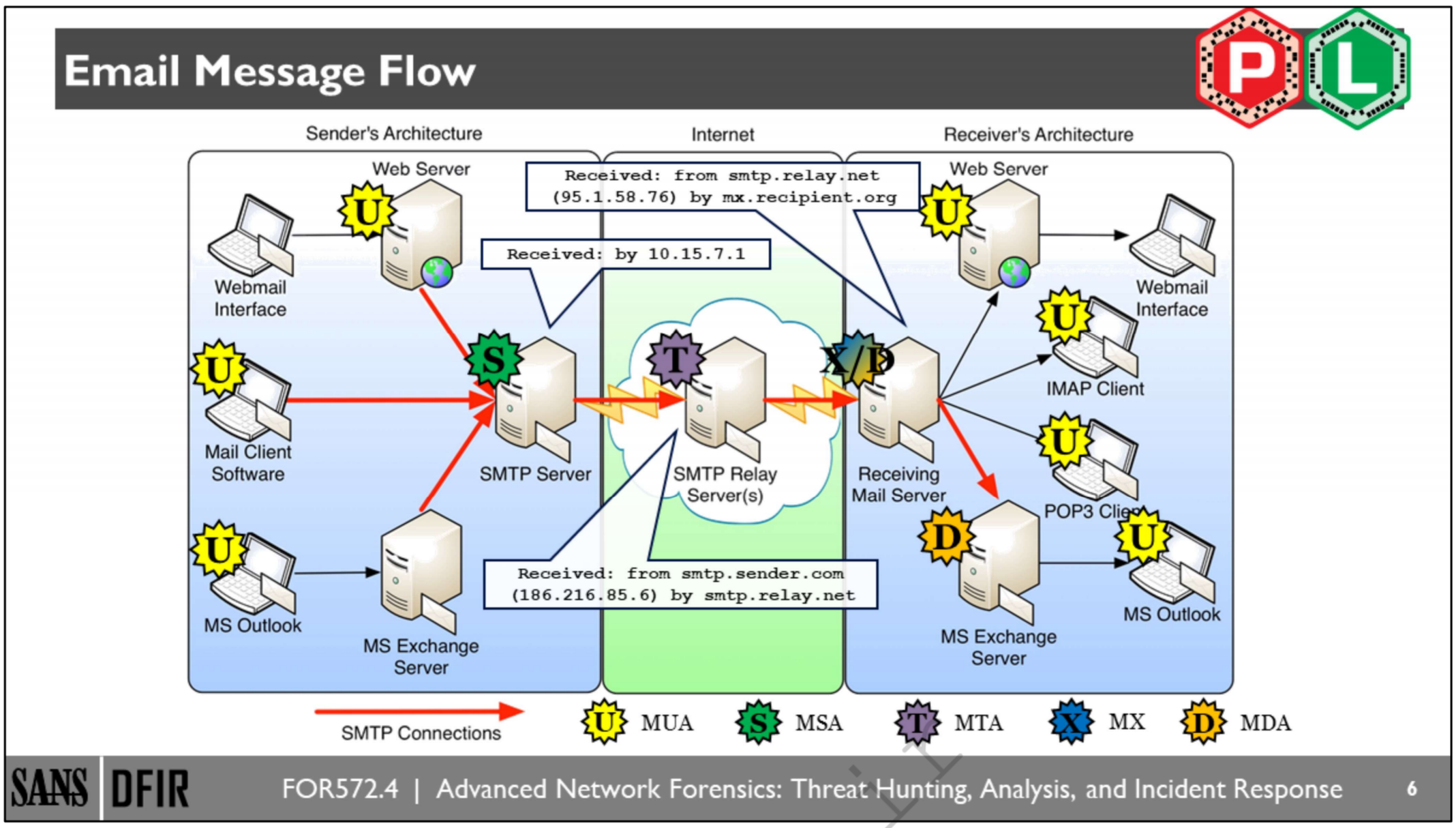


Here, you can see examples of several common email paths. Although there are too many possible configurations to detail here, these cover those you'd expect to see in most environments.

Note that although some clients (MUAs) such as the webmail and Outlook systems don't natively use SMTP to send email messages, their immediate upstream systems (MSA) pass sent messages via SMTP-speaking paths very soon into the message's life cycle. As soon as the message is routed via the Internet, SMTP is the name of the game. Usually, one or more relays (MTAs) will receive a message, and then pass it along based on the preconfigured routing rules. Finally, the designated recipient server (MX) receives the message and places it into the receiving user's mailbox (MDA). At the other end of the process, the user's MUA receives and displays the message. (Unless they ignore or delete it.)

What's interesting to note is that at each stage of the transmission, servers generally add a message header indicating the server's hostname, date, and time the message was processed, and some other details that can be valuable in determining the path a message took on its journey from sender to recipient. Of course, these headers are only required by RFC and not law—so some servers may alter the existing or expected header values.

An example of a message a Gmail user sent to the SANS DFIR mailing list is on the next page. The headers are typically in reverse chronological order. Therefore, the last header indicates that the originator used the web interface to send the message, with each subsequent node adding its identity as well as the previous node. This establishes a clear path that the message took, including both hostnames and corresponding IP addresses (public and private). A second set of sample headers reflects a spam message that was apparently sent with the Gmail API.

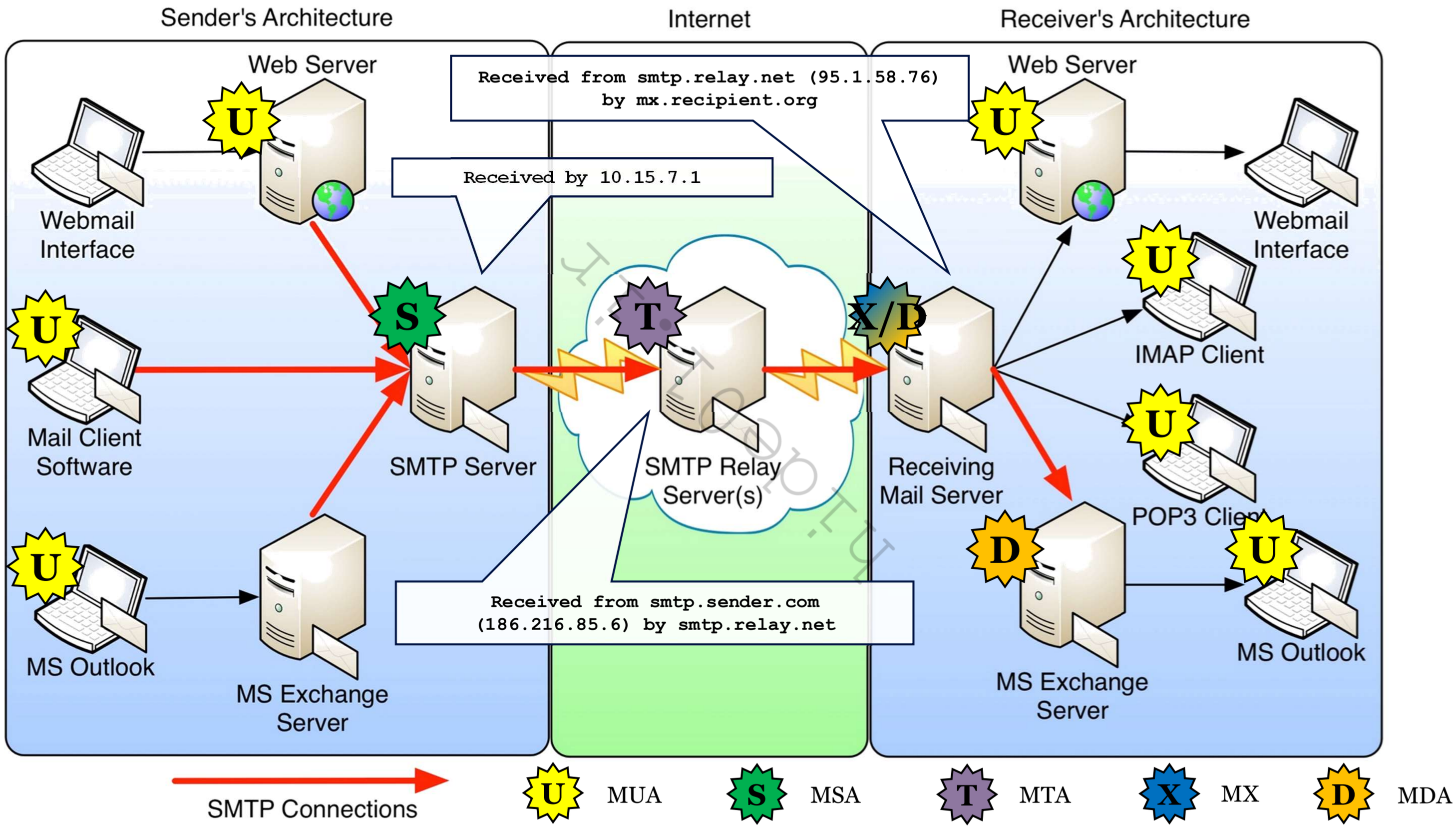
Making sense of these headers, specifically the "`Received:`" data points, can be confusing. Fortunately, tools such as the G Suite Messageheader web app[1] can visualize them, provided OPSEC is appropriately addressed.

1. https://for572.com/dvg9l

```
Return-Path:
  <SRS0=vduv=C6=lists.sans.org=advisory-board-open-
  bounces@identityvector.com>
X-Original-To: phil@lewestech.com
Delivered-To: phil@simcoe.identityvector.com
Received: by simcoe.identityvector.com (Postfix) with ESMTPS id 02C9561871
  for <phil@lewestech.com>; Wed, 12 Jul 2023 14:44:18 +0000 (UTC)
X-Original-To: advisory-board-open@lists.sans.org
Delivered-To: advisory-board-open@lists.sans.org
Received: from mail-pj1-f52.google.com (mail-pj1-f52.google.com
  [209.85.216.52]) (using TLSv1.2 with cipher AES256-GCM-SHA384 (256/256
  bits)) (No client certificate requested) by lists-mta.sans.org (Postfix)
  with ESMTPS id 5D96684F2D for <advisory-board-open@lists.sans.org>; Wed, 12
  Jul 2023 14:42:04 +0000 (UTC)
Received: by mail-pj1-f52.google.com with SMTP id
  98e67ed59e1d1-262ff3a4659so5124361a91.0 for
  <advisory-board-open@lists.sans.org>; Wed, 12 Jul 2023 07:42:04 -0700 (PDT)
X-Google-Smtp-Source:
APBJJlE6K3vIA7ot4jpkEcfiK/m/sHsI8iXbck2hxZ658QhOvAvyb8AbLq2KnIDxv67jsgi+DUSmO
D/qc2hLnzt56HE=
X-Received: by 2002:a17:90a:6283:b0:262:ea30:2cc3 with SMTP id
  d3-20020a17090a628300b00262ea302cc3mr19412879pjj.2.1689172923379; Wed, 12
  Jul 2023 07:42:03 -0700 (PDT)
MIME-Version: 1.0
Date: Wed, 12 Jul 2023 10:41:50 -0400
Message-ID:
  <CAB3qvf95nsa2QoZSGbvPxCHzBrCvHZExycQBP6R3v71xWj7HXA@mail.gmail.com>
List-Id: Unmoderated list for GIAC Advisory Board
  <advisory-board-open.lists.sans.org>
Errors-To: advisory-board-open-bounces@lists.sans.org
Sender: "advisory-board-open" <advisory-board-open-bounces@lists.sans.org>



From: Spammer <dumb.spammer.fakeemail@gmail.com>
To: Recipient User <recipient@lewestech.com>
Message-ID:
   <CAG_jHOgGc403TRSzLSfQtxRyEfNzJszuqi3803-41373yYu62w@mail.gmail.com>
X-Google-Smtp-Source:
   AFSGD/Xu/ey9i1zaLX/1veZinzPE4qfmuehL4JttTITQ8mIgoaSTQcVSIjJU9lVJ6fuxNy1VRp
   yEVDK3pX1n4yYS+vA=
X-Google-Sender-Auth: 8WCA33I51hGwBZ0im2PyGSUpXE0
Received: by simcoe.identityvector.com (Postfix) with ESMTPS id F31BE61D9F
   for <recipient@lewestech.com>; Thu, 13 Dec 2022 04:36:19 +0000 (UTC)
Received: by mail-vs1-f67.google.com with SMTP id b74so412194vsd.9
   for <recipient@lewestech.com>; Wed, 12 Dec 2022 20:36:19 -0800 (PST)
X-Received: by 2002:a67:98c3:: with SMTP id
   g64mr10601079vsh.225.1544675779190; Wed, 12 Dec 2022 20:36:19 -0800 (PST)
Received: from 52669349336 named unknown by gmailapi.google.com with
   HTTPREST;
 Wed, 12 Dec 2022 23:36:18 -0500
Sender: Archana <dumb.spammer.fakeemail@gmail.com>
```

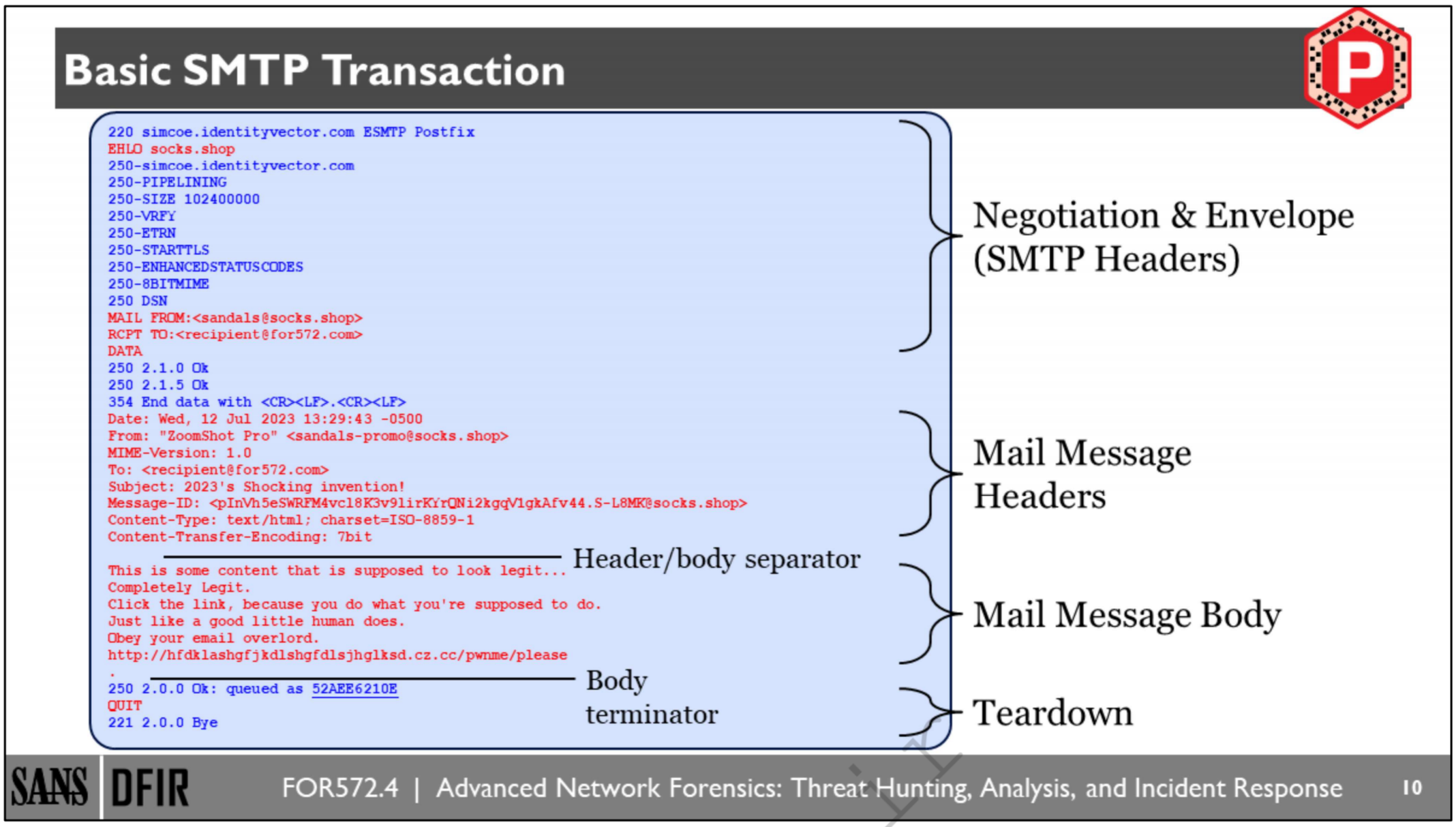


SMTP Transmission Characteristics

- Multiple ports, same basic protocol
  - TCP/25 (often with STARTTLS)
  - TCP/587 (with authentication, usually with STARTTLS)
  - TCP/465 (TLS-wrapped)
- Created in simpler times
  - English ASCII was sufficient, SPAM didn't exist, and trust ruled the Internet
  - Protocol extensions and updates address these and other developments

Primarily, SMTP is associated with TCP port 25 but has more recently expanded to include port 587. Typically, SMTP usage over port 587 requires authentication, which we'll discuss shortly. TCP/465 may also be used, especially for server-to-server relays. This involves TLS-wrapped SMTP, much like HTTPS uses TLS-wrapped HTTP. However, the "STARTTLS" option on ports 25 or 587 can also provide encryption, as we'll see shortly.

It's important to recognize that many legacy protocols—including SMTP—were created when the Internet was a completely different entity than it is now. Today's globally ubiquitous and often hostile environment demands a different mindset for protocol design. For example, the original SMTP specification had no considerations for authentication/authorization, non-English text, SPAM abatement, or even attachments. These features, which we consider integral to the modern email landscape, were all established after SMTP was already in widespread use. We'll look at a few of the protocol extensions that brought such features into being and how they affect our analysis of the protocol.



This is perhaps the most rudimentary example of an SMTP submission, as seen from a network vantage point. In this case, the "`socks.shop`" server is relaying a message to the "`mail.identityvector.com`" server, which uses the internal hostname "`simcoe.identityvector.com`". The "`mail.identityvector.com`" server has been designated as an MX for the "`for572.com`" domain, so this transaction represents a transaction between an MTA and an MX.

First, the exchange contains a series of greetings between the client and server that establish the mutually supported protocols and protocol extensions. This starts with an "`EHLO`", or "extended hello" command, but some rare exchanges may include the original "`HELO`" instead.

The two components of the SMTP negotiation are the "`MAIL FROM:`" and "`RCPT TO:`" commands, which designate the immediate sender and intended recipient of the message, respectively. After the SMTP negotiation, the sender sends mail message header data. Long headers would be wrapped and indented on subsequent lines. The protocol stipulates that indented lines are treated as continuations of their predecessor.

The separator between headers and body content is a "double-CRLF", consisting of a carriage return and newline followed by another carriage return and newline. In hex, these bytes are `0x0D0A0D0A`. This sequence is a common header/body separator in other ASCII protocols such as HTTP as well.

The message body here contains HTML content, and the client indicates it is done with the message by sending a single period character on a line by itself.

After the body—and therefore the entire message—is complete, the server provides the queued message identifier (which is generally available in the mail server's logs) and the client initiates a clean teardown procedure.

```
220 simcoe.identityvector.com ESMTP Postfix
EHLO socks.shop
250-simcoe.identityvector.com
250-PIPELINING
250-SIZE 102400000
250-VRFY
250-ETRN
250-STARTTLS
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
MAIL FROM:<sandals@socks.shop>
RCPT TO:<recipient@for572.com>
DATA
250 2.1.0 Ok
250 2.1.5 Ok
354 End data with <CR><LF>.<CR><LF>
Date: Wed, 12 Jul 2023 13:29:43 -0500
From: "ZoomShot Pro" <sandals-promo@socks.shop>
MIME-Version: 1.0
To: <recipient@for572.com>
Subject: 2023's Shocking invention!
Message-ID: <pInVh5eSWRFM4vcl8K3v9lirKYrQNi2kgqV1gkAfv44.S-L8MK@socks.shop>
Content-Type: text/html; charset=ISO-8859-1
Content-Transfer-Encoding: 7bit

This is some content that is supposed to look legit...
Completely Legit.
Click the link, because you do what you're supposed to do.
Just like a good little human does.
Obey your email overlord.
http://hfdklashgfjkdlshgfdlsjhglksd.cz.cc/pwnme/please
.
250 2.0.0 Ok: queued as 52AEE6210E
QUIT
221 2.0.0 Bye
```

Negotiation & Envelope (SMTP Headers)

Mail Message Headers

Header/body separator

Mail Message Body

Body terminator

Teardown

The Results

```
Received: by simcoe.identityvector.com (Postfix) with ESMTP id 52AEE6210E
        for <recipient@for572.com>; Wed, 12 Jul 2023 18:48:54 +0000 (UTC)
From: "ZoomShot Pro" <sandals@socks.shop>
To: <recipient@for572.com>
Subject: 2023's Shocking invention!
Date: Wed, 12 Jul 2023 13:29:43 -0500
Message-ID: <pInVh5eSWRFM4vcl8K3v9lirKYrQNi2kgqV1gkAfv44.S-L8MK@socks.shop>
MIME-Version: 1.0
X-Spam-Checker-Version: SpamAssassin 3.4.0 (2014-02-07) on
        simcoe.identityvector.com
X-Spam-Level:
X-Spam-Status: No, score=-1.3 required=4.1 tests=BAYES_00,DKIM_SIGNED,
        DKIM_VALID,DKIM_VALID_AU,HEADER_FROM_DIFFERENT_DOMAINS,HTML_MESSAGE,
        HTML_MIME_NO_HTML_TAG,MIME_HTML_ONLY,NO_RELAYS,T_SCC_BODY_TEXT_LINE,
        URIBL_BLOCKED autolearn=ham autolearn_force=no version=3.4.0
X-Original-To: recipient@for572.com
Authentication-Results: simcoe.identityvector.com;
        dkim=pass (1024-bit key) header.d=socks.shop header.i=sandals@socks.shop
        header.b="grdRzF7i"
X-Greylist: delayed 609 seconds by postgrey-1.37 at simcoe.identityvector.com;
        Wed, 12 Jul 2023 18:48:54 UTC

This is some content that is supposed to look legit...
Completely Legit.
Click the link, because you do what you're supposed to do.
Just like a good little human does.
Obey your email overlord.
http://hfdklashgfjkdlshgfdlsjhglksd.cz.cc/pwnme/please
```

New "`Received:`" header

Additional headers (added by MX server, sometimes also by receiving client)

Header/body separator

The resulting message, as delivered to the recipient's Inbox (and possibly stored on disk, depending on the MDA software), is shown here. You can see that the headers included by the sending SMTP server remain intact during the MTA-to-MX transaction, but that the MX has added some headers of its own.

Most important to note is that an additional "`Received:`" header is added at each hop in the transaction. It's also useful to recognize that any SMTP server or even the ultimate receiving client in the path can add, remove, or alter whatever message headers or body data it's configured to mangle. For example, most mail servers will remove headers regarding the validation/authentication of a message such as DKIM or DomainKeys, since those are often relevant to just a single hop on the delivery path. If that message were to be relayed further, new authentication headers would be generated and inserted for the validation/authentication of the next hop on the path.

Any MTA, MX, or mail client along the way can optionally add whatever "`X-`" headers, indicating free-form extension data. Often this includes antivirus, antispam, and other features, but can truly be anything the server is configured to add. Most webmail providers now include a header containing the encoded or encrypted IP address of the message originator. These can aid in tracking down abuse and other malicious activity. Mail clients might remove headers that are no longer relevant after the message was delivered or add headers that enable client-specific features like message threading, read/unread status, and more.

However, it would be uncommon for any well-behaving mail server or client to remove or alter the "`Received:`" headers. These often provide a valuable series of "breadcrumbs" indicating the timestamps and systems involved in the message's pathway from sender to recipient.

```
Received: by simcoe.identityvector.com (Postfix) with ESMTP id 52AEE6210E
        for <recipient@for572.com>; Wed, 12 Jul 2023 18:48:54 +0000 (UTC)
From: "ZoomShot Pro" <sandals@socks.shop>
To: <recipient@for572.com>
Subject: 2023's Shocking invention!
Date: Wed, 12 Jul 2023 13:29:43 -0500
Message-ID: <pInVh5eSWRFM4vcl8K3v9lirKYrQNi2kgqV1gkAfv44.S-L8MK@socks.shop>
MIME-Version: 1.0
X-Spam-Checker-Version: SpamAssassin 3.4.0 (2014-02-07) on
        simcoe.identityvector.com
X-Spam-Level:
X-Spam-Status: No, score=-1.3 required=4.1 tests=BAYES_00,DKIM_SIGNED,
        DKIM_VALID,DKIM_VALID_AU,HEADER_FROM_DIFFERENT_DOMAINS,HTML_MESSAGE,
        HTML_MIME_NO_HTML_TAG,MIME_HTML_ONLY,NO_RELAYS,T_SCC_BODY_TEXT_LINE,
        URIBL_BLOCKED autolearn=ham autolearn_force=no version=3.4.0
X-Original-To: recipient@for572.com
Authentication-Results: simcoe.identityvector.com;
        dkim=pass (1024-bit key) header.d=socks.shop header.i=sandals@socks.shop
        header.b="grdRzF7i"
X-Greylist: delayed 609 seconds by postgrey-1.37 at simcoe.identityvector.com;
        Wed, 12 Jul 2023 18:48:54 UTC

This is some content that is supposed to look legit...
Completely Legit.
Click the link, because you do what you're supposed to do.
Just like a good little human does.
Obey your email overlord.
http://hfdklashgfjkdlshgfdlsjhglksd.cz.cc/pwnme/please
```

New "`Received:`" header

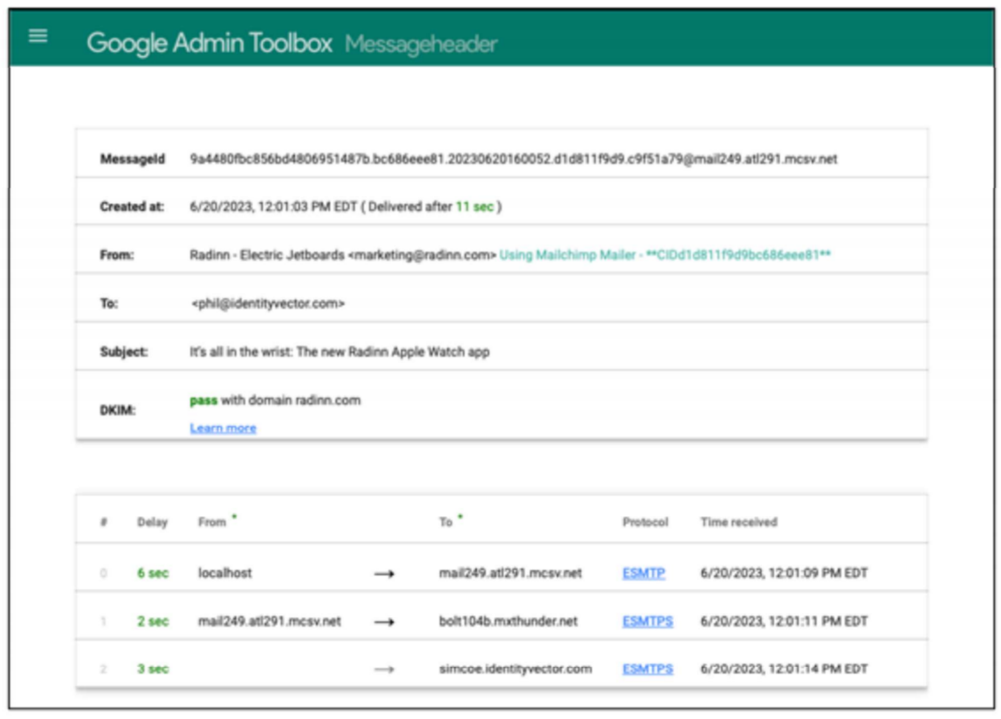
Additional headers (added by MX server, sometimes also by receiving client)

Header/body separator

Mail Pathway Visualization Tools

- Parse and visualize "`Received:`" headers
- Google Messageheader
- MxToolBox Email Header Analyzer
- Numerous others

- Consider OPSEC!



Given the potential investigative value of the "`Received:`" headers in a message, it can sometime be helpful to parse and visualize those artifacts to make what could be a long and convoluted history easier to understand. There are a number of tools that can provide this functionality.

Google's Admin Toolbox provide a tool called "Messageheader"[1] and MxToolBox provides an "Email Header Analyzer"[2] that provide this capability. There are numerous additional sites, as well as some offline tools that perform a similar function. Of course, any time an analyst uses an online tool, the potential impact to operational security, or OPSEC, must be considered. Using online tools with case data could be against laws or policies, but at a more fundamental level, could leak sensitive case data to unknown third parties.

Regardless of the site or tool used, such header analysis allows the analyst to provide SMTP headers which are then ordered chronologically and depict the flow and timing of each SMTP hop on the path from the sender to the recipient. This is useful for identifying common upstream paths of messages that may be part of a larger phishing or spearphishing campaign. It can also help to pick out anomalies such as large apparent delays and servers that are inconsistent with the message content or the rest of the pathway—potentially indicating falsified mail headers attempting to conceal the true source of a message.

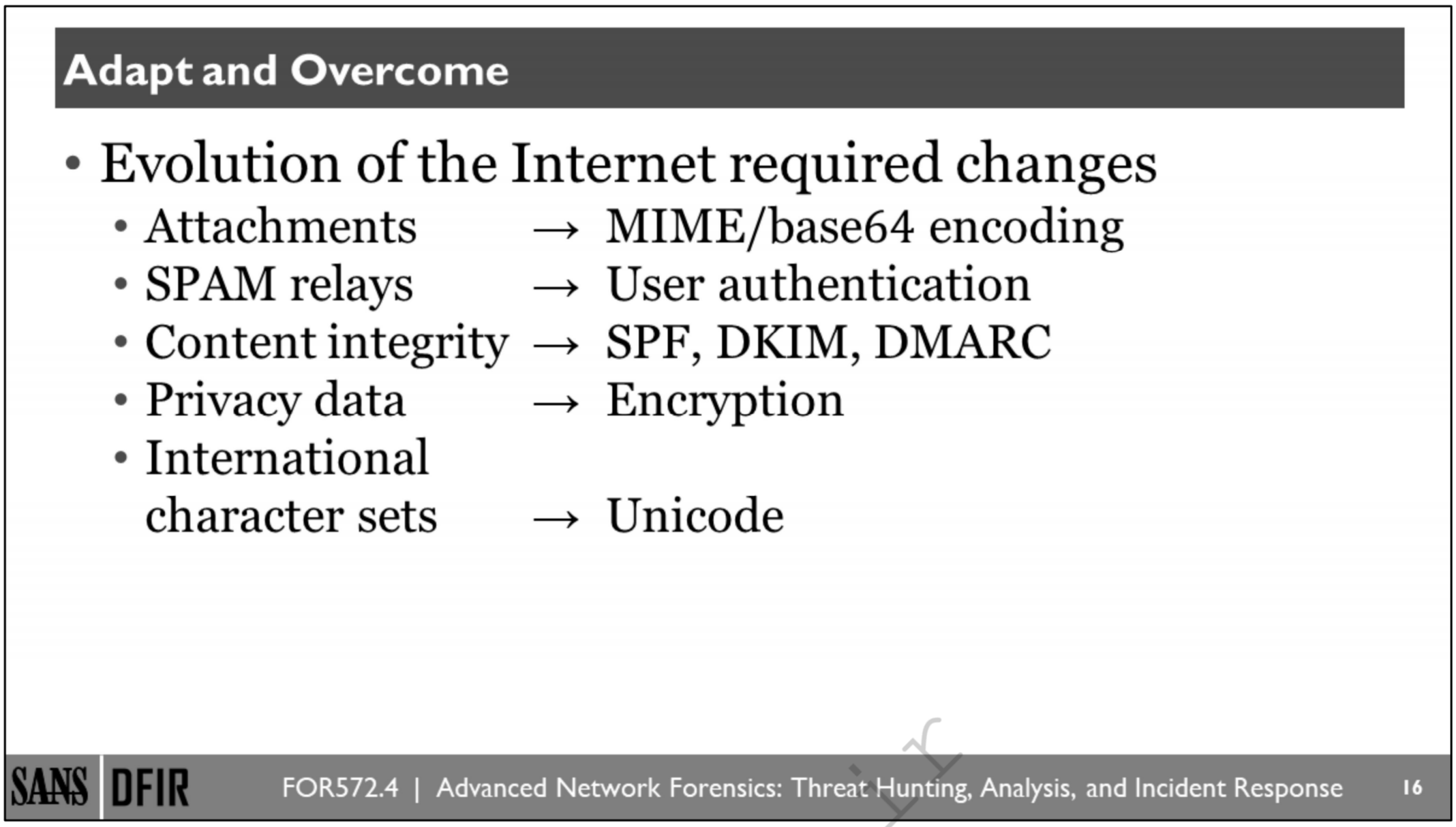
In particular, the MxToolBox Email Header Analyzer tool performs quite extensive parsing and reporting capabilities—far beyond just the "`Received:`" headers. This includes SPF, DKIM, and other email validation techniques, as well as logically grouping different categories of headers for easier analysis. Consider the Google header report shown in the screenshot above with the MxToolBox report of the same message.[3]

1. https://for572.com/google-mha
2. https://for572.com/mxtoolbox-eha
3. https://for572.com/p18im

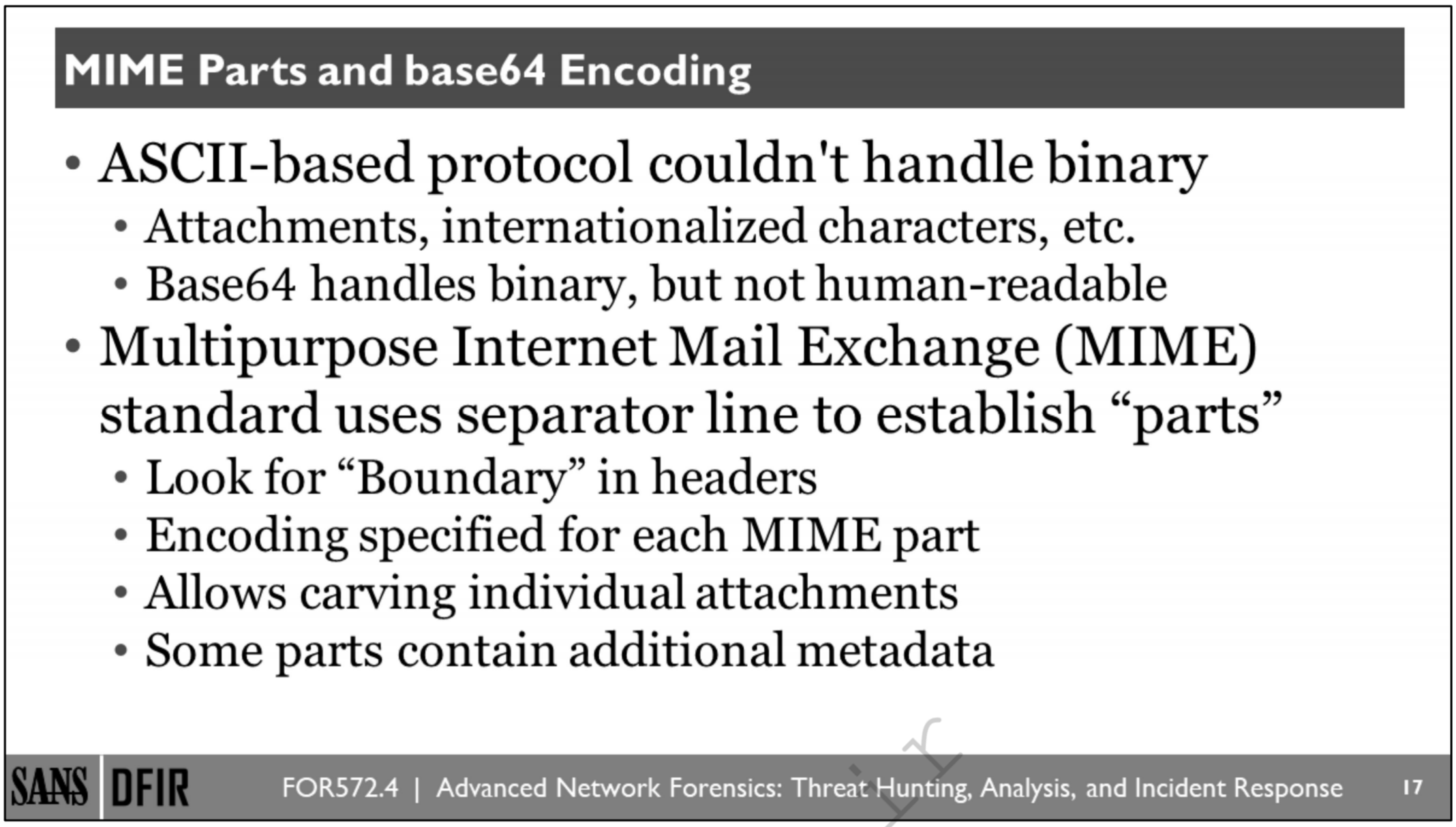
Google Admin Toolbox Messageheader

MessageId	9a4480fbc856bd4806951487b.bc686eee81.20230620160052.d1d811f9d9.c9f51a79@mail249.atl291.mcsv.net
Created at:	6/20/2023, 12:01:03 PM EDT (Delivered after 11 sec)
From:	Radinn - Electric Jetboards <marketing@radinn.com> Using Mailchimp Mailer - **CIDd1d811f9d9bc686eee81**
To:	<phil@identityvector.com>
Subject:	It's all in the wrist: The new Radinn Apple Watch app
DKIM:	**pass** with domain radinn.com Learn more

#	Delay	From *		To *	Protocol	Time received
0	6 sec	localhost	→	mail249.atl291.mcsv.net	ESMTP	6/20/2023, 12:01:09 PM EDT
1	2 sec	mail249.atl291.mcsv.net	→	bolt104b.mxthunder.net	ESMTPS	6/20/2023, 12:01:11 PM EDT
2	3 sec		→	simcoe.identityvector.com	ESMTPS	6/20/2023, 12:01:14 PM EDT



As we mentioned previously, SMTP has evolved significantly since its early incarnations. We'll take a look at each of these more significant modifications in turn.



Although it was great for (English) text, SMTP was never designed to handle that 175MB PowerPoint file, endless JPEGs of kitty cats, or even non-Western languages. To bring these now-fundamental features into existence, base64 encoding was selected. This encoding method represents arbitrary binary bytes in an all-ASCII character set consisting of 64 possible characters. However, dropping massive blocks of base64 text into the body of an email was not an option that most humans would be able to deal with.

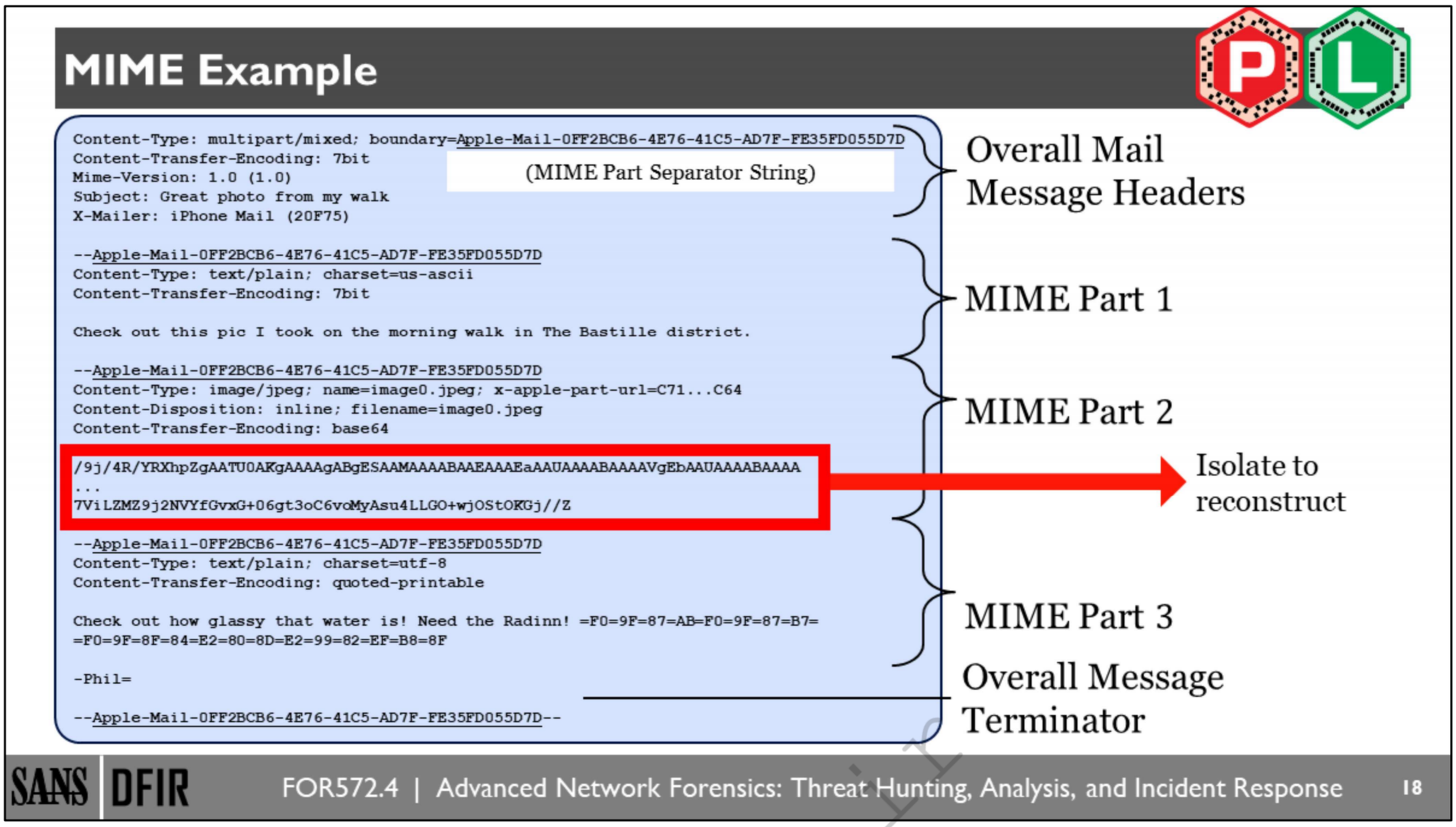
To address this, the Multipurpose Internet Mail Extension, or MIME, standard was adopted for use in SMTP. Email messages that include MIME data contain a "`boundary`" indicator in the headers. This string designates the beginning of each new MIME part in the overall mail message.

Each part can also include its own headers, which often contain useful metadata such as the attachment's filename, encoding method, etc.

By tracking these boundaries and referencing each part's header metadata, we can easily discern each subsequent message part, making for rather simple attachment carving.

References:

- `https://for572.com/9kof7`
- `https://for572.com/ap-86`
- `https://for572.com/0dry7`
- `https://for572.com/71qhr`



This slide depicts excerpts from an SMTP exchange including three MIME parts. The headers indicate the MIME version, but most importantly, the boundary string. This unique string will be used within this individual message to designate the start of a new part.

By looking into the message body, we see that each part consists of its own headers and body. Each part is preceded two hyphens and the unique boundary string. The headers may include encoding and other metadata, such as the encoding method used, an attachment's filename, and other information. The headers continue until the `0x0D0A0D0A` separator. After the four separator bytes , the part body continues until the next boundary string.

Each MIME part has its own headers, which help characterize the part's content. Note that the example's two text-based parts, the first and the third, indicate the use of "`Content-Transfer-Encoding: quoted-printable`". This method[1] establishes a means of representing 8-bit data bytes in a 7-bit space, a historical limitation of the SMTP protocol. Essentially, quoted-printable encoding replaces each source byte with an equal sign followed by the two hex characters of the source byte. The quoted-printable strings in the third part of this message reflect the encoded version of two Unicode emoji characters.

In the case of base64-encoded parts, we can simply extract the encoded section and decode the original attachment or content part to its original form.

Additionally, the "`Content-Disposition: inline`" header used on the JPEG in this example requests that the recipient's mail client display a thumbnail of the image instead of an icon. This may be useful for an attacker who wishes to trigger a vulnerability in a rendering library rather than one in a client-side helper application.

1. `https://for572.com/9otd8`

```
Content-Type: multipart/mixed; boundary=Apple-Mail-0FF2BCB6-4E76-41C5-AD7F-FE35FD055D7D
Content-Transfer-Encoding: 7bit
Mime-Version: 1.0 (1.0)
Subject: Great photo from my walk
X-Mailer: iPhone Mail (20F75)

--Apple-Mail-0FF2BCB6-4E76-41C5-AD7F-FE35FD055D7D
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

Check out this pic I took on the morning walk in The Bastille district.

--Apple-Mail-0FF2BCB6-4E76-41C5-AD7F-FE35FD055D7D
Content-Type: image/jpeg; name=image0.jpeg; x-apple-part-url=C71...C64
Content-Disposition: inline; filename=image0.jpeg
Content-Transfer-Encoding: base64

/9j/4R/YRXhpZgAATU0AKgAAAAgABgESAAMAAAABAAEAAAEaAAUAAAABAAAAVgEbAAUAAAABAAAA
...
7ViLZMZ9j2NVYfGvxG+06gt3oC6voMyAsu4LLGO+wjOStOKGj//Z

--Apple-Mail-0FF2BCB6-4E76-41C5-AD7F-FE35FD055D7D
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: quoted-printable

Check out how glassy that water is! Need the Radinn! =F0=9F=87=AB=F0=9F=87=B7=
=F0=9F=8F=84=E2=80=8D=E2=99=82=EF=B8=8F

-Phil=

--Apple-Mail-0FF2BCB6-4E76-41C5-AD7F-FE35FD055D7D--
```

(MIME Part Separator String)

Overall Mail Message Headers

MIME Part 1

MIME Part 2

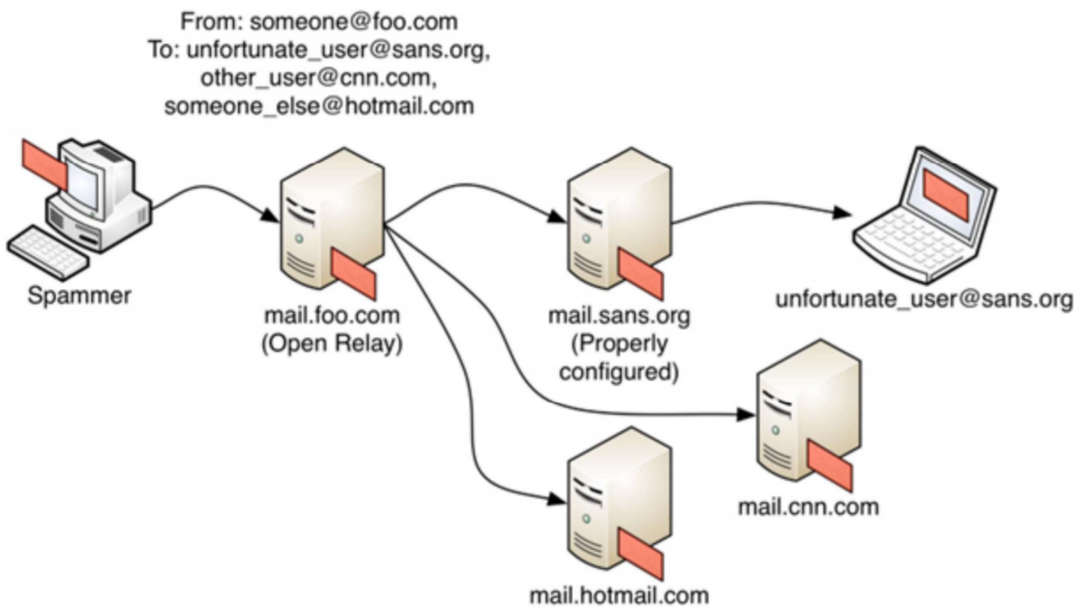
Isolate to reconstruct

MIME Part 3

Overall Message Terminator

User Authentication

- User authentication prevents SPAM relays



- Multiple user authentication methods
  - Plaintext (should require encrypted SMTP)
    - LOGIN, PLAIN, OAUTHBEARER: base64-encoded strings
  - Encrypted (generally considered "secure")
    - CRAM-MD5, DIGEST-MD5, GSSAPI, NTLM, etc.

Before spam became such a pervasive problem, there was no need to authenticate users who were requesting to send email. Server operators had a reasonable expectation that most or all SMTP transactions were legitimate, and MTAs unquestioningly passed email messages on toward their final recipients.

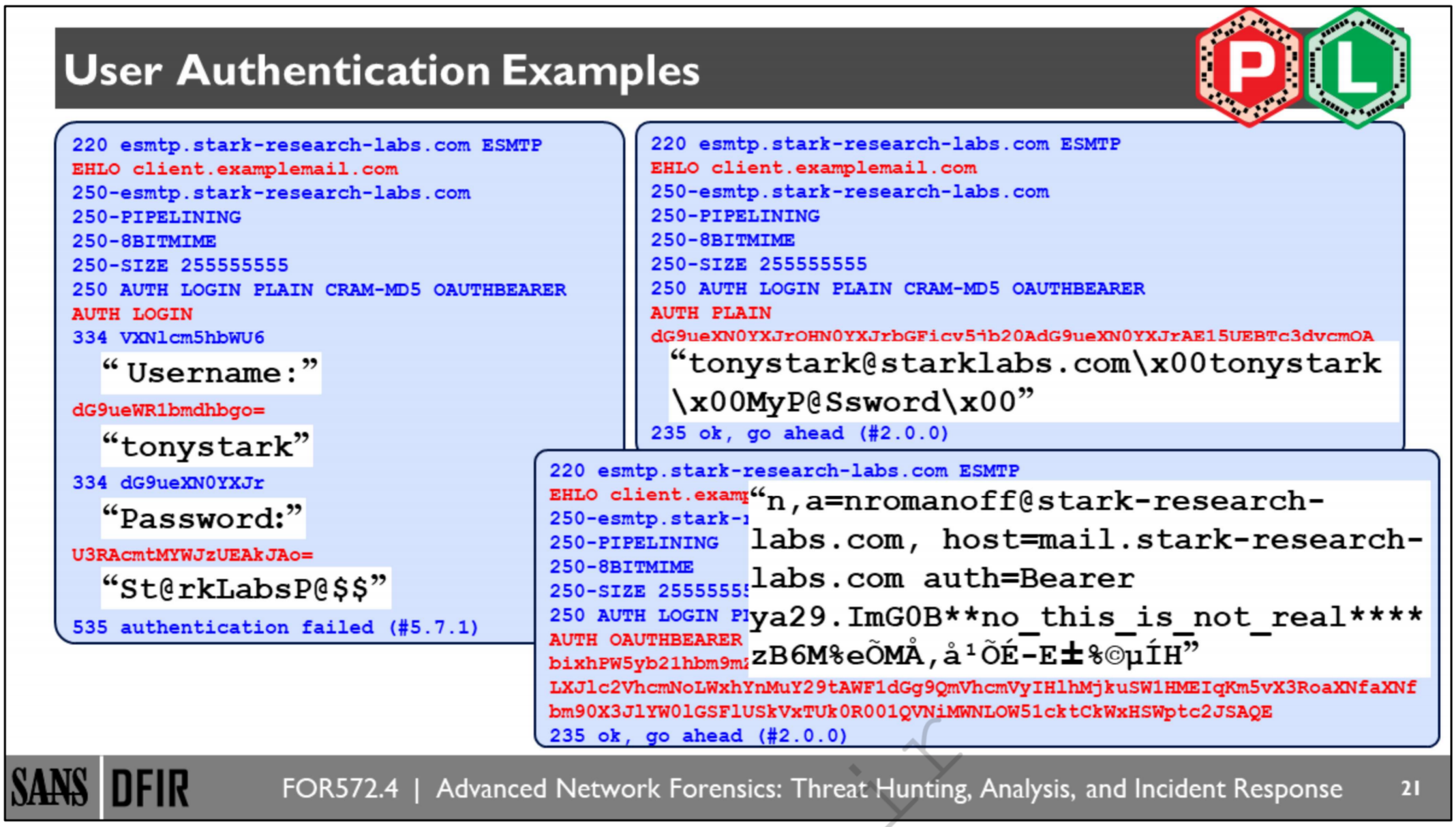
In the example illustrated on this slide, a spammer is using a poorly-configured SMTP server at "`mail.foo.com`" to relay a forged message from "`someone@foo.com`". Although this depicts just one message, even a small spamming operation can send many hundreds of thousands of messages per day using this methodology.

Obviously, that model was soon exploited by spammers, so RFC 2554 was created in 1999, establishing the SMTP-AUTH standard. (This RFC has since been replaced by RFC 4954.) SMTP-AUTH establishes multiple authentication methods, some of which are considered more secure than others.

The LOGIN and PLAIN methods are widely used. As you'll see in a moment, these are simple, consisting merely of base64-encoded username and password credentials. Remember: Encoded is not encrypted! These methods should never be used over unencrypted connections, and many SMTP servers will not advertise their availability unless TLS is in use.

A method commonly seen in modern configuration is the OAUTH family of authentication methods, used for numerous mail services including Gmail. This method involves a granted token that can then be revoked if and when needed. Although this token does not include the account's password, it is also just a base64-encoded string representing a self-contained authorization token and should be considered plaintext.

Because of this major security shortfall, the RFCs also created a number of authentication methods that provide a greater level of security. These are generally considered acceptable even over non-TLS connections because they provide their own cryptographic protections. Although we won't examine all of these methods in detail, it's important to note that there are multiple authentication methods, and the two encoded methods can provide valuable credential evidence for the investigator.



Shown here are examples of several "insecure" authentication methods we discussed on the previous slide—`LOGIN`, `PLAIN`, and `OAUTHBEARER`. As shown in the server's capability listing, the `esmtp.stark-research-labs.com` server can perform authentication using all three of these methods plus `CRAM-MD5`.

After the client requests the `LOGIN` method, the server responds with the base64-encoded string "`Username:`". The client responds in kind with the encoded username. The parties then perform a similar exchange for the password. In this case, the authentication attempt was unsuccessful.

In the second example, the client requests the `PLAIN` method. For this method, the client can immediately send an encoded string with either two or three components, each terminated with a null byte. These components are:

- Authorization identity (optional): The entity as which the client is attempting to authenticate.
- Authentication identity: The username
- Password

In the second case, the authentication request was successful, so the client would then initiate with the email message submission process.

In the final example, a client requests the `OAUTHBEARER` method, which consists of a unique string allocated by the mail provider to a given user account. For our purposes, this should be considered like an alternate password for the account. Any party with this bearer token can access the account to which it is associated. The exact makeup of these tokens is solely at the discretion of their issuers.

References:

- `https://for572.com/37zjq`
- `https://for572.com/kw7z3`
- `https://for572.com/tfxkm`

Content Integrity Validation

- With no inherent source of authority, SMTP allows trivial spoofing of senders, content, and more
- Several mechanisms designed to mitigate that risk
- All use DNS TXT records to provide direction

Validation Method		Function
SPF	Sender Policy Framework	Identifies IP addresses permitted to send mail on behalf of a given domain
DKIM	DomainKeys Identified Mail	Public key cryptographic signature to assure integrity of the From: header and optionally others
DMARC	Domain-based Message Authentication Reporting and Conformance	Requests that receiver report SPF or DKIM failures

Since the original implementation of SMTP had no consideration for how to detect spoofed senders, message content, or really any component of the mail message, additional techniques were developed that work in parallel to the SMTP transaction itself. There are three primary mechanisms that are designed to mitigate the risk of spoofed email traffic and while they are independent of each other, they are generally implemented in parallel by most mail service providers or server administrators.

All of these methods use DNS TXT records for their Internet-facing implementations. The Mimecast company has an online tool that parses, validates, and explains these TXT records.[1]

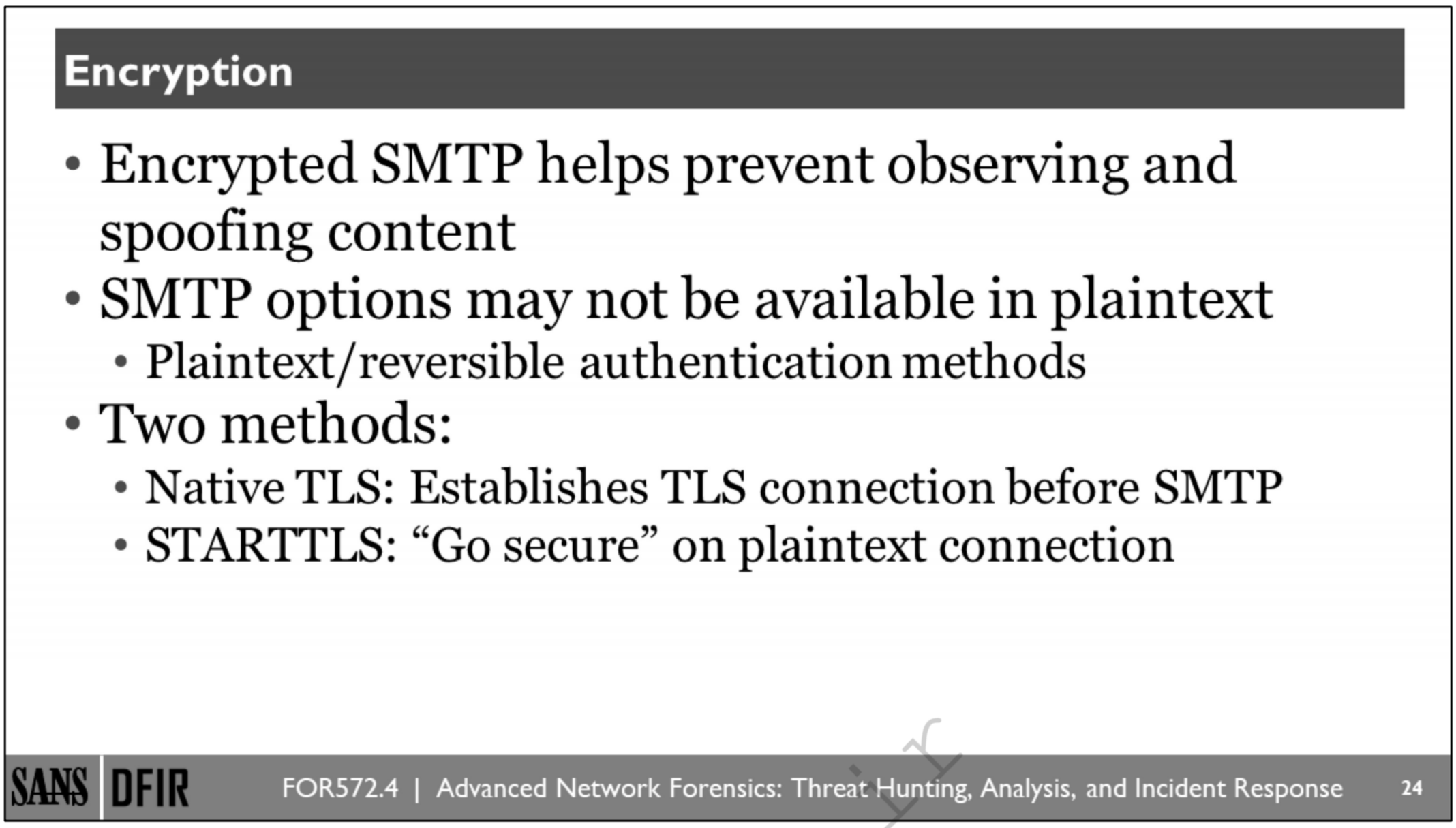
- SPF, or Sender Policy Framework[2], is a method for a domain owner to designate the IP addresses that are authorized to send email claiming to be from the domain. An MTA or MX server may or may not add a header to indicate the success or failure of the validation.
- Example TXT record content:
  `v=spf1 a mx ip4:70.32.97.206 include:spf.spamhero.com ~all`
- Example validation header generated by receiving server:
  `Received-Spf: pass (google.com: domain of 003-yru-314.0.96664.0.0.8251.9.1472502@em-sj-77.mktomail.com designates 199.15.215.82 as permitted sender) client-ip=199.15.215.82;`

- DKIM, or DomainKeys Identified Mail[3], uses a public/private key pair to generate a signature that permits validation of certain mail headers. The "From:" header is the only one that must be signed if DKIM is implemented, but an email service provider or administrator can configure any additional headers to be signed as well. The TXT record contains the base64-encoded public key that corresponds to the private key the server uses to generate the signatures, which are added to the headers by the originating server. Multiple Dkim-Signature headers may exist, especially if the message was sent from a subscription list service. The receiving server also may or may not add a header to indicate the success or failure of the validation.
    - Example TXT record content (truncated for space):
      `v=DKIM1; k=rsa; p=MIIBIjANBgkqhkiG9w0B...AQEFAAOCAQ8AMIIBCgQIDAQAB`
    - Example header generated by sending server (truncated for space):
      `Dkim-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed; s=e2ma; d=e2ma.net; h=Content-Type:MIME-Version:Subject:From:To:Date:Message-ID:Reply-To: List-Unsubscribe:Feedback-ID; i=@mail65-4z9c.e2ma.net; bh=L6dVBoJ...ZtHcC34xbCE=; b=Ptw5... B/ZDQ +dvZgL...txeo=`
    - Example validation header generated by receiving server:
      `Authentication-Results: simcoe.identityvector.com; dkim=pass (1024-bit key) header.d=e2ma.net header.i=@mail65-4z9c.e2ma.net header.b="Ptw5EKhQ"`
- DMARC, or Domain-based Message Authentication Reporting and Conformance[4], is slightly different than SPF or DKIM in that it does not provide validation functionality itself. Rather, DMARC allows a domain owner to request that any server that encounters an SPF or DKIM validation failure send an emailed report of those failures for further investigation. This allows the domain owner to identify either SPF or DKIM records that are misconfigured, as well as if any messages are being maliciously spoofed from their domain(s).
    - Example TXT record content:
      `v=DMARC1; p=none; sp=none; rua=mailto:postmaster@identityvector.com; ruf=mailto:postmaster@identityvector.com; rf=afrf; pct=100; ri=86400`
- Validation headers may also be consolidated into one record, which may exist by itself or in addition to the individual validation headers.
    - Example consolidated authentication header:

```
Authentication-Results: mx.google.com;
       dkim=pass header.i=@redcanary.com header.s=m1
header.b="KVg9du/Y";
       dkim=pass header.i=@mktomail.com header.s=m1 header.b=KKP8pX0v;
       spf=pass (google.com: domain of 003-yru-
314.0.96664.0.0.8251.9.1472502@em-sj-77.mktomail.com designates
199.15.215.82 as permitted sender) smtp.mailfrom=003-YRU-
314.0.96664.0.0.8251.9.1472502@em-sj-77.mktomail.com;
       dmarc=pass (p=QUARANTINE sp=QUARANTINE dis=NONE)
header.from=redcanary.com
```

These technologies can also aid investigators in determining the likely validity or lack thereof for a suspicious message, as well as to expose the sender's infrastructure and configuration. Additionally, the reliance on DNS records is important to consider-–especially in DNS logging strategies.

1. https://for572.com/byvuj
2. https://for572.com/186n2
3. https://for572.com/8325g
4. https://for572.com/3962i

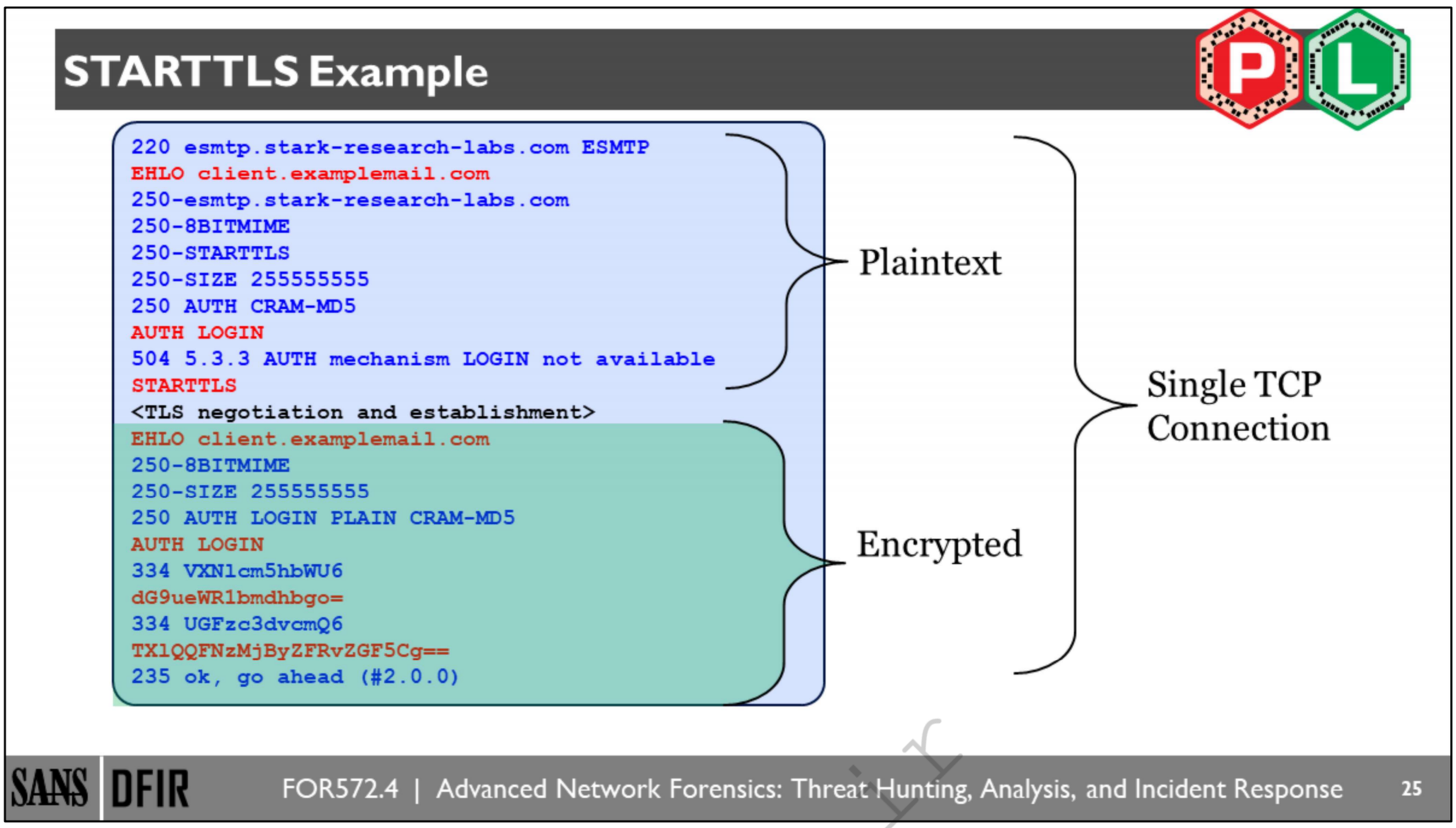


As you know, encryption can provide both confidentiality and integrity, depending on how it is implemented. We'll discuss encryption at great length later, but let's briefly touch on its application within the context of SMTP.

Considering the inadequacy of the LOGIN and PLAIN authentication methods to protect the user's login credentials, encryption becomes very important in most SMTP transactions. Most SMTP servers will not even advertise these two methods unless the connection has been secured.

There are two primary mechanisms of securing the SMTP connection: TLS and STARTTLS. Although not as common today as it once was, the TLS mechanism starts the new TCP connection with an immediate exchange to encrypt the connection. After it is secured, the SMTP transaction starts.

On the other hand, with the STARTTLS mechanism, the connection starts as a standard, unencrypted one. The client requests that the server “goes secure”, at which point an TLS negotiation occurs over the same connection. The benefit in using STARTTLS is that a separate port doesn't need to be allocated and managed across an architecture. In this model, an SMTP server may present different capabilities on the same connection, depending on whether a STARTTLS command has been issued and the negotiation completed.



Here, you can see an example of the STARTTLS process. The client system connects to the esmtp.stark-research-labs.com server, which advertises that it can use the STARTTLS mechanism as well as authenticate using the CRAM-MD5 method.

The client initiates the negotiation, and everything shown in the shaded region represents the conversation that occurs over the encrypted channel. The key point here is that the same connection is used throughout.

During the encrypted phase of the conversation, the server advertises that it will accommodate the LOGIN and PLAIN methods in addition to the CRAM-MD5 method. The client then selects the LOGIN method and proceeds with the SMTP transaction.

Reference:

- https://for572.com/1unt8

Unicode in Mail Message Headers

- Encoding can be changed as needed in headers
  - RFC 2047: `=?<charset>?<encoding>?<data>?=`
  - Mail clients usually render the decoded version
  - Some forensic tools may show native transported form

```
From: =?US-ASCII?Q?Nick_Fury?= <nfury@stark-research-labs.com>
To: =?ISO-8859-1?Q?J=F8rn_S=F8rensen_?==?UTF-
    8?Q?=F0=9F=87=A9=F0=9F=87=B0?= <jorn@wakanda-tours.dk>
Subject: =?UTF-8?B?UGxhbm5pbmcgZm9yIHZhY2F0aW9uIHRvIFdha2FuZGE=?=
```

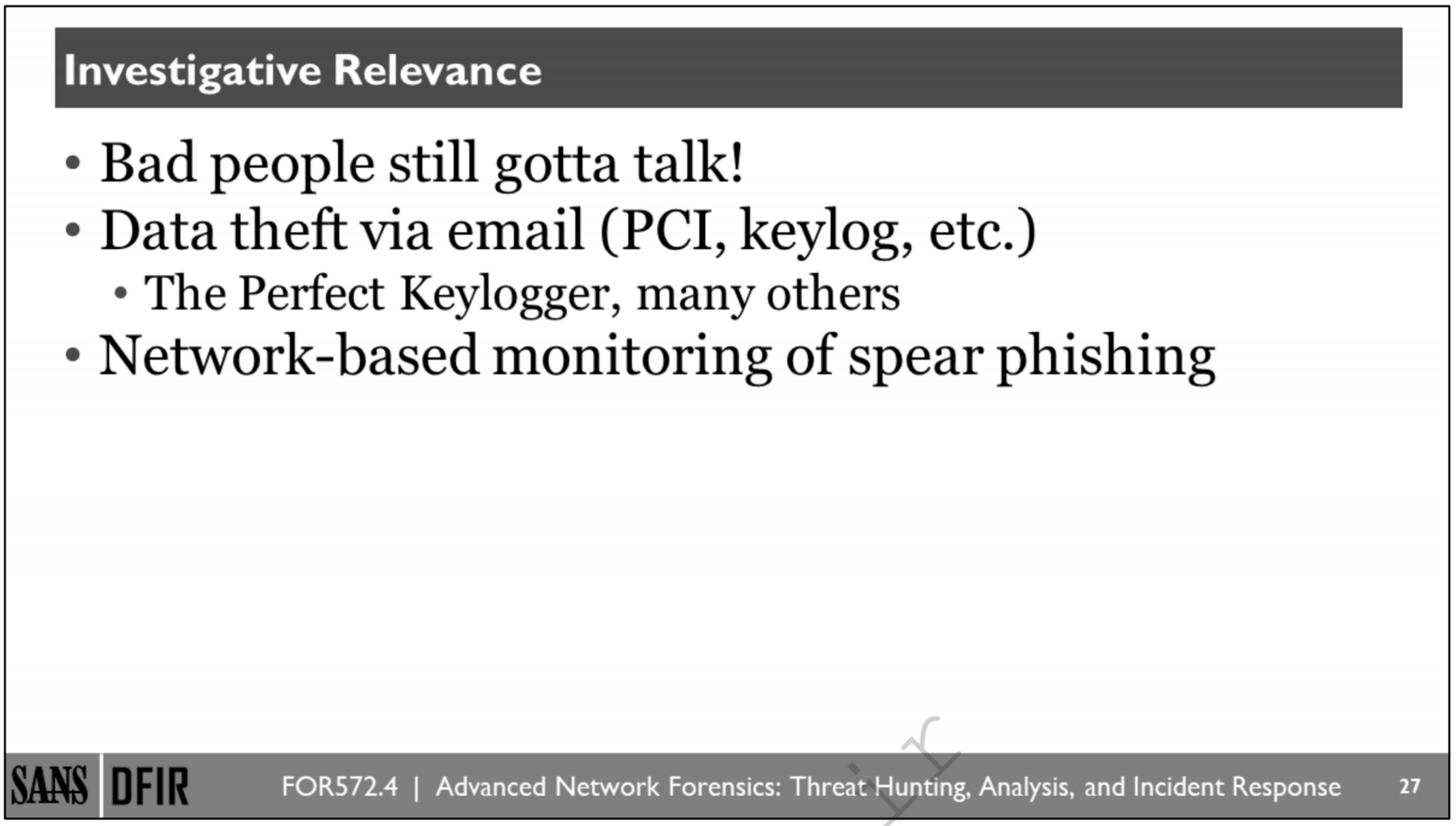
```
From: Nick Fury <nfury@stark-research-labs.com>
To: Jørn Sørensen DK <jorn@wakanda-tours.dk>
Subject: Planning for vacation to Wakanda
```


Since SMTP still exclusively uses US-ASCII for its content transport, special handling must be used for any characters outside of that set. While modern mail client software will render the text as it is meant to be read, looking at the version in flight or at rest in a mail spool can provide an exercise in decoding.

As seen in the example above, even the US-ASCII "`From:`" header is caveated as such even though no replacements were needed. The "`To:`" header, however, shows that it is partially encoded with ISO 8859-1, often referred to as the "`latin-1`" character set because its 256 characters generally provide coverage for a wide variety of Latin-based languages. There is a second portion of the "`To:`" header in UTF-8 as well, representing an Emoji character. Finally, the example "`Subject:`" header, shows that any Unicode-defined character set can be used, with both base64 and UTF-8 encoding shown.

This encoding standard is defined in RFC 2047[1], and an online decoder[2] may be helpful for testing purposes. However, for any sensitive case data, an offline equivalent should be used.

1. https://for572.com/vbwe9
2. https://for572.com/502uv



Of course, we're most interested in knowing how SMTP is relevant during an incident response or investigation. After all, SMTP is a rather old protocol, and probably not one that most malicious actors use… or do they?

One of the key things to remember is that bad guys still have to communicate somehow! In many cases, they'll use what's available—such as email. Although it's unlikely that a nation-state actor would use email within a victim's network to coordinate operations, a rogue employee might certainly use email as a communication mechanism, or possibly a means of sending privileged information outside of the network.

In other pockets of the pure criminal world, software keyloggers are still used quite efficiently. In fact, many credit card readers are no more than keyboard devices from the computer's perspective, so a keylogger on a point-of-sale or cash register system is a common tactic for PCI data theft. Keylogger software can be configured to send captured data to an external email account, providing attackers with an easy data theft collection mechanism.

Another key presence of SMTP in the modern threat environment is the pervasive use of spear phishing. There is no discounting the fact that this attack method is both widely and efficiently used by strategic attackers to gain footholds within their targets' networks. Of course, the penultimate leg of every such message is across SMTP—right before the MDA puts it into the Inbox of their target's client software. The ability to examine SMTP exchanges can provide useful insight into the flow of spear phishing messages and their payloads. Understanding the nature of SMTP headers can show the hops a message took in getting to its destination, and knowing how they can be forged or altered can help to keep “red herring” findings in check. Using the MIME standard to effectively and efficiently carve attachments can provide a reverse-engineering team with the initial dropper binary that an attacker used to gain access to the target's environment. Of course, analysis of this dropper may very well lead to additional network indicators that can be used to scope an incident and start to plan toward remediation.