[WP-010]

Whitepaper: I Know Something You Don't Know (Email Secrets)

A Theoretical Discussion Concerning Secretive or Covertive Methods of Communication via an Unchecked Feature of the SMTP Protocol

Version 040727

June 2004

Author: Bob Radvanovsky, rsradvan@unixworks.com

(A special thanks goes to those listed for being my "sounding board" on this project.)

Limited Liability Statement

In no event shall the author(s) be liable for any found errors contained herein or for any direct, indirect, special, incidental or consequential damages (including lost profit or lost data) whether based on warranty, contract, tort, or any other legal theory in connection with the furnishing, performance, or use of this material.

The information contained within this document may be subject to change without any notice. No trademark, copyright, or patent licenses are expressly or implicitly granted (herein) with this whitepaper.

Documentation pertaining to any security-related technical or proprietary information, its data and all information provided and contained within this document is considered proprietary in nature and subject to copyright protection and is intended solely for use by its owner. Additionally, this documentation is solely for the purpose of discussing a possible, proposed infrastructure issue, and is not dependent upon any specified infrastructure, architectural condition or its issue(s).

No portion of this document is intended to promote, disable, destroy or alter the energy transmission lines at any location, whether within, throughout or out of the United States; instead this paper is intended to identify methods by which will reduce the possible risks associated with the operation of Internet-based public communications channels.

Under the United States Patriot Act of 2001, Title VIII, Section 802(a)(5), no portion of this document is intended to involve activities that:

> (A) involve acts dangerous to human life that are a violation of the criminal laws of the United States or of any State;
> (B) appear to be intended--
> - (i) to intimidate or coerce a civilian population;
> - (ii) to influence the policy of a government by intimidation or coercion; or
> - (iii) to affect the conduct of a government by mass destruction, assassination, or kidnapping; and
>
> (C) occur primarily within the territorial jurisdiction of the United States.

Any such information is intended for *"educational purposes only"*, and is intended solely for the necessary recipients thereof; no other information will be provided that will demonstrate any such acts deemed as "domestic terrorism" under the laws applicable from within the U.S. Patriot Act of 2001.

All other product names mentioned herein are trademarks or registered trademarks of their respective owners. NOTE: Any names not outlined or mentioned above are fictional in nature; as such, any relation to any name or trademark (if any) is purely coincidental.

Introduction

Current electronic mail transfer methods utilize an unsecured method of transferring messages between two (or more) mail servers. The exchange method for each message involves transmission based upon criteria in which the method, by how and how much, determines how the messages are exchanged between each mail server and how they are routed. The initial specification of the SMTP protocol dates back to August 1982, in which RFC 821 specifies the requirements by how the mail servers might exchange messages. The protocol was later revised to accommodate for additional types of traffic via electronic mail (such as MIME headers, file extension type definitions, etc.), which was RFC 2821 as of April 2001 (this obsoleted RFC 821).

To understand what is being conveyed, the user must first have an underlying fundamental knowledge of what and how electronic mail messages are transferred and how they function. To an end-user, they simply supply a valid email address (rsradvan@unixworks.com), enter the subject text in the "Subject:" area, followed by the actual body of the message (which can now include special language text, graphics and sounds), then press the "Send" button to send their mail message to its destination (one or more recipients). Upon pressing the "Send" button on their offline email reader, the message is first sent to the POP server, then finally onto the mail server. As with many end-user configurations, most end-users utilize an offline reader (such as Qualcomm's Eudora or Microsoft's Outlook) to send and receive their email messages. The offline readers interface with the mail server via another server, called a "POP server". "POP" is short for "Post Office Protocol", which is another networking protocol used to authenticate the end-user wishing to send (or receive) an email message; and once authenticated, the POP server transmits the email message to (or from) the mail server for final delivery. The mail server then validates the header, adding or subtracting header information (as needed) prior to transmission, then routing the email to its final target destination.

A sample email header might look something similar to what is shown below:

```
Return-Path: <rsradvan@unixworks.com>
Received: from lappie (laptop.intranet.unixworks.net [192.168.1.90])
  by srvr1006.unixworks.net (8.12.11/unixworks.net) with SMTP id i5UHr5J3031807
  for <rsradvan@unixworks.com>; Wed, 30 Jun 2004 12:53:05 -0500
Message-ID: <020501c45ec9$ec840910$040bfa0a@unixworks.net>
Reply-To: "Bob Radvanovsky" <rsradvan@unixworks.com>
From: "Bob Radvanovsky" <rsradvan@unixworks.com>
To: "Bob Radvanovsky" <rsradvan@unixworks.com>
Subject: TEST MESSAGE
Date: Wed, 30 Jun 2004 12:44:36 -0500
Organization: UNIXWORKS.NET
MIME-Version: 1.0
Content-Type: text/plain;
  charset="iso-8859-1"
Content-Transfer-Encoding: 7bit
X-Priority: 3
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 6.00.2800.1409
X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2800.1409

THIS IS A TEST MESSAGE.
```

This is how email is transferred from one mail server to another. Within large corporate networked environments, there usually are several "hops" between the sending mail server and its final destination, in which either another mail server acting as a gateway from one source to another, or a firewall or content filter acting as the initial mail server to review email content, will most undoubtedly act as a filtering spam agent. Thus, this method may be used to protect corporate intellectual property and its assets, as it will intercept all mail messages going into and out of these corporate environments.

As outlined within RFC 2821, the email header is used (mostly) for post facto issues; that is, the email header is used for forensics reasons, often to verify email delivery, IP addresses and domain names. If there are any mistakes, errors (intentional or otherwise), will generally show (or not show, depending upon the circumstances) within the email header. Information that isn't entirely what it is supposed to be, or appears to be false is often called a "spoofed header", signifying that someone has falsified to an end-user into thinking that the email received is valid and from a reliable source. Today, spam is a growing concern for everyone connected to the Internet, and continues to plague just about everyone who sends email onto, across or through the Internet. Email addresses may be harvested by automated processes, then compared and checked against databases and stored for later use. These addresses may be used to push undesirable or wanted products that no one really needs or wants (body-part enlargements or reductions, sexual activity enhancers, et. al). Spammers may attempt to defraud people into providing critical, private personal information to a fictitious individual, organization or entity (classic one: rich executive or king dies and they need your help to transfer funding into a bank account, but will need your bank account number to do it – see where this is going?).

Are There Hidden Intentions?

Of course, the obvious ramifications are limited to a few cultural reasons as to why someone might want to falsify an email header. There were several initial thoughts about how email message headers may be falsified (majority are for less-than-honorable intentions), in which the intention of crafting an email header may be utilized for:

- Establishing communications channel for illegal purposes (such as pirated software)
- Establishing communications channel for unethical purposes (disabling pirates from utilizing pirated/hacked copies of a software company's product)
- Hiding an identity of an individual for political reasons
- Hiding corporate intellectual property (corporate espionage)
- Hiding government secrets and/or mission plans (covertive intelligence)
- Locating telemetry data (satellite, GPS, longitude/latitude specifications)
- Testing an application for possible flaws (quality assurance and measurement)
- Sending email "caller ID" – refer to Trusted Computing Platform Architecture (TCPA)

The utilization of such an application is very specific to its use, one that would be tailored either for a corporate entity (such as corporate espionage) or government use (covertive intelligence). Nonetheless, the implications are potentially serious enough to warrant further investigation as to why someone would want to hide a message, in plain text, unhidden, within an email message.

In most circumstances, the use of steganography is applied for hiding a simple text message (can be a few words, a small phrase or entire sentence) within the plain text email message. However, not all applications would utilize this form of crafting, as another negative aspect of this function is deployment of an activation (or perhaps de-activation) code to disable or render useless pirated/hacked illegal copies of a software company's product. For example, Microsoft might send a hexadecimal string embedded within an email that appears to be valid from Microsoft Corporation (such as a product announcement) to render useless any environment not properly licensed with or through Microsoft. A prime example may be the history involving Microsoft-pirated software distributed throughout China could be disabled utilizing such a method, or activate an automated, hidden removal code if an unlicensed product is not activated within a predetermined timeframe, etc. The possibilities of license enforcement now traversing the Internet employing this unchecked feature are huge, thus reinforcing Microsoft's position of ensured its market viability.

Conversely, government agencies could employ similar mechanisms that would cause internal memorandums in electronic form to be deleted when a specific activation code is received via the offline email reader (think similarly to the Mission Impossible introduction: "*This message will self-destruct in 5 seconds …*"). This would ensure that government secrets have a shortened life expectancy, esp. for classified documents, et. al. Mind you, there are mechanisms already implemented (e.g. Microsoft Outlook 2003), but this may be used as an alternative method – just in case – and could be used on architectures older than Windows Server 2003 or Windows XP.

I'll Have My Stego 'Medium-Rare'

There has been the term of "plain-text encryption" more often referred to as a form of encryption that utilizes unencrypted text, often over unsecured communication channels, utilizing a symmetrical key to encrypt or decrypt a message, file or text block (NOTE: a symmetrical key is used at both encryption and decryption of the data or message). This term may also be referenced for encryption that is in "plain text"; that is, the message itself, although appearing to be a sentence (either gibberish or coherent) is actually an encrypted message. Numerous movies have outlined one or more plausible methods of encryption, and the rules to perform such levels of encryption are highly complex and extremely secretive (unconfirmed, just assumed that it is possible considering that much of the technology that was referenced within the movie, "Enemy of the State", did in fact at one in time, exist). For further clarification, the term "text-based steganography" will be used instead of "plain-text encryption" for the remainder of this whitepaper.

The term "steganography" is both an art and a science, in that it's sole intention is to hide messages in clear view; that is, steganography's whole purpose is to obfuscate and confuse someone reading your messages or viewing your pictures into thinking that they do not contain any messages within them. Essentially, steganography is "camouflage" for data.

Steganography is the method by which to conceal the presence of an encrypted message within another message. Historically, several techniques have been used to hide information as this method of concealment has dated as far back as the 12th century. Several creative individuals have used secret compartments in objects, invisible ink, microdots and grilles to hide letters amid innocent text, some of which are now considered famous today (Leonardo DaVinci was one such individual that used quite a bit of messaging concealment in his works, both within his artwork as well as his scientific discoveries). Today, encrypted messages may be hidden as imperceptible noise within graphics and sound files; these two formats are the more common as there are places within those files that have unused sections where bits may be stored to contain textual data within them.

However, text-based steganography is somewhat tricky, if not difficult, to produce a valid and usable method of encrypting, then hiding a message within another message, both of which are textual data. An advantage for text-based steganography is that it does not expand the message size, nor does it significantly increase the size of the file, both of which are considered weaknesses of image-based steganography.

How does this affect electronic mail messaging capabilities?

Simply put, it is possible to conceal several messages within the electronic mail header through "packet crafting", such that mail servers, content filters (firewalls, intrusion detection systems, et. al) allow and pass through email messages – with these concealed messages within their email headers – without ever knowing that code has been passed onto a target destination. The inherent risk isn't so much that the message was passed through, but that it was passed through undetected. This poses a serious risk and can have potentially devastating consequences if exploited. To put the risk into another perspective, since just about everyone utilizes the SMTP protocol in some fashion or another, a carefully crafted email header could pass through, undetected, to its final destination. During its transmission, it is either carrying a secret message (though small, it would still be possible to send one) or carry an activation code to initiate hidden, remote code within a commercial-grade software product. Thus, the crafted email header isn't a steganographic method, nor an encryption device; but rather it a transport mechanism. The lovely thing this circumstance is that stripping such headers removes valuable auditing capabilities (such as tracing an email message back to its originating location); henceforth, the email header needs to remain intact, untouched, as modifying, removing, or tampering with the email header could have a serious impact.

How the email header may be utilized for such concealment is through a method called "packet crafting", in which generation of network packets for non-common protocols are encapsulated within (TCP|UDP)/IP stack, usually for testing purposes. This type of packet generation is considered (almost) as an art form as the individual would require a significant amount of knowledge and experience with the network protocol in question. This art form is more often used for diagnostics purposes such that networking and security engineers attempt to re-create networking packets that were used to exploit a server (such as through a denial of service attack, backdoor attack, etc.). Packet crafting is also used to measure and perform threshold analysis of systems (think of it as a modified form of "unit testing"), or the need to confirm that devices communicating over encapsulated networking protocols for the sensitivity to various malformations of the packets and to see if they are secure and accurate. Packet crafting would enable either finding – or exploiting – unknown, unused or undisguised methods of modifying the header and payload packets of an encapsulated networking protocol (such as SMTP).

So how does "packet crafting" have anything to do with text-base steganography encryption via electronic mail? As stated before, there are numerous sections within the electronic mail header(s) that would enable the emplacement of data (or its code) that could be used to convey, relay and transfer data and messages without visibly being noticed.

There are several methods by which a message or encryption key may be hidden within the email header, or even the payload itself. So far, this concept has been tested with a crafted email header sent via several mail server software packages. As stated within RFC 2821, the mail server looks for a valid email address within the first time stamp header (looks like "**Received by …**"). If the first time stamp header does not appear to be valid, then the mail server checks the next time stamp header, and so on and so forth. As long as the email header appears to be valid, and the sending server can be verified through DNS lookup (depending upon which flag options have been set within the mail server), then the mail server passes whatever header and payload through onto the receiving mail server.

A sample portion of the email header that is modified is shown below:

```
Received: from srvr1006 (srvr1006.unixworks.net [64.32.214.198])
  by srvr1006.unixworks.net (8.12.11/unixworks.net) with SMTP id i5UHr5J3031807
  for <rsradvan@unixworks.com>; Wed, 30 Jun 2004 12:53:05 -0500
Message-ID: <1324413214321243@unixworks.net>
```

NOTE: The modified section of the email header is shown in **BOLD** and **yellow**.

In most circumstances, it is not necessary to provide the "**Return-Path:**" time stamp within the email header. Note that the "**Received:**" time stamp within the email header points to a valid server (in this example, this is the SMTP server for the "**unixworks.net**" domain), which should include both the name AND the IP address of the sending server. Only the server name (not FQDN), the full server name (as a FQDN), and the IP address; are require; however, in some circumstances, simply the server name and IP address are sent. The IP address is always required. An "FQDN" means “Fully Qualified Domain Name”; for example, “**srvr1006.unixworks.net**” is an FQDN, the name “**srvr1006**” represents the server name, the remainder, “**unixworks.net**” is simply the domain name.

Please note that the areas (shown in yellow) have been crafted, and appear to be valid. As long as the source mail server can be confirmed as being valid, then the email will be passed along. Note the “**Message-ID:**” header where the concealed message payload, activation key, encryption key, or bit shift key may be transported. This is where the risk is involved, esp. when a smart client recognizes any of these suggested transport methods.

Validity for the Crafted Header

For this whitepaper, the intention is a specified, (hopefully) undeterminable method for hiding a message in an unsecured, unencrypted text (the definition is actually a play on words; the term “text” can include: alphabetic characters [small and large], numeric characters and special characters [question mark, exclamation point, period, etc.]). In today’s networked world, no methodology is safe from prying eyes as being perceived as “unbreakable” (although the federal government would like to state otherwise). The architecture outlined within this whitepaper is a proof-of-concept in that it demonstrates the ease of which an encapsulated networked protocol may act as a transportation mechanism for the steganographic method used. The reasoning for that is it may contain the hidden, concealed message within it within the header or payload of the packet itself.

The method by which the steganographic concept will be performed will require a custom-written application (proof-of-concept of this special application is currently being developed) to act as the sending mail server. Under normal circumstances, the steganographic methodology will generate the concealed message, wrap the message within the header, bundle the header and payload of the email message, and then ship the message to the mail server for final delivery (or act as an out-going only mail server to “ship” the payload to its target destination). Most mail servers won’t notice that there was a crafted email message passed onto them, as the first line within the email header is valid and confirmable.

How It All Works Together

The methodology may be explained through a series of steps within the entire process. The proof is limited to a single small word (less than 8 characters), or a group of words (say 3 words, all words combined with their letters, minus the spacing, are 30 characters or less). The process will require that an external text source is provided for embedding the hidden text within the steganographic message. Actually, the un-steganographic message remains intact, merely that the steganographic message are references within the external text source, sort of plotting a line along a map – the map remains the same, but the course and its direction may be altered based upon the trajectory.

The proof-of-concept is not a foolproof method – it is flawed. However, this is a demonstration of its ability to send/receive steganographic messages within an electronic mail messaging payload, thus reinforcing the method of embedding other methods of covertive communications, such as activation or deactivation methods of hidden subroutines within offline applications, or lesser utilized methods for such methods previously outlined. The proof-of-concept is outlined within the next several paragraphs.

The hidden text source must not exceed 30 characters. The maximum character length is the maximum number of characters (30 characters) times 2 (the proof-of-concept will be represented as a decimal string). An arbitrary number of 40 times the maximum number of characters of the hidden text is what will be needed to “hide” the steganographic message within the electronic mail message payload; this number may be more or less, and is arbitrary to the proof-of-concept outlined. For this proof, there will be only one (1) steganographic message; however, since this proof-of-concept does outline a maximum character length of 30 characters, then a complete sentence of 100 (or so) words will be broken down into separated messages that are pieced apart, then pieced back together. In this case, there is only one (1) message, so the initial part of the string begins with "0101", signifying Message #01, Part #01. If the message length were to exceed the 30 character length limitation (not including spacing within the hidden text message), and this was part of the pieced-together message, there would be a "0102", "0103", and so on and so forth for each additional steganographic message. The external text source is translated into uppercase text only. The external text source is parsed into manageable "chunks" for easy ingestion of the hidden text. Text is separated in 15 character increments, which includes spacing and special characters. If an external text source were from, say a news article, the article would be translated into uppercase text, the length checked, then an iteration loop is initiated that would parse the news article into 15 character "chunks". The hidden text is comparatively checked against the modified external text source for the characters outlined, skipping over spacing and special characters.

Once the text is confirmed as a match, the location is tagged and marked. Tagging and marking assumes English writing principles of top-to-down, left-to-right reading patterns. For languages/cultures that utilize methods of right-to-left and/or bottom-to-top reading patterns, this proof-of-concept will not function properly. Marking each character is assuming <*x,y*> coordinates, with <0,0> being the top-most, left-most section of the external text source. To accommodate for word wrapping, the chunking process takes into account linefeed and carriage-return characters, and removes then, thus enabling the ability to stream the entire text message into a contiguous text stream. Once the text has been tagged and marked, the location is connected to a temporary hexadecimal string. Once the entire hidden text has been marked, the decimal string is bit shifted. In the external text source, supposed that the letters "C", "R", "Y", "P", "T", and 'O" were contained within the external's text source as "Can we read what you plan on seeing today?" The text would be confirmed within the external text source. The external text source would consist of an array of no more than 15 character length records, with approximately 3 records: [Record #1] "Can we read wha"; [Record #2] "t you plan on s"; [Record #3] "eeing today?" – and so on and so forth. For the rules set forth, esp. with the arbitrary figure of 40 times the number of characters within hidden text string would not work (with these rules). This is merely demonstrative of how this might function. Once the hidden text has been tagged and marked, and the locations within the external text source have been identified, the ending of the string is set to "0000" as there beginning text started at X-pos "01", Y-pos "01" – there is no X-pos "00" or Y-pos "00". We could utilize something else, but figured that this would make things easier to understand.

The "**Message-ID:**" string might be represented as something similar to this: "**Message-ID: <01010101010802030207020c0000@domain.com>**". The "**Message-ID:**" along with the external text source, is sent as an electronic mail message to the sending mail server, which is utilizing a anonymous remailer utility. Within the body of the email text, the message would contain the target destination (defined as: **Anon-To: target-user@target-domain.com**) the encryption identifier (defined as: **Steak-To: Message-ID: 01010101010802030207020c000@domain.com**), the subject field, and finally the body of the message. The anonymous remailer utility would bundle the package as a legitimate email message with a crafted "**Message-ID:**" header within the email header. The target-user would whomever is the recipient of the hidden text message.

Once received, the target-user would utilize a decryption utility, which would strip out the "**Message-ID:**" from within the electronic mail message header, then reverse the process against the external text source as the message. The message is revealed. Again, proof-of-concept is flawed in that it is not "foolproof"; however, it does demonstrate capabilities that would otherwise not be known.

Since there may be mail servers that may strip the "**Message-ID:**" header within the email message header, this method might not work; however, there are several other methods of obfuscating similar techniques. One method is something similar by embedding a spoofed email session ID header (using the crafted header outlined previously):

```
Return-Path: <rsradvan@unixworks.com>
Received: from srvr1006 (srvr1006.unixworks.net [64.32.214.198])
  by srvr1006.unixworks.net (8.12.11/unixworks.net) with SMTP id i5UHr5J3031807
  for <rsradvan@unixworks.com>; Wed, 30 Jun 2004 12:53:05 -0500
```

NOTE: The modified section of the email header is shown in **BOLD** and **yellow**.

This identifier represents the session ID number when the message was transferred into or throughout each mail server (in this case, all mail servers, both the originating and target, MUST have a session identifier). I have seen session identifiers much longer than "**i5UHr5J3031807**", but am uncertain if there is a length specification located within the RFC. Remember: the message identifier is the email message (and its payload with headers) traveling from one server to another, and so on and so forth. The session identifier represents that mail message for that particular server. Both identifier strings are incremental.

Flaws in Utilizing the Message-ID: Header

In the real world, there are no such things as a perfect situation; thus, this concept is flawed in its very design and its implication is that it cannot be stopped. Anything, given sufficient beer and time, can be stopped, altered, modified and crafted to how we want things to be perceived, and how we perceive the rest of the world. What is being said is that in an extremely paranoid society today, some organizations may completely strip out everything but the sending mail address header, the target destination mail header, date and time, subject and the body of the message text. Some corporations, through their filtering methods, may not consider the "**Message-ID:**" email header as something relevant, but log it within their mail or logging servers for forensics purposes. As such, stripping the "**Message-ID:**" header could stop hidden text messaging from being exchanged.

There are alternative methods if this method is blocked:

- ❖ The mail server session identifier may be utilized as an alternative payload mechanism, though I am uncertain if there are any length limitations.
- ❖ Injected mail headers, such as "**X-Comment: <blah>**" may be inserted as first-line injection into the email message payload. This too, can be removed or stripped.
- ❖ Use hexadecimal representation of subject headers (a little more sophisticated, but still probable) to correlate the external text source with the hidden text source.
- ❖ Length variations of the words used to signify positioning within the external text source; that is, if a word is 3 characters long, and next word is 7 characters long, the position is X-pos "3", Y-pos "7" *<3,7>*. The limitation is an evenly spaced sentence or paragraph, but this too, may be manipulated in such a manner to check oddly spaced or length sentences or paragraphs.

Conclusion

Though it isn't foolproof, the concept of transporting hidden text or commands within a text string can have devastatingly huge impacts upon the end-user community. From a legal perspective, end-users would like to say that they own their computer workstations, and whatever software is located on their workstations is however they see fit; software development companies, however, do not see the legal aspects in that light. Since this opens up a rather messy issue in legal debates, I would rather not go any further, except that the risk of having a software development company strategically using specifically emplaced hidden routines or commands within their applications, poses a serious threat to canned, end-user software. There are no implications stipulating that all software development companies (such as Microsoft) are injecting such code into end-user environments. However, in an ever-shrinking and highly competitive world for market dominance, there may be a point where a company may become desperate enough to exercise control of their share of the software application market (specific to whatever application they provide and support).

The implication is that something as simple as an electronic mail message, may be utilized to transport hidden or obfuscated text or commands from one place to another, with little or no detection of their existence or use.