6-2016

Covert Channel in HTTP User-Agents

Susan Heilman

Jonathan Williams

Daryl Johnson

Recommended Citation

S. Heilman, J. Williams, and D. Johnson, "Covert Channel in HTTP User-Agents," 11th Annual Symposium on Information Assurance (ASIA'16), Albany, NY, 2016, pp. 68-73.

Covert Channel in HTTP User-Agents

Susan Heilman
B. Thomas Golisano College of
Computing and Information Sciences
Rochester Institute of Technology
Rochester, New York 14623
sih4146@rit.edu

Jonathan Williams
B. Thomas Golisano College of
Computing and Information Sciences
Rochester Institute of Technology
Rochester, New York 14623
jpw2408@rit.edu

Daryl Johnson
B. Thomas Golisano College of
Computing and Information Sciences
Rochester Institute of Technology
Rochester, New York 14623
dgjics@rit.edu

Abstract—**A subliminal covert channel establishes a nearly undetectable communication session within a pre-established data stream between two separate entities.This document explains how HTTP can be utilized to facilitate a covert channel over both local and wide area networks. The Hypertext Transfer Protocol (HTTP) accounts for a majority of the Internet's daily web traffic and is permitted within almost all network topologies. Therefore, HTTP is a prime medium for hiding messages and information communicated between separate parties. This paper illustrates a new approach to covertly encoding messages in the an HTTP message through use of the User-Agent and referer strings in the HTTP Request Header.**

Index Terms—**Covert, Covert Channel, HTTP, User-Agent, HTTP Request**

I. Introduction

Research in the field of covert communications and covert channels has grown steadily since the topic was first introduced in the 1970s. Four decades later the Internet has expanded to include several hundred networking and application-based protocols. This means that data and information can be hidden in almost any level of the Internet's hierarchical structure. One of the most commonly used Internet protocols is the Hypertext Transfer Protocol, abbreviated HTTP for short. This protocol is responsible for the delivery of web pages and web content to a particular user or device. With 3,585,000,000 users in just North America and over 8.7 billion devices connected to the Internet, thousands of HTTP requests and HTTP responses are generated every minute accounting for nearly half of all traffic on the Internet [1]. This opens up a myriad of possible places to store hidden data as a means of communication. Some individuals have already discovered different methods hiding messages using the HTTP protocol. The covert channel that is proposed below provides a new way of hiding multiple messages in the Hypertext Transfer Protocol by manipulating fields within the HTTP request header. In some cases modifying the HTTP request header can be problematic if certain network security or web server security mechanisms are in place. When properly deployed these mechanisms may possibly cause the HTTP request to not be recorded into a security access log. The proposed covert channel manipulates areas of the HTTPS request header that have been known to not be kept under surveillance or are not policed often. This characteristic of the covert channel, among many others, increases the likelihood that the covert message will successfully reach the intended recipient.

In the following text below you will be introduced to the definition of what a covert channel consists of and a summary of the Hypertext Transfer Protocol. Next, there is a demonstration of how the proposed covert channel hides a message within the HTTP request header with a detailed analysis of the channel's design and properties followed by data that illustrates a proof of concept of the covert channel on a local network and over the Internet. Finally, there is an analysis of the proposed covert channel to measure the channel's effectiveness as well as examine its disadvantages.

II. Covert Channels

A covert channel uses the structure of an existing medium to send and receive small parts of data through unused portions of the medium. A covert channel hides in the nooks and crannies of the formal structure of the operation and is virtually undetectable. In Lampson's famous paper [2] he coins the idea of covert channels by describing them as "those not intended for information transfer at all, such as the service program's effect on the system load". For our purposes, we rely on the dismissive behavior towards HTTP User-Agents and referrer fields to hide discrete messages in the User-Agent string.

A. Characteristics of Covert Channel

The characteristics of a covert channel help define and gauge the effectiveness of a given covert channel when it is being used. There are three main characteristics that establish the foundation for a covert channel which are stealthiness, followed by bandwidth and last timing. In the subsections below each individual characteristic will be further explained.

1) Stealthiness: If a covert channel is described as stealthy it means that the channel does not provide an obvious view of the conversation to outside parties. In essence, if the two parties of the covert communication session were to have their network traffic monitored no flags or warnings would be generated by the monitoring party. It is as if the communication is not taking place at all.

2) Bandwidth: Bandwidth is a very sensitive factor when developing a reliable and robust covert channel. Bandwidth refers to the ratio of total messages sent in a given unit of time. Typically with overt channels a higher bandwidth rate is a good characteristic of networking communication. However, for covert channels a higher bandwidth of the secret messages sent to and from two parties means that the covert channel is creating more traffic thus rendering it more detectable. That is to say, the more bandwidth used to send a secret message the more likely the communication will be seen by an outside party.

B. Types of Covert Channels

Below are five of the commonly known ways to implement a covert communication channel into a Internet networking or application-based protocol.

1) Steganographic: Steganographic channels involve hidden covert information within a overt information medium. Steganography on the Internet usually entails hiding a covert message or photo within another image file like the .jpg or .png file extensions. [3].

2) Subliminal: Subliminal Covert channels are similar to Steganographic channels in that it hides a message within a normal pre-existing channel. However, when compared the subliminal channel and the overt channel are virtually indistinguishable from each other when compared against each other. [4].

3) Behavioral: A behavior-based covert channel is defined as a communication channel achieved by modulating the internal states of the sender or receiver by purposely selecting certain inputs to the systems. [5]. [4]

4) Storage: Storage channels involve directly or indirectly writing to a storage location by one application and the directly or indirectly reading the bits of a storage location by another application. This is different from regular file reading and writing information because a storage channel will write information to the file in a way that communicates a secret message instead of reading the intended information that the file is storing [3].

5) Timing: Timing channels work by one party signaling the other party with a message via the modulation of system resources that cause a change in response time that is noticed by the second party. A clock value or time measurement would convey the value being sent. For example, the clock or time measurement would be altered by the first party so as to communicate with the second party [3].

III. Related Work

Applications exist to generate covert messages using steganography. Snow [6], for example, uses steganography as its means of covertly communicating messages from one party to another by relying on whitespace - the spaces and tabs that naturally appear in writing that are invisible to the human eye when displayed at the end of lines in text viewing applications. This way, the hidden message can be placed in regular ASCII without modifying the original message's appearance.

On the other hand, [7] hides messages via steganography by mapping known ASCII characters to unused ASCII symbols that are visibly indistinguishable to the human eye.

Moving to the HTTP protocol, a covert channel can exist in HTTPT by manipulating the Query_String field when a HTTP redirect is executed. Such examples have been demonstrated by [8] through HTTP redirects.

IV. HTTP: Hypertext Transfer Protocol

The Hypertext Transfer Protocol was first standardized for use in June of 1999. The revolutionary protocol provided a simple and scalable hypertext and media delivery solution. HTTP is a stateless protocol with an open ended set of methods and headers that indicate the purpose of each HTTP request [9]. HTTP request chains and response chains are sent between a User-Agent and the origin server. Participants in the HTTP request and response chains can be known to initiate multiple simultaneous communications. This same protocol is used in other application layer information delivery mechanisms such as the FTP and SMTP Internet protocols.

The combination of widespread HTTP 1.1 use and the HTTP 1.1 design qualifies the Hypertext Transfer Protocol to be the best candidate for a robust, reliable covert channel. Because of the HTTP standard communicating properties a large number or requests or replies may be generated to facilitate a minimal request. This covert channel uses this HTTP communication property to its advantage in order to send multiple characters of a message to a HTTP origin server where it will be received and processed into a human readable format.

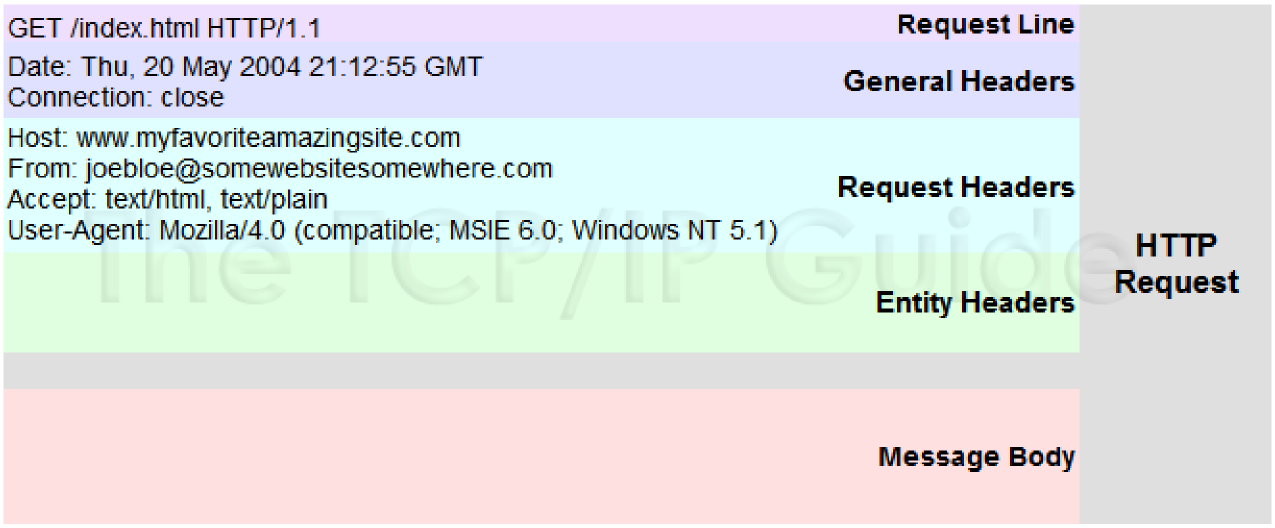
A. The HTTP Message Format

HTTP messages exist in two different types; They are either requests or responses. In a typical environment, the client is the one sending requests and receiving responses while the server receives requests and sends responses. It is not likely for a client to send HTTP responses, which is why the proposed covert channel avoids this interaction. The format for an HTTP message (without specifying request or response) follows this structure:

```
<start-line>
<message-headers>
<empty-line>
[<message-body>]
[<message-trailers>]
```

To get a better understanding of where the proposed message location fits into the HTTP message, a breakdown of the HTTP request message is needed. Figure 1 visually explains where the User-Agent string fits into the request.

The HTTP request "Host" variable shows the URL that is being requested. The "User-Agent" field is filled by the browser that is being used to send the request. The above User-Agent was filled by a Mozilla Firefox browser. This is where the message is hidden. An additional field to note in order to better understand the covert channel is the optional "referer" field that fits into the Request Header. This field identifies the original address of the webpage from which the current resource was linked. The referer field is typically populated



```
GET /index.html HTTP/1.1
Date: Thu, 20 May 2004 21:12:55 GMT
Connection: close
Host: www.myfavoriteamazingsite.com
From: joebloe@somewebsitesomewhere.com
Accept: text/html, text/plain
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
```

Fig. 1: HTTP Request

when the user clicks on a link on a webpage. The webpage that had the link is referring the user to the new link, so the address of the first page is put in the referer field [10].

For the purpose of the this covert channel, the HTTP response is not a concern. The HTTP request solely exists to confirm that the HTTP request sent in the channel was successful in reaching the server.

V. A Covert Channel in the User-Agent String

The proposed covert channel works by altering the User-Agent string that is present in the HTTP request header. In order for the message to be covertly communicated through this string, the message must undergo changes and mapping before being relayed to the server.

A. *Encoding*

The ASCII message entered by the user is converted to hexadecimal which is then hidden by modulating the User-Agent string. The chosen User-Agent string is a commonly seen User-Agent of "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.71 Safari/537.36." In this string there are several white spaces that are utilized to hide the message. In order to send the “0” hexadecimal value to the receiving server, the first white space that appears in the chosen User-Agent string is doubled. That is to say, instead of there being one white space, “ “, there becomes two, “ “. This is very difficult for the naked eye to notice. To convey a “1”, the second available white space is doubled, and so on. Figure 2 outlines which hexadecimal value corresponds to the extra space or spaces. The highlighted values that contain two hyphens are the areas where the extra spaces can be found.

Common User-Agent validation tools use regular expressions to parse the User-Agent field to detect bots and other malicious requests. If User-Agent validation tools were used parse our selected User-Agent string it would most likely focus on the "Chrome/46.0.2490.71" substring. To avoid running the risk of throwing an error, we avoided manipulating this specific substring of the User-Agent field.

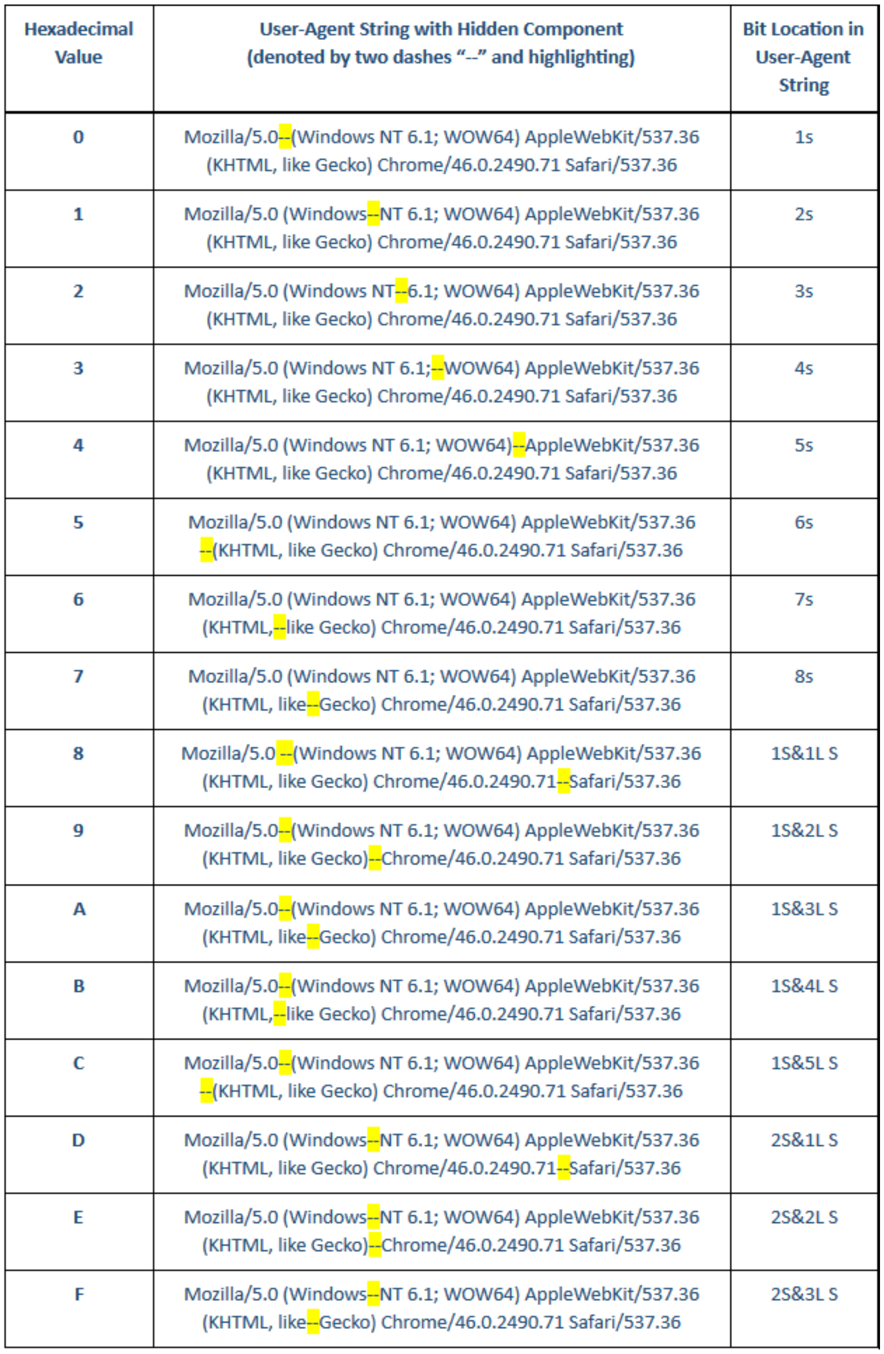
B. *Design*

The implementation of the proposed covert channel relies on the following components:

- Server
  - Acts as message receiver
- Client
  - Acts as message sender
- Linux "curl" command
- Linux "xxd" command
- Defined table mapped to hexadecimal characters (See Figure 2)

The channel works in the following manner: A server running Apache with port 80 open awaits a message. The client machine or sender runs the client.sh script that takes input from the user. The user input is then converted from an ASCII message to a hexadecimal string, and depending on the hexadecimal character a different User-Agent string is crafted as outlined in Figure 2. Once the User-Agent string is crafted with the necessary added spaces, the script executes a curl command to send an HTTP request to the server. The server response to the curl request is irrelevant to the channel for two reasons. Firstly, the client.sh script is setup to only send messages to a specified IP address and will discard the response. Secondly, even if the receiving server processing the curl request and does not provide a usable resource to the user encoded message characters are still logged and stored in the access.log file.

The “-A” and “-e” flags for curl are used to specify the User-Agent string and the referer field, respectively. The purpose of



Hexadecimal Value	User-Agent String with Hidden Component (denoted by two dashes "--" and highlighting)	Bit Location in User-Agent String
0	Mozilla/5.0--(Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.71 Safari/537.36	1s
1	Mozilla/5.0 (Windows--NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.71 Safari/537.36	2s
2	Mozilla/5.0 (Windows NT--6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.71 Safari/537.36	3s
3	Mozilla/5.0 (Windows NT 6.1;--WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.71 Safari/537.36	4s
4	Mozilla/5.0 (Windows NT 6.1; WOW64)--AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.71 Safari/537.36	5s
5	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 --(KHTML, like Gecko) Chrome/46.0.2490.71 Safari/537.36	6s
6	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML,--like Gecko) Chrome/46.0.2490.71 Safari/537.36	7s
7	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like--Gecko) Chrome/46.0.2490.71 Safari/537.36	8s
8	Mozilla/5.0 --(Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.71--Safari/537.36	1S&1L S
9	Mozilla/5.0--(Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)--Chrome/46.0.2490.71 Safari/537.36	1S&2L S
A	Mozilla/5.0--(Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like--Gecko) Chrome/46.0.2490.71 Safari/537.36	1S&3L S
B	Mozilla/5.0--(Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML,--like Gecko) Chrome/46.0.2490.71 Safari/537.36	1S&4L S
C	Mozilla/5.0--(Windows NT 6.1; WOW64) AppleWebKit/537.36 --(KHTML, like Gecko) Chrome/46.0.2490.71 Safari/537.36	1S&5L S
D	Mozilla/5.0 (Windows--NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.71--Safari/537.36	2S&1L S
E	Mozilla/5.0 (Windows--NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)--Chrome/46.0.2490.71 Safari/537.36	2S&2L S
F	Mozilla/5.0 (Windows--NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like--Gecko) Chrome/46.0.2490.71 Safari/537.36	2S&3L S

Fig. 2: User-Agent to Character Mapping Table. Note, the denotation of yellow in this table is to provide visual aid to a notation of a double dash marking, and does not reflect any meaning to the content provided.

the referer field for the proposed covert channel is to signify when a message is at the beginning, middle, or end stage. In other words, the referer field of the HTTP request header is a delimiter that is used by the receiver to distinguish between the start and stop of messages.

The client will deliver one HTTP request for each character sent. Once the client has completed sending the server has received and processed numerous requests with seemingly normal parameters that are stored in Apache access log. For Apache, the default path is "/var/log/apache2/access.log". Once the client has sent the message through HTTP requests with the curl command, the server.sh script need only pull the message from the access log. The script that is run on the server reads in the access.log file and performs a series of checks to search for the double spaces and reassembles the message in hexadecimal. Finally, the message is then converted from hexadecimal to ASCII.

This concludes the sending and receiving of a message over

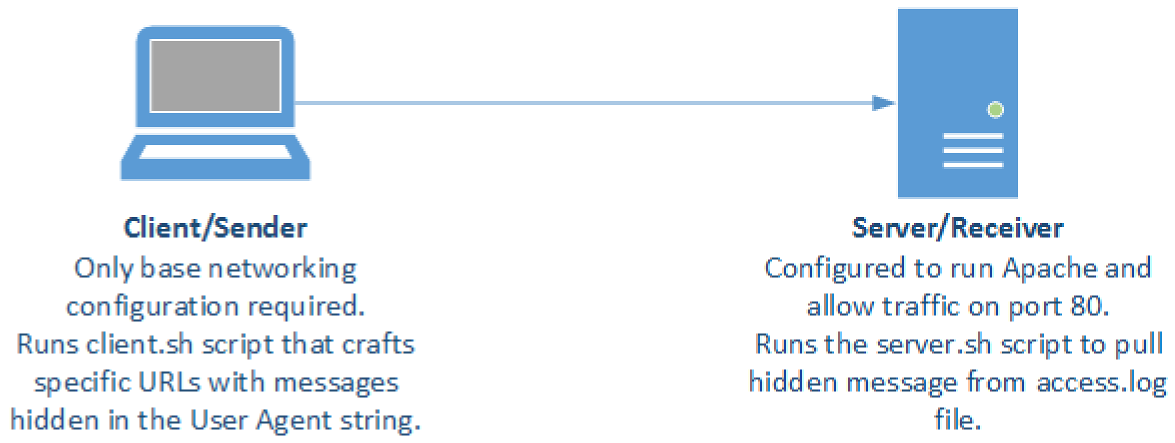

Client/Sender
Only base networking configuration required.
Runs client.sh script that crafts specific URLs with messages hidden in the User Agent string.
Server/Receiver
Configured to run Apache and allow traffic on port 80.
Runs the server.sh script to pull hidden message from access.log file.

Fig. 3: Client/Server Environment

a network given the proposed covert channel.

C. Testing

In order to test the proposed covert channel implementation, a virtual machine running a Linux OS was created. Apache was installed and configured on this server to ensure that the client/sender would have some webpage to which the the curl commands could send messages. The sender script was run from a Bash terminal window which prompted the user for a desired message that the script would then convert to hexadecimal prior to sending. Once the sender script finished its execution, the receiver script was run against the apache2 access.log file which converted the delivered message back to ASCII before printing it to Standard Output.

The rate at which data can be sent through the channel was tested by determining how many HTTP requests the server could process and log from the same client. In order to ensure that all requests would be logged on the server side, which also ensures that the entirety of the message is logged, sleep timers were used to guarantee that the requests were sent consecutively and in the correct order.

Further testing was performed by sending the proposed covert channel through a commonly used proxy service. By modifying the curl command to include proxy support, the sender.sh script forwarded the encoded covert traffic to a squid3 proxy server that used the ’combined’ logging setting. When the message was sent from the client each hexadecimal encoded character was received in order in the ’/var/log/squid3/access.log’. To read the message in this case the receiver script was modified to parse the squid access.log rather than the ’/var/log/apache2/access.log’ file. Considering that the proxy service stores the User-Agent and referer fields by default in its access.log, our covert channel can effectively deliver its message to a HTTP proxy log file as well.

VI. Qualities and Properties of the Channel

A. Categorization

Covert channels exist in various classifications and implementations. The proposed covert channel is best described as a subliminal covert channel. A subliminal channel hides its secret message in a normal channel’s medium. The subliminal channel and regular channel are indistinguishable without prior knowledge of the secret. The proposed channel’s secret is the hexadecimal conversion and its mapping to different User-Agent strings.

B. Stealthiness

The scripts written for the client and server are written in the command language, Bash. The Bash programming language was selected because nearly all Linux distributions are capable, at the very least, of running a bash shell. Furthermore, all commands and string manipulation used within the scripts is done solely in bash without relying on any interpretation other than bash. This is done through regular expressions and bash commands such as ’curl’, ’tr’ (trim) and ’xxd’ (to convert to and from hexadecimal).

C. Data Rate

As mentioned in the Testing section, a long message means many ’curl’ command executions to send many HTTP requests to the server. The proposed channel uses a one second delay to avoid loss of data transmission between the sender and receiver. With this in mind, the two parties communicating are not severely limited by the length of the message to be sent.

D. Robustness

The proposed covert channel is transmitted through one of the most commonly seen protocols in the wild: HTTP. In addition to its popularity, HTTP is rarely monitored or policed for discrepancies such as a referer field manipulation. It is also routable over wide area networks which allows the sender and receiver to be geographically separated.

E. Potential Issues

The scenario for which the covert channel must be successful is commonly available but relies on certain key aspects.

- There must be an existing and running web service with an open port, namely, Apache since the proposed channel searches in the folder path for Apache's log file.
- If the server has disabled logging then there is no access log for the message to be sent.
- The rate at which HTTP requests can hit the server is also an area of caution for the covert channel. If the Apache server sees too many requests from on particular client within a given time value, the server may ignore the requests.

F. Benefits

Thought simplistic in nature, the covert channel contains design specifications that enhance its usability and stealthiness. HTTP traffic is common and unlikely to be flagged by a firewall, especially since the covert channel uses legitimate User-Agent strings. By using whitespace to code the message, the communication is obfuscated in the log fie and nearly undetectable to the human eye. The script was intentionally written in Bash due to the language's popularity across Linux distributions (and now in Microsoft's newest Windows 10 update). The script also contains the minimal number of commands needed to avoid detection in respect to system resources. Overall, the decisions made to use Bash and limited commands was done to avoid dependencies across different operating systems. Finally, the issue of HTTP encryption (SSL/TLS) is moot since the receiver script is run server side.

VII. Future Work

Some future improvements to the proposed covert channel include dynamic User-Agent white space manipulation and a feature where HTTP requests can be tailored to the web page being requested. Rather than having to manually add the extra white space characters into the User-Agent string for each character in the hexadecimal encoding, the code would incorporate a looping feature that would automatically detect spaces in the User-Agent string and add additional spaces where necessary. This may be more complicated to implement, however it will allow the proposed covert channel to be used with other future User-Agent strings if a particular User-Agent becomes flagged as malicious or unauthorized. The other feature that will be implemented into the proposed covert channel involves using the URLs that are returned from the first 'curl' command to legitimized the web traffic that is being sent to the receiving server. After the first 'curl' command returns the web-page's HTML text it will parse through the returned HTML text line by line searching for any 'HREF' links that are on the web-page. Once all of the 'HREF' links have been stored into an array the following hexadecimal characters that are transmitted to the receiver will cycle through and use one of the array entries of web-page links to help legitimized the web traffic, thus making the covert channel harder to detect.

VIII. Conclusion

This paper introduced an implementation of a covert channel that utilizes the User-Agent string in the HTTP Request Header. In order to fully detail the mechanism of this covert channel, the paper first discuss the various types of covert channels, the characteristics a covert channel possess, and the the basis of the HTTP Message Format. This covert channel mimics stealthy behavior by relying minimally on system resources and calling up the most base Linux shell and command language. Furthermore, the transmission of the message is carried through a widely and abundantly used protocol: HTTP. The use of HTTP not only allows this channel to hide within network traffic, but it stretches across wide area networks to allow the two parties in the channel to be virtually anywhere in the world. As more and more business is conducted online, the availability of this HTTP User-Agent channel has the potential to become more widespread, accessible, and available to users capable of running the Bash shell.

References

[1] B. M., "Ellacoya data shows web traffic overtakes peer-to-peer (p2p) as largest percentage of bandwidth on the network," 2007.

[2] B. W. Lampson, "A note on the confinement problem," *Communications of the ACM*, vol. 16, no. 10, pp. 613–615, 1973.

[3] S. E. J. W. E. Pennington, W. Oblitey, "An overview of covert channels."

[4] S. E. J. W. R. Trimble, W. Oblitey, "Covert channels: The hidden threat."

[5] D. Johnson, B. Yuan, P. Lutz *et al.*, "Behavior-based covert channel in cyberspace," 2009.

[6] The snow home page.

[7] S. Bhattacharyya, P. Indu, and G. Sanyal, "Hiding data in text using ascii mapping technology (amt)," *International Journal of Computer Applications*, vol. 70, no. 18, 2013.

[8] E. Brown, B. Yuan, D. Johnson, and P. Lutz, "Covert channels in the http network protocol: Channel characterization and detecting man-in-the-middle attacks," *The Proceedings of the 5th International Conference on Information Warfare and Security: The Air Force Institute of Technology, Wright-Patterson AFB, Ohio, USA, 8-9 April 2010*, p. 56, 2010.

[9] R. T. Fielding, T. Berners-Lee, and H. Frystyk. Hypertext transfer protocol – HTTP/1.0. [Online]. Available: https://tools.ietf.org/html/rfc1945

[10] C. M. Kozierok, *The TCP/IP Guide a Comprehensive, Illustrated Internet Protocols Reference*. No Starch, 2005.