hash, or a defective label, or defective folding characters, which should result in DKIM validation failure.

At the test system receiver side, the sendmail milter filters headers and bodies of messages from correspondents, for SPF arguments (IP address, domain, original **`MAIL FROM`**), and the DKIM signature. The milter then retrieves the appropriate DNS records and performs SPF, DKIM and DMARC authentication checks, adds **`X-spf, X-dkim`** and **`X-dmarc`** authentication headers, and writes the results to a database. The milter signals to sendmail whether to deliver, discard or reject the message. Messages marked for delivery are delivered to the Pythentic owner's mailbox. Messages marked for discard are dropped silently. Messages marked for reject are sent back to the originating MTA with the protocol specified SMTP error message.

A DMARC reporter runs in the background and polls the database for updates every two minutes. Newly arrived messages are signaled by a "Reported" field of value 0. After processing, the field is set to 1, to avoid repeated replies. The components of the test system are illustrated in Figure 1 and described below.

4.1 Correspondent

Correspondents learn about the capabilities of the HAD email authenticator from our DMARC testing website[8], and other outreach avenues such as the DMARC.org website[9]. After learning the various test options, the correspondent initiates a message to **`pythentic@had-pilot.biz`** and examines the results in an email reply. It is most useful for the correspondent to deploy records for SPF, DKIM and DMARC in their DNS, so that all authentication protocol deployments can be tested. Messages initiated with **`spf, dkim`**, **`dmarc`** respectively as subject will trigger analyses focusing on those particular results. Messages initiated with the subject **`p.spf.spoof`** triggers a second component to send a spoofed email to pythentic purporting to be from the

[8] https://www.had-pilot.com/py/had.html.
[9] https://dmarc.org/resources/deployment-tools/

correspondent's domain, and a forensic report returned to the correspondent's domain. This is to test and demonstrate the ability of DMARC to detect and report spoofing attempts.

4.2 Domain Name System

Email correspondents store their SPF, DKIM and DMARC authentication records in the DNS so the tester can retrieve and process them. Similarly, the tester stores its authentication records in the DNS so the correspondent can authenticate tester replies. The Pythentic related DNS records are typical for an email server, so we discuss them here. These include an SPF, a DKIM and a DMARC record.

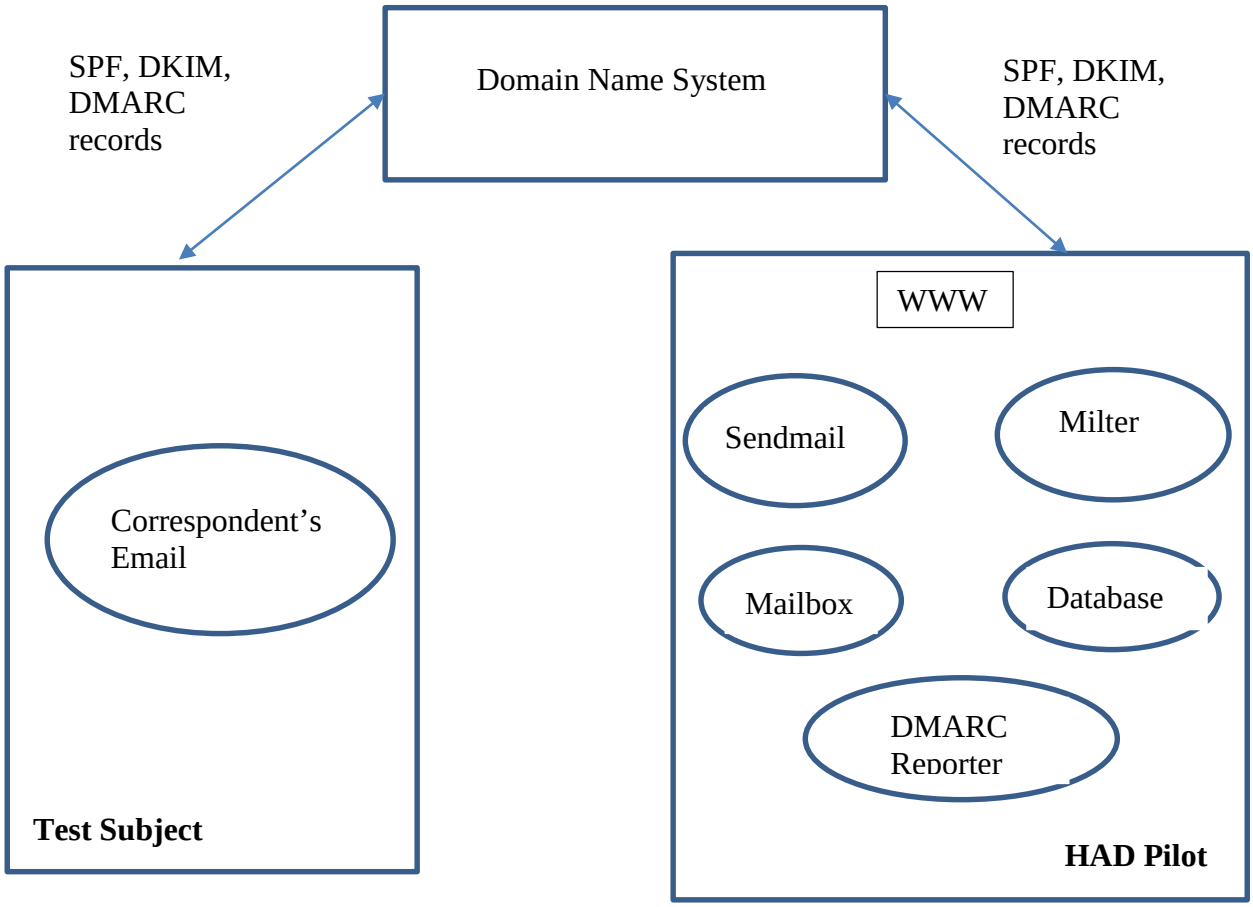


Figure 4-1: DMARC Tester Components

4.3 Web

The High Assurance Domains project maintains a webpage[10] that instructs correspondents to send email to *pythentic@had-pilot.biz* and gives a selection of tests, and their intended outcomes. The static page links to an HTML form that allows individual recipients to paste in information allowing them to review results of completed tests. The form is linked to a Python CGI process at the had-pilot site.

4.4 Sendmail

Sendmail is the main engine by which the test system exchanges mail with correspondents. There is a number of specific configurations required to tailor send, receive, header writing and evaluation policies. Principal among these is configuring mail filters, or milters. Sent mail has to be modified to add a DKIM signature. Received mail has to be authenticated for SPF, DKIM and DMARC policies to determine its deliverability. These modifications are performed in a sending milter[11] and a receiving milter respectively. Sendmail is also configured to set the domain name, allowing for a spoofed mail test.

4.5 Sending Milter

The HAD pythentic milter is an adaptation of the pure python ppymilter.py[12]. It implements the Sendmail milter protocol, a command and response protocol controlled by the MTA (Sendmail) for exchanging option negotiation, **`HELO`** information, SMTP headers, SMTP body parts, and Deliver/Discard/Reject advices. While Sendmail acts as a pure relay, and the milter protocol is not sensitive to whether the message source is 'local' or 'remote', in practice two separate milters are deployed, a sending milter to handle outgoing messages, identified with IP address 127.0.0.1 (the 'local' address), and a receiving milter that handles incoming messages, and is identified by any non-local IP address.

The sole purpose of the outgoing milter for the test system is to add DKIM Signatures to outgoing messages. So the milter captures all SMTP headers and body parts, and employs the DKIM module to create a signature, using the private key stored locally (the public key is in the DKIM record in the DNS). This signature is incorporated in a DKIM Signature SMTP header and sent back to Sendmail for incorporation into the outgoing message to the correspondent. The system includes tests for defective DKIM signatures and headers. These are signaled by subject of **`p.dkim.bad`**, **`p.dkim.bh`**,

[10] https://www.had-pilot.com/py/had.html. The image of this page is depicted in Appendix 1.
[11] A milter is a mail filter, acting in close cooperation with a Mail Transfer Agent to modify and/or authenticate messages received. See https://en.m.wikipedia.org/wiki/Milter.
[12] Originally written by Eric de Friez at Google, but later forked and transferred to GitHub. https://www.github.com/jmehnle/ppymilter.

p.dkim.nolf, **p.dkim.nocr**, and the specific effects are created by the milter modifying the DKIM signature header before passing it to Sendmail.

4.6 Receiving Milter

The purpose of the receiving milter is to authenticate the received message with SPF, DKIM and DMARC modules, create a database record, and signal to Sendmail whether the message is to be delivered, discarded or rejected. All SMTP headers and body parts are filtered and considered in turn. When the milter macro callback is triggered on connection prior to any SMTP headers, the IP address, envelope From domain and the full mail address are extracted.

The SPF module gets the correspondent's SPF record from the DNS and authenticates the envelope From and IP addresses against the sequence of mechanisms in the SPF record. If the first mechanism to match is an address, the message authenticates. If none of the mechanisms match, SPF authentication fails. DKIM Signature authentication proceeds after storing the DKIM Signature header, receiving the complete message and body, and getting the DKIM DNS record with the correspondent's public key. The DKIM module verifies the signature over the message using the public key, and returns a pass or fail verdict. The test system augments this module with diagnostic information, that is passed back to the milter to be saved in the database.

After the milter receives and processes the complete message, the correspondent's DMARC record is queried and, if found, the message is authenticated according to the policies encoded in it. The DMARC module assigns a delivery disposition and adds diagnostic instrumentation for the milter to save in the database entry. The milter generates authentication headers for the SPF, DKIM and DMARC results, hands them to Sendmail for delivery to the Pythentic mailbox, and concludes by writing the message, DNS records and result data to the database. Examples of the **X-spf**, **X-dkim** and **X-dmarc** authentication headers are included in the next section, in the Mechanics of a Typical Test.

4.7 DMARC Reporter

All of the delivery analysis for a received message is done in the receiving milter and written to the database. Subsequent processing of the database records written by the milter is left to a DMARC reporter cron job that wakes up every two minutes to process all as yet unreported records in the database. These are processed in four broad categories:

1) Register messages: messages received with subject 'register' cause the DMARC Reporter to send a 'register' message back to the correspondent with a uniquely

identifying hash. This information allows the correspondent to view their accumulated results from the HTML form page.[13]

2) Bad DMARC Tests: Messages received with subjects: **`p.dkim.bad`**, **`p.dkim.bh`**, **`p.dkim.nolf`**, **`p.dkim.nocr`** cause the DMARC reporter to generate DKIM signatures with intentionally bad values:
    a. A bad hash header type,
    b. A bad body hash,
    c. Bad signature folding with omitted carriage-return or line-feed.

   These are for the correspondent's DKIM validation module to analyze and dispose. The test system cannot, of course, know the ultimate outcomes of these tests.

   Messages received with the subject **`p.spf.spoof`** enable the dmarcreporter to initiate a spoof test, and generate a forensic report. On processing this message, the DMARC reporter sends a call to a spoofing agent, which initiates a spoofed message to pythentic@had-pilot.biz using the **`p.spf.spoof`** message initiator's domain. Pythentic rejects the message and writes it to the database. Pythentic Sendmail may also initiate a 500 series[14] rejection message aimed at the domain owner. If the correspondent domain has a DMARC record with a **`ruf:`** address set, the DMARC reporter sends a forensic report to the domain owner (and initiator of the **`p.spf.spoof`** message), describing the received spoof.
3) Test Messages: Messages with a valid test subject (e.g. **`spf`**, **`dkim`**, **`dmarc`**, test) cause the DMARC reporter to send a reply to the originating domain containing the test details that were written to the database. These are:
    a. **`spf`**: the reply contains the results of SPF processing, including the SPF disposition, delivery result, full message received, and correspondent's SPF record, or full recursive list of records with SPF **`include`** and/or **`redirect`** mechanisms.
    b. **`dkim`**: the reply contains the results of DKIM processing, including the DKIM result, delivery result, DKIM DNS record and the full message.
    c. **`Dmarc`**: the reply includes both the SPF and DKIM result, if relevant, as well as the DMARC result and record, and the full message.
    d. **`test`**: The same results as a **`dmarc`** subject.
    e. **`forensic`**: messages received with this subject are treated the same as a **`p.spf.spoof`** test, and precipitate a forensic report back to the correspondent, if a **`ruf:`** address is set in the DMARC record.
4) Spam and Spoofed Messages: Messages with a non-test subject that result in a DMARC Discard or reject do not generate a test reply, as they are deemed to be spam, or spoofed messages.

[13] https://www.had-pilot.com/dmarcresults.html.
[14] All of the SMTP status codes are listed at https://www.ietf.org/assignments/smtp-enhanced-status-codes/smtp-enhanced-status-codes.xml.

5 The Mechanics of a Typical Test

In addition to the Pythentic test system for DMARC, DKIM and SPF which is the subject of this technical note, the dmarc.org website identifies a number of other message authenticators, from Port25, ReturnPath, and UnlockTheInbox. The mechanics of a test with pythentic are described in Section 5.1. The additional authenticators are compared in Section 5.2.

5.1 Mechanics

The inset box below with italicized text includes an exchange of two messages, from an example Pythentic client to the Pythentic test responder. These represent a completed DMARC test. Descriptive annotations in **bold text** are interspersed with the *italicized message text.* First some preliminary description:

- **The correspondent domain is gmail.com and the correspondent is pythentic.client@gmail.com.**
- **Before any message is initiated the correspondent domain publishes to the DNS and SPF record, a DKIM record and a DMARC record. The SPF record in this example test is recursively nested, with redirect and include elements leading to additional DNS entries.**
  - v=spf1 redirect=_spf.google.com
  - v=spf1 include:_netblocks.google.com include:_netblocks2.google.cominclude:_netblocks3.google.com ~all
  - v=spf1 ip4:64.18.0.0/20 ip4:64.233.160.0/19 ip4:66.102.0.0/20 ip4:66.249.80.0/20 ip4:72.14.192.0/18 ip4:74.125.0.0/16 ip4:108.177.8.0/21 ip4:173.194.0.0/16 ip4:207.126.144.0/20ip4:209.85.128.0/17 ip4:216.58.192.0/19 ip4:216.239.32.0/19 ~all
  - v=spf1 ip6:2001:4860:4000::/36 ip6:2404:6800:4000::/36 ip6:2607:f8b0:4000::/36 ip6:2800:3f0:4000::/36 ip6:2a00:1450:4000::/36 ip6:2c0f:fb50:4000::/36 ~all
  - v=spf1 ip4:172.217.0.0/19 ~all
- **The DKIM record simply contains the key type and public key:**
  - *k=rsa; p=MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA1Kd87/UeJjenpabgbFwh+eBCsSTrqmwIYYvywlbhbqoo2DymndFkbjOVIPIldNs/m40KF+yzMn1skyoxcTUGCQs8g3FgD2Ap3ZB5DekAo5wMmk4wimDO+U8QzI3SD07y2+07wlNWwIt8svnxgdxGkVbbhzY8i+RQ9DpSVpPbF7ykQxtKXkv/ahW3KjViiAH+ghvvIhkx4xYSIc9oSwVmAl5OctMEeWUwg8Istjqz8BZeTWbf41fbNhte7Y+YqZOwq1Sd0DbvYAD9NOZK9vIfuac0598HY+vtSBczUiKERHv1yRbcaQtZFh5wtiRrN04BLUTD21MycBX5jYchHjPY/wIDAQAB*
- **The DMARC record*:***
  - *v=DMARC1; p=none; rua=mailto:mailauth-reports@google.com*
- **These records guide the receiver's processing and delivery disposition of the received message.**
- **The initial message is from pythentic.client@gmail.com to pythentic@had-pilot.biz, with subject dmarc. The version with headers fully enumerated is given in the return reply, below.**

Pythentic Client *<pythentic.client@gmail.com>* *Mon, Jun 20, 2016 at 3:09 PM*
To: pythentic@had-pilot.biz

A sample DMARC reflector test.

- **The mail system comprising a series of Mail Transfer Agents forwards the message to had-pilot.biz and it is received by the Sendmail MTA.**
- **Sendmail uses the Milter protocol to send the envelope data and message header by header and body part by body part to the receiving Milter.**
- **From the envelope data, the Milter saves the originators IP address and mailbox address. SPF is authenticated. The DKIM header is saved for later authentication.**
- **With SPF and DKIM results in hand, the Milter gets the DMARC record, noting that SPF and DKIM domain restrictions (aspf, adkim) are not specified, and uses the defaults of aspf=relaxed, adkim=relaxed, and there are no subdomain restrictions. It determines that since SPF passes and DKIM passes, the message can be marked for delivery.**
- **When all headers and body parts are assembled, the Milter gets the originator's DKIM record with the public key and authenticates the message.**
- **The Milter adds three additional SMTP headers: X-SPF, X-DKIM and X-DMARC to record the authentication results, adding in also the DNS records, and returns these headers to Sendmail. The database record is written with "reported=0" value.**
- **Sendmail delivers the augmented message to the Pythentic mailbox.**
- **The dmarcreporter process polls the database every two minutes. All records with "reported=0" are processed, and replied to or ignored according to disposition.**
- **The database record for the message from pythentic.client and its DNS records are processed. This message contains a standard test subject ('dmarc'), and is marked to Deliver, so a test feedback report is prepared. The feedback report is included in the inset box below.**

NIST HAD Email Authenticator *<pythentic@had-pilot.biz>* *Mon, Jun 20, 2016 at 4:04 PM*

To: pythentic.client@gmail.com

```
================================================================
==================================
Testing for: dmarc
================================================================
```

```
==================================
```

- **The summary of results including datetime, subject, from address, spf, dkim and dmarc dispositions, is added, here**:

```
Summary of results:
[4461]:  Jun 20 15:09:46
2016   subj:dmarc, from:pythentic.client@gmail.com (209.85.161
.194), spf:pass, dkim:1, dmarc:Deliver, rep=0

==================================================================
==================================
```

- **The recursive set of SPF records is processed. The redirect leads to includes, which lead to the netblocks.* record. This is processed mechanism by mechanism. The ip4 mechanisms are matched in sequence, each one failing until ip4:209.85.128.0/17 is found as a match for the source address 209.85.161.194. Thus, an SPF pass is recorded for this message.**

```
SPF Analysis:
        result: pass
        Reason:
        SPF Match: (209.85.161.194 in ip4:209.85.128.0/17).
        SPFRecords:     gmail.com == 'v=spf1
redirect=_spf.google.com'
        _spf.google.com == 'v=spf1
include:_netblocks.google.com include:_netblocks2.google.com i
nclude:_netblocks3.google.com ~all'
        _netblocks.google.com == 'v=spf1
ip4:64.18.0.0/20 ip4:64.233.160.0/19 ip4:66.102.0.0/20 ip4:66.
249.80.0/20 ip4:72.14.192.0/18 ip4:74.125.0.0/16 ip4:108.177.8
.0/21 ip4:173.194.0.0/16 ip4:207.126.144.0/20ip4:209.85.128.0/
17 ip4:216.58.192.0/19 ip4:216.239.32.0/19 ~all'
        _netblocks2.google.com == 'v=spf1
ip6:2001:4860:4000::/36 ip6:2404:6800:4000::/36
ip6:2607:f8b0:4000::/36 ip6:2800:3f0:4000::/36
ip6:2a00:1450:4000::/36 ip6:2c0f:fb50:4000::/36 ~all'
        _netblocks3.google.com == 'v=spf1
ip4:172.217.0.0/19 ~all'

InterimResults:
        Syntax Results for: gmail.com Good Syntax.
        Syntax Results for: _spf.google.com Good Syntax.
        Syntax Results for: _netblocks.google.com Good Syntax.
        Syntax Results for: _netblocks2.google.com Good
Syntax.
```

```
        Syntax Results for: _netblocks3.google.com Good
Syntax.


        No Match: (209.85.161.194 in ip4:64.18.0.0/20)
        No Match: (209.85.161.194 in ip4:64.233.160.0/19)
        No Match: (209.85.161.194 in ip4:66.102.0.0/20)
        No Match: (209.85.161.194 in ip4:66.249.80.0/20)
        No Match: (209.85.161.194 in ip4:72.14.192.0/18)
        No Match: (209.85.161.194 in ip4:74.125.0.0/16)
        No Match: (209.85.161.194 in ip4:108.177.8.0/21)
        No Match: (209.85.161.194 in ip4:173.194.0.0/16)
        No Match: (209.85.161.194 in ip4:207.126.144.0/20)

        SPF Match: (209.85.161.194 in ip4:209.85.128.0/17).

        Interim Result for _netblocks.google.com: pass,
Reason:
        SPF Match: (209.85.161.194 in ip4:209.85.128.0/17).
```

- **The DKIM DNS record is extracted and the DKIM signatures in the message extracted. The X-Google-DKIM-Signature seems to be for internal use only.**
- **The remaining DKIM Signautre is tested for the body hash, this matches, and it is verified against the DKIM algorithm, and it verifies. Thus a DKIM Pass is recorded for this message.**

```
DKIM Analysis:
        result: True
        Reason: DKIM Pass.
        DKIM Record: k=rsa;
p=MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA1Kd87/UeJjenpabg
bFwh+eBCsSTrqmwIYYvywlbhbqoo2DymndFkbjOVIPIldNs/m40KF+yzMn1sky
oxcTUGCQs8g3FgD2Ap3ZB5DekAo5wMmk4wimDO+U8QzI3SD07y2+07wlNWwIt8
svnxgdxGkVbbhzY8i+RQ9DpSVpPbF7ykQxtKXkv/ahW3KjViiAH+ghvvIhkx4x
YSIc9oSwVmAl5OctMEeWUwg8Istjqz8BZeTWbf41fbNhte7Y+YqZOwq1Sd0Dbv
YAD9NOZK9vlfuac0598HY+vtSBczUiKERHv1yRbcaQtZFh5wtiRrN04BLUTD21
MycBX5jYchHjPY/wIDAQAB
        Explanation:

DKIM Signatures in the message:
[0] DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed;
        d=gmail.com; s=20120113;
        h=mime-version:from:date:message-id:subject:to;
        bh=NEfHE3dmEz5MON7/kUoUk6Wq1Batfpd9JxvcnY1/dAw=;
        b=qJh18V/1WzeKC9ePo9/lhxa0rM2Zi/PTjHQNw3lj/Hi30dT9MrZT
f2sYW2JmaaHvfX
         hi0qSm31voTlHQEc34uQa/uG6hO2WhUJgd5oNvHI768oybYR7Gs0T
5txXqFDdmCYgpCs
         YCqhspHmPNlm9eGj+L7M3W/5Ybi2sgb2c6mVV0ylZ83Loqe0v3pkT
```

```
5FInjrdwNuzvU8r
         TuSgTfVbcsGojVfDxA3gyLC0NDX7X+m2dKWSCGqg+iyh1Q4jE1RNV
fEj9uaDZo8H+Vlo
         tdr9YGgUZ575NSudzt8k1w+d1Q/LWbAk4Ivm8R9dHgxkYp7lWM/Pz
/edEef8N4VGyOwa
         wJqg==

[1] X-Google-DKIM-Signature: v=1; a=rsa-sha256;
c=relaxed/relaxed;
        d=1e100.net; s=20130820;
        h=x-gm-message-state:mime-version:from:date:message-
id:subject:to;
        bh=NEfHE3dmEz5MON7/kUoUk6Wq1Batfpd9JxvcnY1/dAw=;
        b=OfwgQkJI2znxoqEgC+0G4ef67C2F7bS4pWnv7t2Rfsho5m37GirY
lbJWboLtsq44iF
         SN4332WRQ+0gB2L3w+UYT0arK96Qg+t0sDhLFG9m2gDyPCt8d2kZl
l8wRq01xHnK6SSw
         uXp3VVxU53gKPIlvv+M0ap/remjpvN7I5/kZoqfQWyZ2R7lv4XOy6
DOPmZRwPp8D1oDw
         2XdJC7wQXMlFfLscWspO+0mjiFZ+ZqX7zooLHYQVktuajrhp/AmaF
a0k7gqa8xZbPtuX
         A9IoGVTMzHkYBvQvZRj6Y5EuGh3wlhKieJuDekNr9ijEmlo5MQEuK
AjXXNfIKP3BnHkS
         4jcg==

BH field in
signature:   NEfHE3dmEz5MON7/kUoUk6Wq1Batfpd9JxvcnY1/dAw=
Computed
BodyHash:       NEfHE3dmEz5MON7/kUoUk6Wq1Batfpd9JxvcnY1/dAw=
Computed Hash matches Message Hash.

Signature:
qJh18V/1WzeKC9ePo9/lhxa0rM2Zi/PTjHQNw3lj/Hi30dT9MrZTf2sYW2Jmaa
HvfX
         hi0qSm31voTlHQEc34uQa/uG6hO2WhUJgd5oNvHI768oybYR7Gs0T
5txXqFDdmCYgpCs
         YCqhspHmPNlm9eGj+L7M3W/5Ybi2sgb2c6mVV0ylZ83Loqe0v3pkT
5FInjrdwNuzvU8r
         TuSgTfVbcsGojVfDxA3gyLC0NDX7X+m2dKWSCGqg+iyh1Q4jE1RNV
fEj9uaDZo8H+Vlo
         tdr9YGgUZ575NSudzt8k1w+d1Q/LWbAk4Ivm8R9dHgxkYp7lWM/Pz
/edEef8N4VGyOwa
         wJqg==

DKIM Signature Verifies.


================================================================
==========================================
```

- **The results reported by the Pythentic Milter and recorded in SMTP headers are included here, for corroboration.**

SPF, DKIM and DMARC results are computed twice: once by the incoming mail milter, and again by the outgoing test responder. Both sets of results are given below. They should be the same.

Results reported by the Pythentic milter:

X-dkim: d=gmail.com, s=20120113, DKIMReason=DKIM Pass., DKIMrecord=k=rsa; p=MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA1Kd87/UeJjenpabgbFwh+eBCsSTrqmwIYYvywlbhbqoo2DymndFkbjOVIPIldNs/m40KF+yzMn1skyoxcTUGCQs8g3FgD2Ap3ZB5DekAo5wMmk4wimDO+U8QzI3SD07y2+07wlNWwIt8svnxgdxGkVbbhzY8i+RQ9DpSVpPbF7ykQxtKXkv/ahW3KjViiAH+ghvvIhkx4xYSIc9oSwVmAl5OctMEeWUwg8Istjqz8BZeTWbf41fbNhte7Y+YqZOwq1Sd0DbvYAD9NOZK9vlfuac0598HY+vtSBczUiKERHv1yRbcaQtZFh5wtiRrN04BLUTD21MycBX5jYchHjPY/wIDAQAB

X-spf: i=209.85.161.194, h=gmail.com., s=pythentic.client@gmail.com, SPFResult=pass, SPFrecord=v=spf1 redirect=_spf.google.com#v=spf1 include:_netblocks.google.com include:_netblocks2.google.cominclude:_netblocks3.google.com ~all#v=spf1 ip4:64.18.0.0/20 ip4:64.233.160.0/19 ip4:66.102.0.0/20 ip4:66.249.80.0/20 ip4:72.14.192.0/18 ip4:74.125.0.0/16 ip4:108.177.8.0/21 ip4:173.194.0.0/16 ip4:207.126.144.0/20ip4:209.85.128.0/17 ip4:216.58.192.0/19 ip4:216.239.32.0/19 ~all#v=spf1 ip6:2001:4860:4000::/36 ip6:2404:6800:4000::/36 ip6:2607:f8b0:4000::/36 ip6:2800:3f0:4000::/36 ip6:2a00:1450:4000::/36 ip6:2c0f:fb50:4000::/36 ~all#v=spf1 ip4:172.217.0.0/19 ~all

X-dmarc: result=Deliver, DMARCAction=Applying Local Policy because DMARC record exists and is good but DMARC policy equals 'none'.: SPF passed so DMARC Authenticates., DMARCrecord=v=DMARC1; p=none; rua=mailto:mailauth-reports@google.com

==========

- **The DMARC delivery disposition is developed based on the DMARC record and the SPF and DKIM results. The outcome for this message is to Deliver it.**

Results generated by the DMARCreporter:

Intermediate Results for: _dmarc.gmail.com
v=DMARC1 : good.

```
p=none : good.
rua=mailto:mailauth-reports@google.com : good.
NOTE: relaxed alignment assumed for adkim.
NOTE: relaxed alignment assumed for aspf.


Applying Local Policy because DMARC record exists and is good
but DMARC policy equals 'none'.: SPF passed so DMARC
Authenticates.
DMARC Result: Deliver, Reason: Applying Local Policy because
DMARC record exists and is good but DMARC policy equals
'none'.: SPF passed so DMARC Authenticates.
Record: v=DMARC1; p=none; rua=mailto:mailauth-
reports@google.com

================================================================
====================================
```

- **The full database record is included in this mesage, with the full message received.**

```
 Full Message record:

Record [4461]:
  DeliveryResult: Deliver

Results and Reasons:
    SPF Result = pass, Reason =
        SPF Match: (209.85.161.194 in ip4:209.85.128.0/17).
    DKIM Result = 1, Reason = DKIM Pass.
    DMARC Result = Deliver, Reason = Applying Local Policy
because DMARC record exists and is good but DMARC policy
equals 'none'.: SPF passed so DMARC Authenticates.

DNS Records:
    SPF Record:
        v=spf1 redirect=_spf.google.com
        v=spf1
include:_netblocks.google.com include:_netblocks2.google.com i
nclude:_netblocks3.google.com ~all
        v=spf1
ip4:64.18.0.0/20 ip4:64.233.160.0/19 ip4:66.102.0.0/20 ip4:66.
249.80.0/20 ip4:72.14.192.0/18 ip4:74.125.0.0/16 ip4:108.177.8
.0/21 ip4:173.194.0.0/16 ip4:207.126.144.0/20 ip4:209.85.128.0
/17 ip4:216.58.192.0/19ip4:216.239.32.0/19 ~all
        v=spf1 ip6:2001:4860:4000::/36 ip6:2404:6800:4000::/36
ip6:2607:f8b0:4000::/36 ip6:2800:3f0:4000::/36
ip6:2a00:1450:4000::/36 ip6:2c0f:fb50:4000::/36 ~all
        v=spf1 ip4:172.217.0.0/19 ~all
```

```
    DKIM Record:
        k=rsa;
        p=MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA1Kd87/Ue
JjenpabgbFwh+eBCsSTrqmwIYYvywlbhbqoo2DymndFkbjOVIPIldNs/m40KF+
yzMn1skyoxcTUGCQs8g3FgD2Ap3ZB5DekAo5wMmk4wimDO+U8QzI3SD07y2+07
wlNWwIt8svnxgdxGkVbbhzY8i+RQ9DpSVpPbF7ykQxtKXkv/ahW3KjViiAH+gh
vvIhkx4xYSIc9oSwVmAl5OctMEeWUwg8Istjqz8BZeTWbf41fbNhte7Y+YqZOw
q1Sd0DbvYAD9NOZK9vlfuac0598HY+vtSBczUiKERHv1yRbcaQtZFh5wtiRrN0
4BLUTD21MycBX5jYchHjPY/wIDAQAB
    DMARC Record:
        v=DMARC1; p=none; rua=mailto:mailauth-
reports@google.com

Original Message:
Received: by mail-yw0-f194.google.com with SMTP id
v77so4163674ywg.2
        for <pythentic@had-pilot.biz>; Mon, 20 Jun 2016
12:09:45 -0700 (PDT)
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed;
        d=gmail.com; s=20120113;
        h=mime-version:from:date:message-id:subject:to;
        bh=NEfHE3dmEz5MON7/kUoUk6Wq1Batfpd9JxvcnY1/dAw=;
        b=qJh18V/1WzeKC9ePo9/lhxa0rM2Zi/PTjHQNw3lj/Hi30dT9MrZT
f2sYW2JmaaHvfX
         hi0qSm31voTlHQEc34uQa/uG6hO2WhUJgd5oNvHI768oybYR7Gs0T
5txXqFDdmCYgpCs
         YCqhspHmPNlm9eGj+L7M3W/5Ybi2sgb2c6mVV0ylZ83Loqe0v3pkT
5FInjrdwNuzvU8r
         TuSgTfVbcsGojVfDxA3gyLC0NDX7X+m2dKWSCGqg+iyh1Q4jE1RNV
fEj9uaDZo8H+Vlo
         tdr9YGgUZ575NSudzt8k1w+d1Q/LWbAk4Ivm8R9dHgxkYp7lWM/Pz
/edEef8N4VGyOwa
         wJqg==
X-Google-DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed;
        d=1e100.net; s=20130820;
        h=x-gm-message-state:mime-version:from:date:message-
id:subject:to;
        bh=NEfHE3dmEz5MON7/kUoUk6Wq1Batfpd9JxvcnY1/dAw=;
        b=OfwgQkJI2znxoqEgC+0G4ef67C2F7bS4pWnv7t2Rfsho5m37GirY
lbJWboLtsq44iF
         SN4332WRQ+0gB2L3w+UYT0arK96Qg+t0sDhLFG9m2gDyPCt8d2kZl
l8wRq01xHnK6SSw
         uXp3VVxU53gKPIlvv+M0ap/remjpvN7I5/kZoqfQWyZ2R7lv4XOy6
DOPmZRwPp8D1oDw
         2XdJC7wQXMlFfLscWspO+0mjiFZ+ZqX7zooLHYQVktuajrhp/AmaF
a0k7gqa8xZbPtuX
         A9IoGVTMzHkYBvQvZRj6Y5EuGh3wlhKieJuDekNr9ijEmlo5MQEuK
AjXXNfIKP3BnHkS
         4jcg==
X-Gm-Message-State:
```

```
ALyK8tLR5dmHlq1hU3oSFRKsmW4Gh0ki60tokHygsPSnmVIDb3ttPUP7D92RnJ
ROJsBSIHax3oqkKfLGTN33jQ==
X-Received: by 10.37.216.19 with SMTP id
p19mr7618243ybg.78.1466449783556;
 Mon, 20 Jun 2016 12:09:43 -0700 (PDT)
MIME-Version: 1.0
Received: by 10.129.16.206 with HTTP; Mon, 20 Jun 2016
12:09:43 -0700 (PDT)
From: Pythentic Client <pythentic.client@gmail.com>
Date: Mon, 20 Jun 2016 15:09:43 -0400
Message-ID:
<CAF4Hq2BZ37hWU5tdumDJXX5Gm7bQ_9enBLfo9HNDwv17Di2HyA@mail.gmai
l.com>
Subject: dmarc
To: pythentic@had-pilot.biz
Content-Type: multipart/alternative;
boundary=94eb2c06afc06e0cc10535ba724a
X-dkim: d=gmail.com, s=20120113, DKIMReason=DKIM Pass.,
DKIMrecord=k=rsa;
p=MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA1Kd87/UeJjenpabg
bFwh+eBCsSTrqmwIYYvywlbhbqoo2DymndFkbjOVIPIldNs/m40KF+yzMn1sky
oxcTUGCQs8g3FgD2Ap3ZB5DekAo5wMmk4wimDO+U8QzI3SD07y2+07wlNWwIt8
svnxgdxGkVbbhzY8i+RQ9DpSVpPbF7ykQxtKXkv/ahW3KjViiAH+ghvvIhkx4x
YSIc9oSwVmAl5OctMEeWUwg8Istjqz8BZeTWbf41fbNhte7Y+YqZOwq1Sd0Dbv
YAD9NOZK9vlfuac0598HY+vtSBczUiKERHv1yRbcaQtZFh5wtiRrN04BLUTD21
MycBX5jYchHjPY/wIDAQAB
X-spf: i=209.85.161.194, h=gmail.com.,
s=pythentic.client@gmail.com, SPFResult=pass, SPFrecord=v=spf1
redirect=_spf.google.com#v=spf1 include:_netblocks.google.com
include:_netblocks2.google.cominclude:_netblocks3.google.com ~
all#v=spf1
ip4:64.18.0.0/20 ip4:64.233.160.0/19 ip4:66.102.0.0/20 ip4:66.
249.80.0/20 ip4:72.14.192.0/18 ip4:74.125.0.0/16 ip4:108.177.8
.0/21 ip4:173.194.0.0/16 ip4:207.126.144.0/20ip4:209.85.128.0/
17 ip4:216.58.192.0/19 ip4:216.239.32.0/19 ~all#v=spf1
ip6:2001:4860:4000::/36 ip6:2404:6800:4000::/36
ip6:2607:f8b0:4000::/36 ip6:2800:3f0:4000::/36
ip6:2a00:1450:4000::/36 ip6:2c0f:fb50:4000::/36 ~all#v=spf1
ip4:172.217.0.0/19 ~all
X-dmarc: result=Deliver, DMARCAction=Applying Local Policy
because DMARC record exists and is good but DMARC policy
equals 'none'.: SPF passed so DMARC Authenticates.,
DMARCrecord=v=DMARC1; p=none; rua=mailto:mailauth-
reports@google.com

--94eb2c06afc06e0cc10535ba724a
Content-Type: text/plain; charset=UTF-8

A sample DMARC reflector test.
```

```
--94eb2c06afc06e0cc10535ba724a
Content-Type: text/html; charset=UTF-8

<div dir="ltr">A sample DMARC reflector test.</div>

--94eb2c06afc06e0cc10535ba724a--

================================================================
====================================
```

- **The registration information including the Paste-in-Hash is added. This allows the correspondent to view historical testing records from the Pythentic database.**

```
 Registration Info:
Thank you for registering on the had-pilot.biz test system.
Here is your hash. Please enter it in the Paste-in-Hash field
of the test form,
with your address in the MailTo field.

The test system is rate limited to one message per minute, to
curb spamming through our server.
Mailto = pythentic.client@gmail.com
Paste-in-Hash = js3p/gb/l3eH1KAPranA
If you register again you will get the same hash as a
reminder.

================================================================
====================================
```

- **The fully developed result is mailed back to pythentic.client@gmail.com.**

The test is completed with a message exchange comprising a **`test`** subject initiated mail, and the test responder reply. Similar tests consist of a message exchange initiated by the correspondent with a subject of **`spf`**, **`dkim`**, **`test`** or **`register`**. The Pythentic database accumulates records for all successfully completed tests. From these records, statistics of use are generated. These are discussed in Section 6.

5.2 Other Authenticators

As discussed above, Pythentic is an email reflector that exercises and authenticates messages from domains that deploy SPF, DKIM and DMARC protocol mechanisms. It implements syntax checking of each of the three DNS record types, and supplies the analyses back to the originating domain, together with the summary of results, the full database record, and the original received message.

The dmarc.org website lists a range of resources for deploying and testing components of a full DMARC deployment. Among these are a group of authenticating message reflectors that include Pythentic, but also Return Path[15], Port25[16] and Unlock the Inbox[17]. These all work the same way: the correspondent sends an email message to the reflector, the reflector returns a reply to the originating domain, containing analyses of a selection of authenticators. There are differences in detail between them, which we explore here. The field structure of each is summarized in Table 5-1.

Pythentic	**Return Path**
Summary of Results	Source
SPF analysis	Identity alignment for SPF and DKIM
SPF recursive records	DKIM results
DKIM analysis	DomainKeys results
DKIM record	SPF results
Pythentic X-results	DMARC results
DMARC analysis	DMARC record
Full SQLite record	
Original message received	
Port 25	**Unlock the Inbox**
Message content herald	Original Message
Summary of Results	SOA server
HELO details	PTR records
SPF check details	Last send domain
DomainKeys check	Mail Domain
DKIM check	ISIPP email cert.
DKIM record	RFC5322From
Sender ID check	Email port checks
SpamAssassin check	SPF
Explanations	SPF alignment
Original Message	Sender ID
	DomainKeys
	DKIM rec and check
	DMARC rec and check
	ADSP
	Abuse
	SpamAssassin

Table 5-1: All Authenticators Analysis Structure

[15] Return Path reflector: checkmyauth@auth.returnpath.net.
[16] Port 25 Reflector: check-auth@verifier.port25.com.
[17] Unlock the inbox: mailtest@unlocktheinbox.com.

Authentication messages contain copious detail with results of all the various mechanisms. The plain text email messages from each of these sources are consequently structured under a series of relevant headings. Thus, Pythentic reply messages start with a Summary of Results section, and continue with the SPF analysis, the fully recursive listing of SPF records, and so on. The labels in Table 5-1 represent these headings, for each authenticator.

Return Path is the closest in function to Pythentic. It provides SPF, DKIM and DMARC results, with the DMARC DNS record analysed. SPF and DKIM alignment results are additionally itemized. The SPF and DKIM records are not analyzed, and Return Path does not provide a summary of results, nor does it incorporate individual protocol results in SMTP headers, in contrast with the X-headers that Pythentic provides to the message recipient.

Port 25 provides SPF check details, DKIM check and DKIM record. There is no DMARC analysis, nor is there a fully recursive SPF record check. Port 25 has additional tests, for the Domain Keys protocol, a predecessor of DKIM, for Sender ID, an experimental alternative to SPF, and also for Spam Assassin. Some discussion of these additional mechanisms is called for here.

> **Domain Keys** [RFC4870]: Provides authentication by signing email at the Admin boundary, with the signature placed in an SMTP header. The public key is retrieved from the DNS, and no certificates are needed. Domain Keys is used for source domain authentication, but there is no role for man-in-the-middle modification checking. RFC4870 has been rendered historic by the IETF, as it is effectively obsoleted by DKIM.
>
> **Sender ID** [RFC4406]: Sender ID is an alternative to SPF that provides a test for spoofing of email domains. This protocol validates the Purported Responsible Address and the Return Path, and receivers should perform at least one of those two tests. The sending domain publishes SPF v2.0 records, which include policy stipulations. RFC4406 has been designated Experimental, and the IESG cautions that it should not be used in parallel with SPF.
>
> **Spam Assassin**[18]: includes recognition of SPF and DKIM checks. This is a Spam filter that uses Bayesian classification together with configurable static rules, to filter Spam. Individually scored, each rule may have a positive or negative score. Any message that cumulatively scores more than 5 points is regarded as spam and is recommended for discard.

Unlock the Inbox: covers a wider range of authentication than all of the above message reflectors. Still, it is not a complete superset of Pythentic. While Unlock the Inbox offers SPF, DKIM and DMARC checks and some of the DNS records, it does not perform the SPF record check – and certainly not the fully recursive check that Pythentic offers.

[18] http://spamassassin.apache.org.

However, the additional range of checks include authoritative SOA server identification, PTR records and email port checks on all intermediate MTAs as signified by every Received SMTP header in the message. Additional protocol tests include Sender ID, DomainKeys, Spam Assassin, ISIPP Email Certification and ADSP.

> **ADSP** [RFC5617] is another evolutionary artifact rendered historic by advancing experience. Author Domain Signing practices is intended as an extension to DKIM in which the "d=" domain must be the same as the "Author Address", which is the RFC5322 From address. The signing practices are signaled by a DNS record at _adsp._domainkey.example.com with a "dkim=" value of 'unknown', or 'all' or 'discardable'. Of these only 'unknown' is encouraged to be published, and it says that the domain may or may not sign any email messages. It is noted that ADSP covers the same use case as DMARC. It has been demoted to historic status in the IETF.

So what can Pythentic learn from the additional publicly accessible email authenticators? Certainly a wide range of techniques is in play, but some of these seem to be holdouts from the early evolution of email authentication mechanisms. In particular, DomainKeys, Sender ID and ADSP all represent early forms of standards solutions that are now fully functional in the combination of SPF, DKIM and DMARC. Deploying them together probably offers no additional authentication: indeed, the IETF warns explicitly against deploying SPF and Sender ID together. SpamAssassin is a different animal: it is not a protocol and standards based solution but a Bayesian classifier with heuristic rules. It is probably a productive exercise in spam control to deploy SpamAssassin and develop a functional set of heuristics. Additional study will allow us to compare the authentication gains to be had in adding it to an SPF/DKIM/DMARC deployment.

6 Experience of Use

The test system described in this report has been in operation since November 2012. The records are stored in a database and the original database schema held for three years, but was modified and extended in February 2016. So two sets of records will be described here, with the first set patched to as best as possible match to the second, field-for-field. However, since the first set lacks a saved **`Subject`** field, the analysis of distribution of messages by subject is applied to the second set only. In what follows the first set is referenced as the 'old' database records and the second set is referenced as the 'new' database records. The old set spans 38 months and includes 4882 records. The new set spans 5 months and includes 5185 records. Table 6-1 shows that the monthly averages over the life of the test system are: 61 test per month in 2013, 185/m in 2014, 146/m in 2015 and 521 test per month in 2016. The peak month overall was 1612 test in June 2016. With the re-design and re-launch in February 2016, the Pythentic test system seems to be on a continuing upward curve. This bodes well for the uptake of DMARC and related email authentication technologies in small domains around the world, as the remainder of this analysis shows.

Year	Monthly Average	Peak Month
2013	61	126
2014	185	285
2015	146	217
2016	521	1612

Table 6-1: Distribution of Use Over Time

The analysis of records and results that follows examines a number of aspects. In Section 6.1, the two sets of records are analyzed by the source domain of the received messages. Frequencies from individual domains, and aggregate frequencies from Top Level Domains, are examined.

The **`Subject`** analysis of the new dataset is conducted in Section 6.2, correlated with delivery dispositions. In particular, the distribution of what messages, by subject, are Delivered, Discarded or Rejected, is discussed.

In Section 6.3 the analysis of SPF results versus DKIM results is conducted, as a way to discover which is the more effective, and whether they are indeed better in combination.

Sections 6.4 and 6.5 investigate some other details, of SPF usage and IPv6 usage, respectively.

6.1 Analysis by Source Domain

The old database records 2229 unique users over 38 months, while the new database records 3576 unique users over 5 months. The relative increase in use perhaps says more about the system's use as a spam target than its popularity as a test system: spam mails are more likely to be discarded, and the new database has a markedly increased rate of discard, possibly as a result of its longer time exposure.

Uses	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**20+**
Old	1366	392	221	94	57	31	17	15	11	5	6
New	3077	232	106	62	32	12	12	11	9	5	5

Table 6-2: Frequency per Domain

The effectiveness of Pythentic as a test system increases with the number of uses for each user, since proper exercise of DMARC, DKIM and SPF responses requires a minimum of 3 messages. Greater numbers than this indicate the system is being exploited as a diagnostic tool. The range of uses between 3 and 10 per user is in the low 100s down to the low 10s.

The userbase of the test system is quite comprehensively global. There are 127 unique Top Level Domains (TLD) in the old database and 119 in the new database.

TLD	COM	NET	US	ORG	DE	UK	ES	FR	NL	RU	BR
Old	933	218	-	128	93	68	51	47	39	33	32
New	2246	274	112	112	50	74	42	26	44	27	36

Table 6-3: Frequency of Use by TLD

The TLDs .com, .net and .org are among the most frequent, though .us usage expands greatly in the new database. The top countries in both databases include Germany (.de), the UK (.uk), Spain (.es), France (.fr), the Netherlands (.nl), Russia (.ru) and Brazil (.br). Some of the new TLDs created since IANA expanded the set in january 2012 are also represented, including .coffee, .clinic, .tirol, .xxx and .zone.

6.2 Analysis by Subject and Delivery Disposition

Subject	Deliver	Discard	Reject
dmarc	639	4	7
dkim	244	0	21
spf	229	8	36

test	323	0	35
register	186	5	25
p.dkim.bad	59	0	5
p.dkim.nocr	31	0	4
p.dkim.nolf	19	0	2
p.dkim.bh	27	0	1
p.spf.spoof	28	0	4
Other (Spam)	549	76	68

Table 6-4: Analysis by Subject and Delivery Disposition

The new database records both subject and delivery disposition. The correlation of these fields is instructive for the frequency of tests run and their success rate. Delivery dispositions include Deliver, Discard and Reject. A message is delivered to the (Pythentic) mailbox if it passes DMARC. It is rejected if it fails DMARC, but contains a valid test subject, thereby precipitating a useable response back to legitimate users. Messages discarded are those that fail DMARC and have a non-test, spam-like subject. Table 6-3 shows there is a large number of messages in this category, with a wide range of subjects. This also includes a substantial proportion of messages with adult contents or likely phishing subjects: these are just the ones we would wish to have selected out, and include things like “instacheat request” and “h00kup request”, as well as “reminder for unpaid account”, and various flavors of “how to make money” type confidence scams. The discards of legitimate subjects like **dmarc**, **spf** and **register** are very low, in the unit numbers. The success of authentication control policies is also shown by minimization in successful delivery of spam subjects. The most frequent of these include “hello bhai” with four deliveries, “grattis du har vumnit en Samsung smart tvo av ass” with four deliveries, and “ups server startups at usd 6” with three deliveries. In summary, the high numbers of legitimate deliveries and correct discards combined with the low numbers of unwanted deliveries suggests that effective deployment of authentication checks goes a long way to curbing the spam problem, even without deploying heuristic-style anti-spam products.

6.3 Analysis by SPF versus DKIM Result

SPF Result	**Old DB**		**New DB**		**Old Total**	**New Total**
	Pass	**Fail**	**Pass**	**Fail**		
none	195	434	90	1647	629	1737
pass	2808	977	1442	536	3785	1978
fail	25	47	40	622	72	662
neutral	12	25	7	326	37	333
softfail	22	38	30	306	60	336
permerror	67	65	346	119	133	465
temperror	133	60	0	0	193	0
Totals	3262	1646	1955	3556	4908	5511

Table 6-5: Analysis by SPF versus DKIM result

Only one of a successful SPF or a successful DKIM result is needed for DMARC authentication and delivery. DKIM outcomes have two possible states: pass and fail. SPF outcomes have seven possible states: **`none`**, **`pass`**, **`fail`**, **`neutral`**, **`softfail`**, **`permerror`**, **`tempfail`**. Of these, none, pass neutral lead to a deliverable result. Thus a message will be delivered by DMARC policy with any DKIM pass, or with a DKIM fail combined with an SPF none, pass or neutral.

From Table 6-4, the number of messages that pass SPF while DKIM fails is 434 + 977 + 25 = 1436 out of a total 4908 in the old database, and 1647 + 536 + 326 = 2509 out of a total 5511 in the new database. On the other hand, the number of messages that fail SPF but pass DKIM is 247/4908 in the old, 416/5511 in the new database, suggesting that SPF is more effective as a single mechanism than DKIM. Messages that pass both SPF and DKIM are 195 + 2808 + 12 = 3015/4908 in the old, 90 + 1442 + 7 = 1539/5511 in the new database. Messages that fail both SPF and DKIM and are therefore discarded or rejected are: 47 + 38 + 65 + 60 = 210/4908 old, and 622 + 306 + 119 = 1045/551 new. The aggregate pass/fail matrix for SPF and DKIM in old and new databases is given in Table 6-6.

	Old		**New**	
	DKIM Pass	**DKIM Fail**	**DKIM Pass**	**DKIM Fail**
SPF Pass	3015	1436	1539	2509
SPF Fail	247	210	416	1045

Table 6-6: SPF versus DKIM Pass/Fail Analysis

In aggregate, with the combination of SPF or DKIM pass, and SPF and DKIM fail, there are 4698/4908 authenticated messages in the old database, and 4464/5511 authenticated messages in the new database. So there are 210/4908 = 4 % not authenticated in the old, and 1045/5511 – 19 % not authenticated in the new database.

6.4 Subverting SPF Records

In evaluating SPF, each mechanism in a record is checked in turn and the first one to trigger a match terminates SPF processing. So for example in verifying a message from a sender at IP address 172.6.148.151, if the retrieved SPF record contains a mechanism (+)ip4:172.6.148.0/24, the mechanism matches and the SPF module returns a ‘pass’. If no intermediate mechanism matches, the ‘all’ placed at the end always matches. Most often this will be linked with a ‘-‘ qualifier (-all), and SPF returns a fail. The qualifier ‘~’ indicating a softfail result or the ‘?’ indicating a neutral result are also legitimately useful outcomes. Ending the SPF record with ‘+all’ effectively says “all other addresses in the IPv4 number space are good”. This is clearly not a useful mechanism for discerning good email messages. The old database contains 17 out of 28826 records with ‘+all’. The new database contains 1349 from 11653 records with ‘+all’, or 11 % of the total. Spammers

seem to be picking up on this trick for getting dubious messages delivered. For this reason, we advise that in processing SPF records, '+all' ought to be treated as an error.

6.5 The Uptake of IPv6

Other efforts at NIST have focused on supporting Office of Management and Busget edicts to drive the deployment of IPv6 in the Federal Government [SP500-177]. It is useful to monitor the uptake of IPv6 in other realms, and the mechanisms of SPF offer this opportunity too. The ip6:<ipv6address> mechanism allows for identification of IPv6 message sources, where the ip4:<ipv4addreess> mechanism identifies IPv4 message sources. The more 'ip6:' mechanisms appearing in SPF records, the more the uptake of IPv6. A comparison in old and new databases shows 3049 v6 versus 23839 v4 mechanisms in the old or 12.8 %, and 1358 v6 versus 7346 v4 mechanisms in the new database, or 18.5 %. This does seem to indicate a gradually improving uptake of IPv6.

7. Coda

DMARC is a component in a compendium of protocols intended to create an open bi-lateral system for authenticating email messages. SPF and DKIM as individual components in that system give message receivers assurance of the authenticity or otherwise of each message from a purported sending domain, but this leaves sending domain owners blind to the effects of their policies posted to the DNS. With DMARC as an additional sending domain policy, also posted in the DNS, receivers can determine sending domain intentions and dispose of messages accordingly, and provide feedback on the effects of those policies. Clearly a full deployment of DMARC gives a reporting burden to receivers, who may have to create aggregate reports periodically, sent out to multiple sending domain owners, and potentially also send out forensic reports on demand, reporting on spoofed and phishing messages received. Sending these reports is ultimately voluntary, and small scale mail administrations may opt not to do this. Is DMARC useful for them? We think it is. In addition to the authentication guidance given by SPF and DKIM, DMARC stipulates domain name alignment requirements and a sender preference on message delivery disposition: useful guidance on false negatives (which should be delivered) and false positives (which should not be delivered).

The usefulness of DMARC to the sending domain is clear: they can receive copious and detailed feedback from a wide spectrum of mail receivers about whether their domain is susceptible to spoofing, quantities of authentic mail delivered, and quantities of inauthentic mail disposed according to posted DMARC policy. Fine tuning of sender policy can be done, the more that receivers initiate feedback. On the other hand, to the extent that receivers choose not to send feedback, sending domains are again partially blind to the effectiveness of their policies.

In creating and deploying the Pythentic email message reflector, the HAD project provides a tool to give specific feedback to message senders on the syntax of SPF, DKIM and DMARC TXT records in the DNS, and on the disposition of messages on a per message basis. This is analogous to instigating forensic reports, but in this case receiving results on successful or failed outcomes. It is useful to senders, to help them debug their DNS records, and to help them tune their policies before subjecting them to the broader world of receivers. In this case our return reply provides confirmation of the test.

Pythentic is useful to receivers in that we post SPF, DKIM and DMARC records which they can use to authenticate our mail, and test their respective protocol modules. In this case our return reply is the test: though we are unable ourselves to evaluate the outcome. It is setting up a sympathetic responder for correspondents to use as they need. Receiver testing is facilitated by our generating a good DKIM signature on every response message, for the recipient to authenticate against our DKIM DNS record. Through specific tests we also generate defective DKIM signatures, to test their DKIM processing modules.

We chose to implement a message authenticating reflector rather than a full reference implementation of DMARC, as we judged that the syntactic and policy based feedback

was more immediately useful than any ‘aggregate’ analysis we might return. The value to an initiating domain is in receiving aggregate reports from a wide spectrum of receiving domains, so that the initiating domain can get a good picture of the use and abuse of its domain. Pythentic is not a large scale message initiator, and would gain scant value from trying to build such a picture, with probably few aggregate reports receivable. As a receiving domain, Pythentic is also small-scale, and any aggregate reports we produce will be insignificant compared with the scale and range the initiating domain requires. It is much more useful for us to simply send test feedback and forensic reports for discrete failures.

`

Bibliography

[SP800-45] NIST Special Publication 800-45 version 2. *Guidelines on Electronic Mail Security*, National Institute of Standards and Technology, Gaithersburg, Maryland, February 2007. http://csrc.nist.gov/publications/nistpubs/800-45-version2/SP800-45v2.pdf

[SP800-177] NIST Special Publication 800-177. *Trustworthy Email*, National Institute of Standards and Technology, Gaithersburg, Maryland. October 2016. http://csrc.nist.gov/publications/nistpubs/800-177/SP800-177.pdf

[RFC821] J. Postel. *Simple Mail Transfer Protocol*. Internet Engineering Task Force, Request for Comments 821, August 1982. https://datatracker.ietf.org/doc/rfc821/

[RFC5321] J. Klensin. *Simple Mail Transfer Protocol*. Internet Engineering Task Force, Request for Comments 5321, April 2008. https://datatracker.ietf.org/doc/rfc5321/

[RF5322] P. Resnick. *Internet Messaging F*ormat. Internet Engineering Task Force, Request for Comments 5322, October 2008. https://datatracker.ietf.org/docs/rfc5322/

[RC1034] P. Mockapetris. *Domain Names – Concepts and Facilities*. Internet Engineering Task Force, Request for Comments 1034, November 1987. https://datatracker.ietf.org/docs/RFC1034/

[RFC1035] P. Mockapetris. *Domain Names – Implementation and Specification*. Internet Engineering Task Force, Request for Comments 1035, November 1987. https://datatracker.ietf.org/docs/RFC1035/

[RC4033] R. Arends, R. Austein, M. Larson, D. Massey and S. Rose. *DNS Security Introduction and Requirements.* Internet Engineering Task Force, Request for Comments 4033. March 2005. https://datatracker.ietf.org/docs/RFC4033/

[RFC4034] R. Arends et al. *Resource records for the DNS Security Extensions*. Internet Engineering Task Force. Request for Comments 4034. March 2005. Https://datatracker.ietf.org/docs/RFC4034.

[RFC4035] R. Arends et al. *Protocol Modifications for the DNS Security Extensions*. Internet Engineeering Task Force Request for Comments 4035. https://datatracker.ietf.org/docs/RFC4035/

[X509] ITU-T Recommendation X.509 (2005). Information Technology – Open Systems Interconnection – The Directory: Authentication Framework 08/05

[RFC5246] T. Dierks and E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.2.* Internet Engineering Task Force Request for Comments 5246, August 2008. https://datatracker.ietf.org/docs/RFC5246/

[RFC5750] B. Ramsdell and S. Turner. Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Certificate Handling. Internet Engineering Task Force Request for Comments 5750. https://datatracker.ietf.org/docs/RFC5750/

[RFC5751] B. Ramsdell et al. *Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification*. Internet Engineering Task Force Request for Comments 5751. https://datatracker.ietf.org/docs/RFC5751/

[RFC4880] J. Callas, l. Donnerhacke, H.Finney, D. Shaw and R. Thayer. *OpenPGP Message Format*. Internet Engineering Task Force Request for Comments 4880. https://datatracker.ietf.org/docs/RFC4880/

[RFC7208] S. Kitterman. *Sender Policy Framework (SPF) for Authorizing Use of Domains in Email Version 1*. Internet Engineering Task Force Request for Comments 7208. https://datatracker.ietf.org/docs/RFC7208/

[RFC6376] D. Cocker, T. Hansen, M. Kucherawy. *Domain Keys Identified Mail (DKIM) Signatures.* Internet Engineering Task Force Request for Comments 6376. https://datatracker.ietf.org/docs/RFC6376/

[RFC7489] M. Kucherawy and E. Zwicky. *Domain-based Message Authentication, Reporting and Conformance (DMARC*). Internet Engineering Task Force Request for Comments 7489. March 2015. https://datatracker.ietf.org/docs/RFC7489/

[RFC854] J. Postel and J. Reynolds. *Telnet Protocol Specification*. Internet Engineering Task Force Request for Comments 854. May 1983. https://datatracker.ietf.org/docs/RFC854/

[RFC793] Information Sciences Institute, Marina del Rey, California. *Transmission Control Protocol*. Internet Engineering Task Force Request for Comments 793. September 1981. https://datatracker.ietf.org/docs/RFC793/

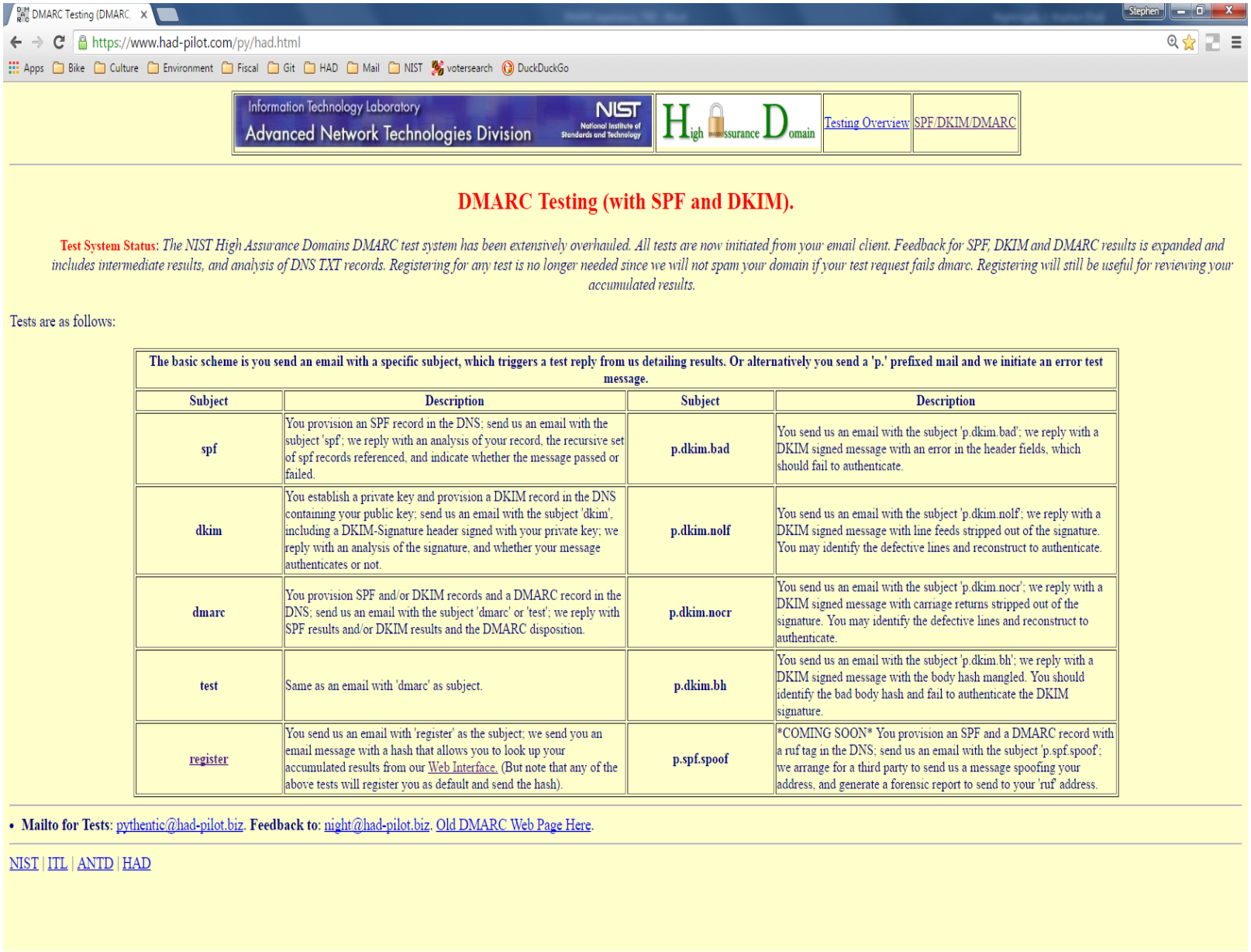
[RFC4870] M. Delany. *Domain based Email Authentication Using Public Keys Advertised in the DNS (DomainKeys).* Internet Engineering Task Force Request for Comments 4870. https://datatracker.ietf.org/docs/RFC4870/

[RFC4406] J. Lyon and M. Wong. *Sender-ID: Authenticating Email*. Internet Engineering Task Force Request for Comments 4406. https://datatracker.ietf.org/docs/RFC4406/

[RFC5617] E, Allman, J. Fenton, M. Delany, J. Levine. *DomainKeys Identified Mail (DKIM) Author Domain Signing Practices (ADSP).* Internet Engineering Task Force Request for Comments 5617. https://datatracker.ietf.org/docs/RFC5617/

[SP800-177] NIST Special Publication 800-177, Trustworthy Email (Draft), National Institute of Standards and Technology, Gaithersburg, MD, Source Pending.

Appendix 1: The Tests (overleaf)



DMARC Testing (with SPF and DKIM).

Test System Status: *The NIST High Assurance Domains DMARC test system has been extensively overhauled. All tests are now initiated from your email client. Feedback for SPF, DKIM and DMARC results is expanded and includes intermediate results, and analysis of DNS TXT records. Registering for any test is no longer needed since we will not spam your domain if your test request fails dmarc. Registering will still be useful for reviewing your accumulated results.*

Tests are as follows:

The basic scheme is you send an email with a specific subject, which triggers a test reply from us detailing results. Or alternatively you send a 'p.' prefixed mail and we initiate an error test message.			
Subject	**Description**	**Subject**	**Description**
spf	You provision an SPF record in the DNS; send us an email with the subject 'spf'; we reply with an analysis of your record, the recursive set of spf records referenced, and indicate whether the message passed or failed.	**p.dkim.bad**	You send us an email with the subject 'p.dkim.bad'; we reply with a DKIM signed message with an error in the header fields, which should fail to authenticate.
dkim	You establish a private key and provision a DKIM record in the DNS containing your public key; send us an email with the subject 'dkim', including a DKIM-Signature header signed with your private key; we reply with an analysis of the signature, and whether your message authenticates or not.	**p.dkim.nolf**	You send us an email with the subject 'p.dkim.nolf'; we reply with a DKIM signed message with line feeds stripped out of the signature. You may identify the defective lines and reconstruct to authenticate.
dmarc	You provision SPF and/or DKIM records and a DMARC record in the DNS; send us an email with the subject 'dmarc' or 'test'; we reply with SPF results and/or DKIM results and the DMARC disposition.	**p.dkim.nocr**	You send us an email with the subject 'p.dkim.nocr'; we reply with a DKIM signed message with carriage returns stripped out of the signature. You may identify the defective lines and reconstruct to authenticate.
test	Same as an email with 'dmarc' as subject.	**p.dkim.bh**	You send us an email with the subject 'p.dkim.bh'; we reply with a DKIM signed message with the body hash mangled. You should identify the bad body hash and fail to authenticate the DKIM signature.
register	You send us an email with 'register' as the subject; we send you an email message with a hash that allows you to look up your accumulated results from our Web Interface. (But note that any of the above tests will register you as default and send the hash).	**p.spf.spoof**	*COMING SOON* You provision an SPF and a DMARC record with a ruf tag in the DNS; send us an email with the subject 'p.spf.spoof'; we arrange for a third party to send us a message spoofing your address, and generate a forensic report to send to your 'ruf' address.

- **Mailto for Tests**: pythentic@had-pilot.biz. **Feedback to**: night@had-pilot.biz. Old DMARC Web Page Here.

NIST | ITL | ANTD | HAD