2010

Covert channels in the HTTP network protocol: Channel characterization and detecting man-in-the-middle attacks

Daryl Johnson

Bo Yuan

Peter Lutz

Erik Brown

Covert Channels in the HTTP Network Protocol: Channel Characterization and Detecting Man-in-the-Middle Attacks

Erik Brown, Bo Yuan, Daryl Johnson, Peter Lutz
Rochester Institute of Technology, Rochester, NY, USA
erik.t.brown@gmail.com
bo.yuan@rit.edu
daryl.johnson@rit.edu
peter.lutz@rit.edu

Abstract: Network covert channels provide two entities the ability to communicate stealthily. Hypertext Transfer Protocol (HTTP), which accounts for approximately half of all traffic on the Internet (Burke, 2007), has become the fertile ground for various network covert channels. Proliferation of network covert channels throughout the World Wide Web and other areas of cyberspace has raised new security concerns and brought both challenges and enhancements to the area of Information Warfare. Covert channels impact our ability to observe and orient in this domain and need to be better understood. They are however, extremely difficult to study as a whole. Network covert channels tend to be protocol, implementation, and/or application specific. Similar to biology or botany, where we classify plants and animals, the first step of research is to define a classification scheme. In the paper, it is intended to define a set of common characteristics, classify and analyze several known covert channels in HTTP with respect to these characteristics. New HTTP based covert channels are discussed and their characteristics presented as well. Although many applications of covert channels are malicious in nature, this paper argues that there are beneficial applications of network covert channels, such as detecting Man-in-the-Middle attacks.

Keywords: Network Covert Channels, Data Hiding, HTTP, Network Security, Man-in-the-Middle

1. Introduction

Since Butler Lampson first coined the term, “covert channels,” (Lampson,1973) their definition and implementations have varied considerably. In a well known text on covert channels, commonly known as the “Light Pink Book”, Virgil Gligor defines covert channels as "a communication channel that allows a process to transfer information in a manner that violates the system's security policy" (Gligor,1993). Such definition most accurately depicts the essence of what designates a given communications channel as being *covert*. This definition is however, given in the context of an automated information system (AIS) responsible for multilevel security (MLS), whereby the goal is responsible access control between different subjects and objects of differing security classifications. Today, a “system” we are also concerned with is the highly interconnected Internet, where entities can be networked applications or nodes instead rather than processes within an AIS. Thus, for purposes of this discussion, a more general definition based on Gligor’s is adopted, whereby covert channels are those which are denied by policy but allowed by property (Gligor,1993). Under this definition, it is easier to expand the study of covert channels to include network-borne covert channels such as those over HTTP. One may refer to these covert channels as network covert channels. Throughout this paper, the terms covert channels and network covert channels are used interchangeably.

Network covert channels now pose even more significant security concerns and challenges, as well as enhancements to Information Warfare (IW). Not only must we be ever more vigilant to ensure that confidential information is not secretly leaked to unauthorized processes, but also that it is not able to traverse the exponentially growing number of links and interconnected nodes throughout and peripheral to the Internet. Another threat of covert channels is the introduction of new attack vectors to our information systems. Covert channels can be used to surreptitiously implant malicious code in our automated information systems. In the physical domain, more often than not, we have the luxury of perception being reality or at least the ability to interrogate and inspect things as they pass through security checkpoints. Covert channels over HTTP or computer networks, in general, often separate perception from reality and make interrogation and inspection in cyberspace very difficult. Observing and orientating in the cyberspace domain is already a challenging task, covert channels make developing situational awareness of the domain much more difficult since there may be much more taking place than what is perceived. Network covert channels are often protocol specific, implementation specific, and application specific. In this paper, it is intended to further understand such covert communications channels by introducing a set of common characteristics, and classify covert channels with respect to these characteristics. HTTP-based covert channels are used to illustrate the process.

The remainder of the paper is organized as follows. In Section 2, a set of characteristics are defined as a means for studying and comparing covert channels. Section 3 evaluates a range of covert channels previously identified in HTTP with respect to the characteristics identified in Section 2. Section 4 presents some new HTTP-based covert channels. Section 5 discusses beneficial applications and uses of covert channels in cyberspace, especially applications of covert channels to detecting and preventing the man in the middle attacks in HTTPS.

2. Covert Channel Characteristics

When evaluating covert channels, it is important to establish a common criterion or set of characteristics and metrics that aim to establish discrete and differential values to qualify, distinguish and assess various implementations of covert channels. The following defines some characteristics that will be used to compare the HTTP covert channels in Sections 3 and 4.

2.1 Mechanism

The mechanism of a covert channel specifies how the channel is constructed and what it does to “hide” and carry information. The hiding mechanism is what differentiates each covert channel from one another. Therefore, the mechanism often influences other characteristics such as types, detection, capacity, prevention, etc.

2.2 Type

One of the most common practices is to classify covert channels by type. While a few different taxonomies have been proposed, such as in (Wang, 2005), the most common practice is to classify by the following types: storage, timing and behavioral based channels (Johnson, et al, 2009).

2.3 Throughput

This metric, otherwise known as capacity, measures the amount of data the channel is able to transmit over a given period of time. The throughput of a covert channel is often in a reciprocal relationship with the “hiddenness” of the channel. The higher the throughput is, the easier the channel can be detected. Covert channels are often used to tunnel or encapsulate other protocols; as such, the throughput or “goodput” can vary considerably, independent of the channels capacity. Only the overall throughput, independent of tunneled protocols is examined for each channel in this paper.

2.4 Robustness

This characteristic addresses the survivability of a given channel. Often network covert channels will encounter store-and-forward devices, firewalls, proxy servers or other similar devices. Channel robustness describes the ability of the covert channel to persist in such circumstances. Only three scales *High, Medium*, and *Low* will be used to characterize a channel based on the best understanding by the authors.

2.5 Detection

The probability of detection is critical to measure when comparing covert channels. Depending on the particular channel, this could be particularly challenging to discretely measure. However for a large number of covert channels, values for probability of detection can be derived from the entropy exhibited by the covert channel. The three scales *High, Medium,* and, *Low* will be used.

2.6 Prevention.

Each channel should also be evaluated for its ability to be prevented or disrupted. Prevention of covert channels can take place at any point along the data path. Prevention is unique from robustness in that robustness is the ability of the covert channel to persist in naturally occurring circumstances. Prevention, on the other hand, is the ability to take explicit action(s), often user or administrator intervention, to disrupt or degrade the covert channel. The three scales *Hard, Moderate,* and, *Easy* will be used.

3. Analysis of Existing Covert Channels in HTTP

HTTP is an application-level protocol used for information transfer on the Internet. It has a defined specification, including format and set of operations it can perform. However, it is flexible enough to facilitate the exchange of data between a wide variety of users and applications on networked information systems. Generally, the protocol consists of request-response pairs. A more detailed description of the modern implementation of the protocol is given in the Internet Engineering Task Force Request for Comments Document 2616 (Fielding, et al., 2009). The protocol's flexibility to transfer various kinds of legitimate data has also lead to its exploited use including the covert transfer of information as evident by a number of HTTP covert channel implementations.

The first several covert channels listed here have been documented by Bauer (Bauer, 2003). Bauer introduces several covert channels in HTTP that covertly communicate information between web servers via unwitting HTTP clients.

3.1 Redirects

Mechanism. HTTP Redirects (RFC 1616 303 messages) are used to redirect clients to a new URL. When the new URL is a CGI script, parameters can be included in the QUERY_STRING field of the new URL (e.g. www.foo.com/page.php?data=bar). The parameters specified in the QUERY_STRING are the set of data that are covertly communicated from one web server to another via a browser client.

Type. Given that data is stored directly in the QUERY_STRING field, this constitutes a storage channel.

Throughput. Given the size limit imposed on a standard URL, the maximum capacity for this channel is 1024 bytes per Redirect.

Robustness. While most common browsers will support HTTP redirects, some browsers do not fully support these redirects and when a redirect is issued to such browsers, the channel would be broken. Other than browser constraints, this channel is very robust over typical network links including via HTTP proxies, giving a high robustness score.

Detection. When a redirect is sent to the browser, in most cases, the browser will recognize it and redirect to the new URL. The new URL is then visible in the address bar of the client's browser. An observant user may notice the presence of additional parameters in the QUERY_STRING. Also, any network monitoring device would clearly see the data specified in the QUERY_STRING. It is quite common for Intrusion Detection Systems (IDS) to scrutinize the QUERY_STRING. The probability of detection is high.

Prevention. This type of covert channel is easily preventable by writing firewall or IPS rules to block HTTP Redirects (RFC 1616 303 messages).

3.2 Cookies

Mechanism. Client-side cookies can store key-value pairs that can be accessed not only by the creating server, but also by any server in the domain, defined within the cookie. Dynamic DNS services like dyndns.org allow arbitrary servers to be placed in the same cookie domain, thereby enabling the covert exchange of cookie information when the client connects to two different servers in .dyndns.org (e.g. foo.dyndns.org and bar.dyndns.org)

Type. Because the data is stored in a cookie which is transferred from the server to the client back to a server, this constitutes a storage channel.

Throughput. One cookie can store up to 4 kilobytes of data in the value field. Since a maximum of 40 cookies can be stored per server, 160 KB of data can be shared as a result of cookies set by one server.

Robustness. Cookies were designed to enhance the user experience by remembering certain data, not play a pivotal role in the functionally of client-server architecture. As such cookies can be deleted on the client at any time. Deletion of such shared cookies would effectively break this covert channel. However, the robustness of this channel is high if the cookies are not deleted.

Detection. While many users and system administrators are aware that cookies may be exchanged between their browser client and a web server, often they do not interrogate the contents of the cookie when everything is working properly. Standard IDSs also do not bother to check for a common cookie being shared between two web servers, and as such probability of detection is relatively low.

Prevention. Web browsers that do not allow web servers to *set-cookie* or browsers that have plug-ins that place cookie management at the discretion and control of the user could be used to prevent the use of cookies as a means for a covert channel. Hence, the prevention is easy.

3.3. Referer

Mechanism. In HTTP requests (i.e. GET or POST), the optional Referer tag in the HTTP header can be manipulated to embed arbitrary data instead of the true referring page.

Type. The data is stored in the value portion of the Referer header attribute and is therefore a storage channel.

Throughput. As an attribute value, the length of the Referer data is limited to the length of a standard URL, which is 1024 bytes.

Robustness. This data stored in the Referer tag would generally persist across standard proxy servers. Few network devices would interfere with the Referer header field, given its widespread use, making this channel highly robust.

Detection. Arbitrary data in the Referer header, while not in the purview of a user, or standard firewalls/IDS's would however, raise suspicion on a deep packet inspection device such as advanced IDSs, Application layer firewalls or Proxy servers. Probability of detection is high.

Prevention. An HTTP proxy, deep packet IDS, or application layer firewall could be configured to sanitize any non-standard URL data embedded in the Referer tag, making prevention easy.

3.4 HTML Element Tags

Mechanism. In this technique, the HTML elements sent to the browser from one web server can be used to specify a new request to another web server and thereby pass information in such request. The common example is the use of the <META HTTPEQUIV> tag/attribute which can embed HTTP protocol headers in the body of an HTTP message (Bauer, 2003). This could also be used to embed additional set-cookie headers.

Type. Although this covert channel moves the hidden data from the header to the message portion of the HTTP packet, it still constitutes a storage channel.

Throughput. This channel has increased throughput over the others discussed previously. This channel seems to only be limited by the amount of such embedded HTML elements one can put in legitimate messages, for which there is no set limit, per the specification.

Robustness. This covert channel is highly robust since few, or perhaps no browsers or network devices are likely to remove HTML elements in the HTTP message content.

Detection. This is a particularly stealthy covert channel because while some deep packet inspection devices (e.g. Application layer firewalls, Proxies, etc) may scrutinized HTTP header information, most neglect the data in the body of the HTTP message. However, if a user decides to view HTML source code, they would see the hidden HTML elements, making probability of detection high.

Prevention. Preventing the use of this covert channel would be particularly challenging as one would have to manually parse the HTTP message contents for these hidden HTML elements being used for covert purposes. While this could eventually be automated, contemporary technology is not currently performing this sort of message scrutiny.

3.5 Active Content

Mechanism. This class of covert channels is high in the protocol stack, towards higher-level application layer covert channels, but because they are almost always implemented over HTTP, it is included in this discussion. The idea is to use code executed on the client (e.g. Java, JavaScript, Flash, ActiveX) to open communication channels between scripts running on different web servers and clients. The scripts could then use any number of mechanisms to covertly leak data, such as opening additional TCP ports, requesting webpages or submitting arbitrary HTML forms.

Type. Active Content can be used to implement different types of covert channels. Some JavaScript may issue redirects or construct invisible HTML forms to send data to another server. Other implementations could be construed to be timing or behavioral based. However, most implementations known to the authors are storage based.

Throughput. Depending on the exact implementation of the Active Content covert channel, the throughput can vary considerably. For example, a JavaScript which creates and populates a hidden HTML FORM and then submits it to a server may have a very large payload, only limited by HTTP message content limits. On the other hand, an ActiveX script implementing a timing channel may have significantly less throughput.

Robustness. Browser plug-ins or Anti-Virus software which prevents Active Content from executing could easily break this class of covert channels in HTTP. However, if scripts are allowed, the channel will be highly robust.

Detection. It is easy to detect any script activities. However, it is very difficult to tell if a running script is a covert channel. Anti-Virus software, browser plug-ins like NoScript, and host-based intrusion detection software may eventually be able to analyze Active Content from web servers for the possible presence of covert channel evidence.

Prevention. The only way to reliably prevent this class of HTTP covert channels would be to prevent the execution of any Active Content received from a web server unless the code could be proved to be free of covert channels. However, this type of code analysis remains a problem yet to be solved. One very easy prevention technique is to stop running any scripts or active content.

3.6 Arbitrary Headers

Mechanism. In the HTTP/1.1 specification (Fielding, et al., 2009), it is clear that headers must follow a defined format, that being ‘field name: field value.’ While standard field names such as ‘user-agent’ and ‘Referer’ exist, the protocol allows the users to develop their own header fields as well. This technique, described by Smeets and Koot (Smeets and Koot, 2006), allows the user to use arbitrary headers in HTTP requests and responses to convey data, for example, Foobar: secret. This channel could be further expanded to be synchronous if one were to define a custom Entity-Body in HTTP responses to these special requests.

Type. Given that this channel stores information in header field names and values, this constitutes a storage channel.

Throughput. There are many variables making it difficult to determine an exact number for the throughput of this channel, as such the throughput is variable. This is due to the fact that there is no fixed length on the size of HTTP headers for requests or responses and even the maximum length supported varies by HTTP server vendors. The throughput of the channel will depend on the number of header field-value pairs that are covertly inserted. There are some mandatory HTTP header fields to account for as well, which could impact throughput. Beyond that, choosing a ratio of legitimate data to covert data will affect not only throughput but also relative stealth.

Robustness. It is a highly robust channel due to the inherent robustness of HTTP protocol. However, networks with a higher security posture could conceivably have deep packet inspection devices, which upon detecting the presence of nonstandard header fields, might scrub such data before forwarding it on to the destination, effectively breaking this channel.

Detection. The presence of non-standard header field names is not that uncommon, especially given the myriad of custom HTTP applications using customized headers in use today. However, since HTTP headers are transmitted in plaintext, for scrutinizing users or software, there is a high probability of detection.

Prevention. To prevent this channel, one could establish a policy effectively white-listing permitted header field names. Through the enforcement of a HTTP-aware proxy, packets containing header fields other than those on the whitelist could be discarded or scrubbed to protocol conformance with white-listed fields only. This prevention measure assumes that one does not accidently white-list a header field which could later be used for covert means. Prevention of this channel is rated moderate.

3.7 Unexpected Entity-Body

Mechanism. The Entity-Body is almost always included in HTTP responses, but with HTTP requests, it is only included in the POST method. This covert channel hides data in an unexpected Entity-Body associated with a request that normally does not have an associated Entity-Body such as a GET request (Smeets and Koot, 2006).

Type. Since the data is stored directly in the unexpected Entity-Body of the request, this is a storage channel.

Throughput. The capacity of this channel varies in that it could potentially hold up to the amount of payload space available to HTTP minus the length of the HTTP header, which is variable in length. Supposing a maximum IP packet size of 65,535 bytes and an MTU of 1500 bytes (Ethernet), the IP packet would have to be distributed over 44 Ethernet frames. Each frame would have 20 bytes of IP header data and 20 bytes of TCP header data, resulting in 63,775 bytes available to each HTTP request or response.

Robustness. This channel uses standard style HTTP transactions. As such, it is a highly robust channel since its robustness is dictated by that of the HTTP protocol itself.

Detection. Because there are legitimate instances where HTTP requests have an Entity-Body section, such as in the case of POST requests, it is unlikely that end nodes or intermediate network devices will detect or strip the request of the unexpected Entity-Body section.

Prevention. Devices capable of detecting protocol conformance or statistical behavior analysis would notice exceptionally large GET requests or detect the unexpected Entity-Body section. If configured to do so, devices such as a web proxy could strip out the extra Entity-Body section as a way of preventing these types of covert channels. The difficulty of prevention is moderate.

4. New Covert Channels in HTTP

4.1 HTTP Timing channel

This channel was originally developed by the first author in July 2007 (Brown 2007) and has evolved since. It may be the only timing-based covert channel in HTTP. However, it may also be considered as a special case of TCP/IP timing channel reported in recent discovered reference (Cabuk, et al, 2004). The methodology of this channel could easily be used to implement covert channels on other protocols as demonstrated in (Cabuk, et al, 2004).

Mechanism. Given an established time window, known as the SLEEP_TIME, which can be negotiated upon channel initialization or determined priorly, the deterministic behavior of whether or not a client makes a GET request is interpreted by the server as either a binary 1 or 0. In this model, information is covertly relayed unidirectional; from the client to the server. Channel operation is depicted in Figure 1.

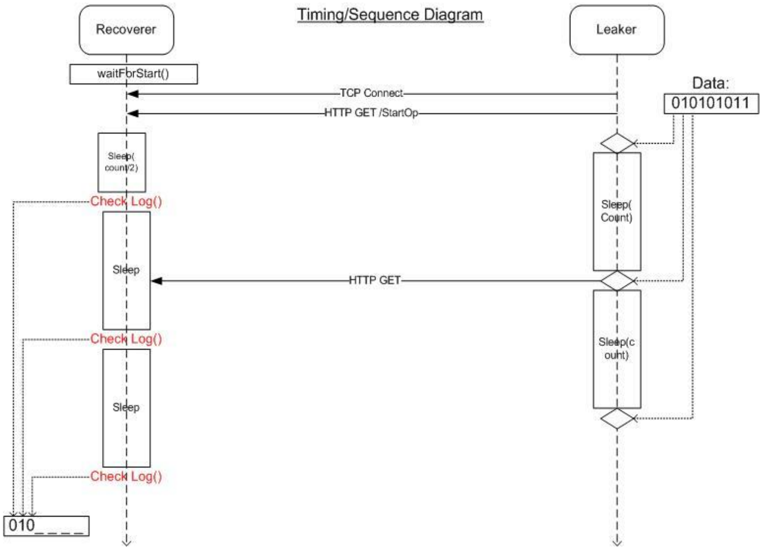

Figure 1: HTTP Timing Channel

Type. Given the channels dependency on relative timing, this is a timing channel.

Throughput. Throughput is inversely proportional to SLEEP_TIME. However, the authors' preliminary experimental tests have shown the highest attainable throughput, without ever encountering transmission errors, to be 1.82bps. These tests were conducted on a local area network (LAN) where both the machine running the "Leaker" program and the machine running the "Recoverer" program were both on the same LAN segment, only separated by a switch.

Robustness. The channel incorporates some accommodation for transmission latency and the threshold for such latency is directly proportional to SLEEP_TIME. Latency exceeding such threshold would result in errors, thereby breaking the channel.

Detection. Since the channel uses normal HTTP requests, it is very difficult to notice any anomalies from a vast amount of other HTTP requests on the Internet.

Prevention. A proxy-like device which caches HTTP GET requests en-route and regulates when they are released could be used to prevent this type of channel by disrupting the timing.

4.2 Web Page Browsing Pattern

A client can browse webpages in certain patterns to convey information to the webserver, forming a covert channel. This channel exists in theory and has yet to be implemented. It, like the timing channel, would be unidirectional, covertly relaying information from the client to the server. An appealing aspect of this channel is that no special client software is required. There are three mechanisms for such a covert channel, which are described below.

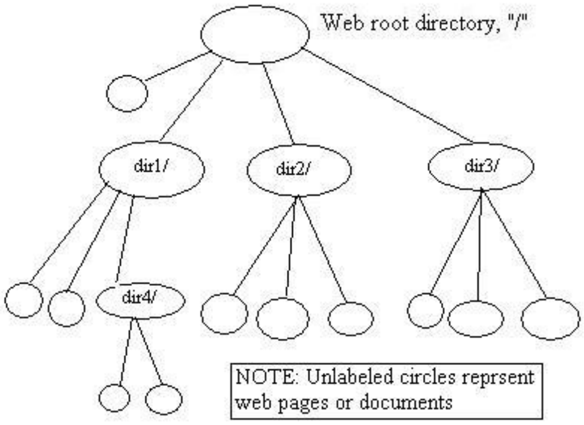

Figure 2: Web Directory Traversal

Mechanism 1. Given a website, a tree or graph structure can be created, with the web root directory, as the root node and pages or files within a hierarchical directory structure constituting the other nodes in the graph (see Figure 2). Whether the client requests a document that is closer to or further away from the root node, relative to the previously requested document in the structure, this can be translated to a binary 1 or 0. This constitutes one of several encoding mechanisms for this particular channel.

Mechanism 2. This mechanism would involve designating values to nodes or pages in the graph. By requesting a certain page, the client effectively communicates the value associated with that node. In a very simple sense, pages can be designated as either a 0 or 1. Since there are often more than two pages on a given website, values could be more intelligently assigned to nodes to increase the channels capacity by relaying more than one bit of information in a given transmission. To effectively communicate *B* number of bits, in a given request, *P* number of pages/nodes would be needed in the graph structure where P=2^B, with each page representing a unique value.

Mechanism 3. This mechanism involves designating certain node(s) as goal node(s) similar to a goal state in a search tree. To eliminate ambiguity, goals nodes must be positioned deep and towards the center of the tree. If the documents requested by the client follow the pattern of depth-first search, that could mean binary 1, while following the pattern of breadth-first search to get to the goal state could mean binary 0. Of course, synchronization would be required to instantiate the channel. This could be achieved by requesting the designated root or “start” page.

Type. These types of covert channels are behavioral based channels because they function based on the behavior of the documents requested by the clients.

Throughput. Depending on the specific mechanism used and how it is implemented, throughput can vary. The first mechanism is able to relay one bit of information per GET request while the second can vary according to how values are assigned to nodes in the tree. In the case of the third mechanism, the search algorithm, the throughput is much lower since multiple GET requests are required to relay just one bit of data.

Robustness. There is nothing about this channel which diverges from the accepted use of HTTP. Additionally the behaviors described here are persistent across proxy servers and application level firewalls, making it very robust.

Detection. It is extremely difficult to detect such covert channels. The only anomaly is that the client is constantly visiting and/or re-visiting certain web pages.

Prevention. Without knowing the endpoint of this communications channel, there are no known prevention measures to the authors’ knowledge.

5. Applications of HTTP Covert Channels – Detecting Man-in-the-Middle Attacks

In some cases, covert channels can enhance overall network security. Yuan and Lutz demonstrated a covert channel in a modified TFTP protocol that is able to download a file and its checksum simultaneously using a covert channel, which makes file altering during transmission detectable (Yuan and Lutz, 2005).

Covert channels can be used to detect man-in-the-middle (MITM) attacks. Such attacks are most common in non-certificate (PKI) based secured connections such as HTTPS using SSL version 1.0 and the Diffie-Hellman key exchange. Often the attacker is positioned between the two end nodes and performs a Diffie-Hellmen key exchange with both parties, resulting in the use of two session keys, one between the client and the attacker and the other between the attacker and the HTTPS server. The challenge here is how to let the client know the session key being used by the client is different from the one being used by the server. One technique is to transmit a checksum or a hash of the session key via a covert channel which is unknown to the MITM. Almost all different types of HTTP covert channels discussed in previous sections can be employed. Behavioral-based HTTP covert channels are often more difficult for the eavesdropper to detect.

Another way to use covert channels to detect or prevent MITM attacks could be using the covert channel to transmit arbitrary data, indicating the absence of a MITM. Consider, for example, the HTTP timing channel. A MITM would inherently disrupt the timing of requests, thereby breaking the covert channel. If clients using the timing channel were to expect some arbitrary message, but do not receive it due to the MITM's effect on the timing of packets, the client would know that there exists a MITM, and could cease further communication with the server. This methodology can be expanded to other types of covert channels, in which the presence of a MITM is known by the disruption of the covert channel.

Both of these applications need not be limited to detecting HTTPS MITM attacks, but can be applied to other protocols as well such as Internet Key Exchange (IKE), Secure Shell (SSH) and any other protocols using Diffie-Hellmen or otherwise, vulnerable to MITM.

4. Conclusion

Due it is ubiquitous nature, HTTP will continue to be targeted and analyzed for covert communication channels. As evident from the channels outlined in this paper, various types of covert channels exist in HTTP, including storage, timing and behavioral based channels. Covert channels can be characterized by their different mechanisms, channel types, capacities, robustness and their ability to be detected or prevented. There are potential beneficial applications of covert channels such as exchanging message integrity information and detecting or preventing Man-in-the-Middle attacks, as highlighted in this paper.

References

Burke, M. (2007) "Ellacoya data shows web traffic overtakes peer-to-peer (P2P) as largest percentage of bandwidth on the network", BusinessWire. [Online]. Available: http://www.businesswire.com/news/google/20070618005912/en

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

Gligor, Virgil. (1993) *A guide to understanding covert channel analysis of trusted systems*. Technical Report NCSC-TG-030, National Computer Security Center, Ft. George G. Meade, Maryland, U.S.A.

Wang, Z. (2005) "New constructive approach to covert channel modeling and channel capacity estimation," [Online]. Available: http://palms.ee.princeton.edu/PALMSopen/ISC05 w cit.pdf

Johnson, D., Lutz, P. and Yuan, B. (2009) "Behavior-based covert channel in Cyberspace," In: Vanhoof, K., et al (eds) *Intelligent Decision Making Systems*. World Scientific, New Jersey, pp. 311-318.

Fielding, R., et al. (2009) “Hypertext transfer protocol - HTTP/1.1., RFC 2616,” Internet Engineering Task Force, 1999, 2009. [Online]. Available: http://www.ietf.org/rfc/rfc2616.txt

Bauer, M. (2003) “New covert channels in http: Adding unwitting web browsers to anonymity sets,” in the ACM workshop on Privacy in the Electronic Society, pp. 72–78.

Smeets, M and Koot, M. (2006) “Research report: Covert channels,” University of Amsterdam, Tech. Rep. [Online]. Available: http://staff.science.uva.nl/~delaat/snb- 2005- 2006/p27/report.pdf

Brown, E. (2007) Covert operations on secure networks. Unpublished seminar paper. US Air Force Research Laboratory

Cabuk, S., Brodley, C. E. and Shields, C. (2004) “IP covert timing channels: Design and detection,” in Proceedings of the 11th ACM Conference on Computer and Communication Security, Washington DC, USA, pp. 178–187.

Yuan, B. and Lutz, P. (2005) “A covert channel in packet switching data networks,” in Proceedings of the Second Upstate New York Workshop on Communications and Networking, Rochester, New York, pp. 2–6.