Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 30, 2015

A. Doria
dotgay LLC
N. ten Oever
Article 19
J. Varon

October 27, 2014

Proposal for research on human rights protocol considerations
draft-doria-hrpc-proposal-00

Abstract

Work has been done on privacy issues that should be considered when creating an Internet protocol. This draft suggests that similar considerations may apply for other human rights such as freedom of expression or freedom of association. A proposal is made for initiating IRTF work researching the possible connections between human rights and Internet standards and protocols. The goal would be to create an informational RFC concerning human rights protocol considerations.

Discussion on this draft at: hrpc@article19.io

Table of Contents

1. Introduction

The recognition that human rights have a role in Internet policies is slowly becoming part of the general discourse. Several reports from former United Nations (UN) Special Rapporteur on the promotion and protection of the right to freedom of opinion and expression, Frank La Rue, have made such relation explicit, which lead to the approval of the landmark resolution "on the promotion, protection and enjoyment of human rights on the Internet" [HRC2012] at the UN Human Rights Council (HRC). And, more recently, to the resolution "The right to privacy in the digital age" [UNGA2013] at the UN General Assembly. The NETmundial outcome document [NETmundial] affirms that human rights, as reflected in the Universal Declaration of Human Rights [UDHR], should underpin Internet governance principles. Nevertheless, the direct relation between Internet Standards and human rights is still something to be explored and more clearly evidenced.

Concerns for freedom of expression and association were a strong part of the world-view of the community involved in developing the first Internet protocols. Apparently, by intention or by coincidence, the Internet was designed with freedom and openness of communications as core values. But as the scale and the industrialization of the Internet has grown greatly, the influence of such world-views started to compete with other values. The belief of the authors is that as the Internet continues to grow, the linkage of Internet protocols to human rights needs to become both structured and intentional.

Standards and protocols form the basis of the human rights enabling infrastructure of the Internet. It needs to be determined whether there is a causal relationship between Internet protocols and standards, and human rights such as freedom of expression. To study the relationship between the two one would need to carefully consider structural and architectural considerations, as well as specific protocols. The Internet Society paper "Human Rights and Internet Protocols" [HRIP] ’explores human rights and Internet protocols comparing the processes for their making and the principles by which they operate and concludes that there are some shared principles between the two.’ Though that paper does not go into possible reasons, dependencies or guidelines, it initiates the discussion. More research is needed to map human rights concerns to protocol elements and to frame possible approaches towards protocols that satisfy the implications of human rights standards.

To move this debate further, a list has been created for discussion of this draft: hrpc@article19.io and related ideas - information or subscriptions at: https://lists.ghserv.net/mailman/listinfo/hrpc

1.1. Requirements Language

As this is an informational document recommending a research effort, it will not make use of requirements language as defined in RFC 2119 [RFC2119].

2. Research topic

In a manner similar to the work done for RFC 6973 [RFC6973] on Privacy Consideration Guidelines, the premise of this research is that some standards and protocols can solidify, enable or threaten human rights, such as freedom of expression and the right to association and assembly online. To start addressing the issue, a mapping exercise analyzing Internet architecture and protocols features, vis-a-vis possible impact on human rights needs to be undertaken. The list below represents the first examples of this exercise.

2.1. Protocol and Standard Examples

Some initial topics that need exploration are indicated in this section. Most of this work has yet to move beyond speculation and casual conversation. The next release of the draft will develop these discussion further, based on discussion to be held on the hrpc@article19.io email list.

2.1.1. Architecture

RFC 1958 [RFC1958] mentions 'the community believes that the goal [of the Internet] is connectivity, the tool is the Internet Protocol'. It continues a bit further: 'The current exponential growth of the network seems to show that connectivity is its own reward, and is more valuable than any individual application such as mail or the World-Wide Web.' This marks the intrinsic value of connectivity which is facilitated by the Internet, both in its principle, and in practice. This shows that the underlying principles of the Internet aim to preserve connectivity, which is fundamental and similar to the part of article 19 of the Universal Declaration of Human Rights [UDHR], which defines a right to receive and to impart information.

2.1.2. Transparency

Another part of article 19 of the Universal Declaration of Human Rights UDHR [UDHR] mentions that one has the right to hold opinions _without interference_ (emphasis added). This same sentiment can be found in IAB RFC4924 [RFC4924] - Reflection on Internet Transparency where it states: 'A network that does not filter or transform the data that it carries may be said to be transparent" or "oblivious" to the content of packets. Networks that provide oblivious transport enable the deployment of new services without requiring changes to the core. It is this flexibility that is perhaps both the Internet's most essential characteristic as well as one of the most important contributors to its success.'

2.1.3. HTML

Websites made it extremely easy for individuals to publish their ideas, opinions and thoughts. Never before has the world seen an infrastructure that made it this easy to share your brainchild with such a large group of other people. The HTTP architecture and standards, including RFC 7230 [RFC7230], RFC 7231 [RFC7231], RFC 7232 [RFC7232], RFC 7234 [RFC7234], RFC 7235 [RFC7235], RFC 7236 [RFC7236], and RFC 7327 [RFC7237], are essential for the publishing of information. The HTTP protocol, therefore, forms an crucial instrument for freedom of expression, but also to the right to freely

participate in the culture life of the community, to enjoy the arts and to share in scientific advancement and its benefits.

2.1.4. Mailing lists

Collaboration and cooperation have been part of the Internet since its early beginning, one of the instruments of facilitating working together in groups are mailing lists (as described in RFC 2369 [RFC2919], RFC 2919 [RFC2919], and RFC 6783 [RFC6783]. Mailing lists are critical instruments and enablers for group communication and organization, and therefore form early artefacts of the (standardized) ability of Internet standards to enable the right to freedom of assembly and association.

2.1.5. IDNs

English has been the lingua franca of the Internet, but for many Internet user English is not their first language. To have a true global Internet, one that serves the whole world, it would need to reflect the languages of these different communities. The Internationalized Domain Names IDNA2008 (RFC 5890 [RFC5890], RFC 5891 [RFC5891], RFC 5892 [RFC5892], and RFC 5893 [RFC5893]), describes standards for the use of a broad range of strings and characters (some also written from right to left). This enables users who use other characters than the standard LDH ascii typeset to have their own URLs. This shows the ambition of the Internet community to reflect the diversity of users and to be in line with Article 2 of the Universal Declaration of Human Rights which clearly stipulates that ’everyone is entitles to all rights and freedoms [..], without distinction of any kind, such as [..] language [..].

3. Proposal

Mapping the relation between human rights and protocols and architectures is a new research challenge, which will require a good amount of cross organizational cooperation to develop a consistent methodology. While the authors of this first draft are involved in both human rights advocacy and research on Internet technologies - we believe that bringing this work into the IRTF would facilitate and improve this work by bringing human rights experts together with the community of researchers and developers of Internet standards and technologies.

At this point we have created a mailing list where we would like to encourage discussion of the issue and capture interest of the IRTF community. A second step would be to create a charter and ask the IRTF for a Research group to further develop methodology and investigate Human rights Protocol considerations.

Assuming that the research produces useful results, the objective would evolve into the creation of a set of recommended considerations for the protection of applicable human rights.

4. Acknowledgements

This builds on work done by RFC 6973 [RFC6973].

Thanks go to those who have discussed and edited the ideas in this draft. Special thanks go to Joy Liddicoat as the co-author of Human Rights and Internet Protocols [HRIP]

5. IANA Considerations

This memo includes no request to IANA.

6. Security Considerations

As this draft concerns a research proposal, there are no security considerations.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

7.2. Informative References

[HRC2011] Human Rights Council, , "Report of the Special Rapporteur on the promotion and protection of the right to freedom of opinion and expression, Human Rights Council, May 2011", 2011.

[HRC2012] General Assembly, UN., "Human Rights Council Resolution on the promotion, protection and enjoyment of human rights on the Internet", 2011, <http://daccess-ods.un.org/TMP/554342.120885849.html>.

[HRC2013] General Assembly, UN., "Report of the Special Rapporteur on the promotion and protection of the right to freedom of opinion and expression, Human Rights Council, April 2013", 2013.

[HRIP] Joy Liddicoat, JL. and AD. Avri Doria, "Human Rights and Internet Protocols: Comparing Processes and Principles", 2012, <https://www.Internetsociety.org/sites/default/files/Human%20Rights%20and%20Internet%20Protocols-%20Comparing%20Processes%20and%20Principles.pdf>.

[ICCPR] General Assembly, UN., "International Covenant on Civil and Political Rights", 1966, <http://www.ohchr.org/EN/ProfessionalInterest/Pages/CCPR.aspx>.

[NETmundial] NetMundial, , "NETmundial Multistakeholder Statement", 2014, <http://netmundial.br/wp-content/uploads/2014/04/NETmundial-Multistakeholder-Document.pdf>.

[RFC1958] Carpenter, B., "Architectural Principles of the Internet", RFC 1958, June 1996.

[RFC2014] Weinrib, A. and J. Postel, "IRTF Research Group Guidelines and Procedures", BCP 8, RFC 2014, October 1996.

[RFC2369] Neufeld, G. and J. Baer, "The Use of URLs as Meta-Syntax for Core Mail List Commands and their Transport through Message Header Fields", RFC 2369, July 1998.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

[RFC2919] Chandhok, R. and G. Wenger, "List-Id: A Structured Field and Namespace for the Identification of Mailing Lists", RFC 2919, March 2001.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, July 2003.

[RFC3869] Atkinson, R., Floyd, S., and Internet Architecture Board, "IAB Concerns and Recommendations Regarding Internet Research and Evolution", RFC 3869, August 2004.

[RFC4440] Floyd, S., Paxson, V., Falk, A., and IAB, "IAB Thoughts on the Role of the Internet Research Task Force (IRTF)", RFC 4440, March 2006.

[RFC4924] Aboba, B. and E. Davies, "Reflections on Internet Transparency", RFC 4924, July 2007.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5564] El-Sherbiny, A., Farah, M., Oueichek, I., and A. Al-Zoman, "Linguistic Guidelines for the Use of the Arabic Language in Internet Domains", RFC 5564, February 2010.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, August 2010.

[RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, August 2010.

[RFC5892] Faltstrom, P., "The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)", RFC 5892, August 2010.

[RFC5893] Alvestrand, H. and C. Karp, "Right-to-Left Scripts for Internationalized Domain Names for Applications (IDNA)", RFC 5893, August 2010.

[RFC6783] Levine, J. and R. Gellens, "Mailing Lists and Non-ASCII Addresses", RFC 6783, November 2012.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, July 2013.

[RFC7230] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, June 2014.

[RFC7231] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, June 2014.

[RFC7232] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests", RFC 7232, June 2014.

[RFC7233] Fielding, R., Lafon, Y., and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Range Requests", RFC 7233, June 2014.

[RFC7234] Fielding, R., Nottingham, M., and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, June 2014.

[RFC7235] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Authentication", RFC 7235, June 2014.

[RFC7236] Reschke, J., "Initial Hypertext Transfer Protocol (HTTP) Authentication Scheme Registrations", RFC 7236, June 2014.

[RFC7237] Reschke, J., "Initial Hypertext Transfer Protocol (HTTP) Method Registrations", RFC 7237, June 2014.

[UDHR] General Assembly, UN., "Universal Declaration of Human Rights", 1948, <http://www.ohchr.org/en/udhr/pages/introduction.aspx>.

[UNGA2013] General Assembly, UN., "UN General Assembly Resolution "The right to privacy in the digital age" (A/C.3/68/L.45)", 2013, <http://daccess-ods.un.org/TMP/1133732.05065727.html>.

Appendix A. Additional Stuff

This is a place holder for an Appendix if it is needed.

Authors' Addresses

Avri Doria
dotgay LLC
Providence
USA

Email: avri@acm.org

Niels ten Oever
Article 19
Netherlands

Email: niels@article19.org

Joana Varon
Brazil

Email: joana@varonferraz.com

NETWORK WORKING GROUP
Internet-Draft
Intended status: Informational
Expires: April 30, 2015

J. Hall
M. Aaron
Center for Democracy and Technology

October 27, 2014

A Survey of Worldwide Censorship Techniques
draft-hall-censorship-tech-00

Abstract

This document describes the technical mechanisms used by censorship regimes around the world to block or degrade internet traffic. It aims to make designers, implementers, and users of Internet protocols aware of the properties being exploited and mechanisms used to censor end-user access to information. This document makes no suggestions on individual protocol considerations, and is purely informational, intended to be a reference.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 30, 2015.

1. Introduction

2. Technical Aggregation

Aggregation is the process of figuring out what censors would like to block. Generally, censors aggregate "to block" information in three possible sorts of blacklists: Keyword, Domain Name, or IP. Keyword and Domain Name blocking take place at the application level (e.g. HTTP), whereas IP blocking tends to take place in the TCP/IP header. The mechanisms for building up these blacklists are varied. Many times private industries that sell "content control" software, such as SmartFilter, provide their services to nations which can then pick from broad categories, such as gambling or pornography, that they would like to block [ref-1]. In these cases, the private services embark on an attempt to label every semi-questionable website as to allow for this metatag blocking. Countries that are more interested in retaining specific political control, a desire which requires swift and decisive action, often have ministries or organizations, such as the Ministry of Industry and Information Technology in China or the Ministry of Culture and Islamic Guidance in Iran, which maintain their own blacklists.

3. Technical Identification

3.1. Points of Control

Digital censorship, necessarily, takes place over a network. Network design gives censors a number of different points-of-control where they can identify the content they are interested in filtering. An important aspect of pervasive technical interception is the necessity to rely on software or hardware to intercept the content the censor is interested in. This requirement, the need to have the interception mechanism located somewhere, logically or physically, implicates four general points-of-control:

- Internet Backbone: If a censor controls the gateways into a region, they can filter undesirable traffic that is traveling into and out of the region by sniffing and mirroring at the relevant exchange points. Censorship at this point-of-control is most effective at controlling the flow of information between a region and the rest of the internet, but is ineffective at identifying content traveling between the users within a region.

- Internet Service Providers: Internet Service Providers are perhaps the most natural point-of-control. They have a benefit of being

easily enumerable by a censor paired with the ability to identify the regional and international traffic of all their users. The censor's filtration mechanisms can be placed on an ISP via governmental mandates, ownership, or voluntary/coercive influence.

- Institutions: Private institutions such as corporations, schools, and cyber cafes can put filtration mechanisms in place. These mechanisms are occasionally at the request of a censor, but are more often implemented to help achieve institutional goals, such as to prevent the viewing of pornography on school computers.

- Personal Devices: Censors can mandate censorship software be installed on the device level. This has many disadvantages in terms of scalability, ease-of-circumvention, and operating system requirements. The emergence of mobile devices exacerbate these feasibility problems.

At all levels of the network hierarchy, the filtration mechanisms used to detect undesirable traffic are essentially the same: a censor sniffs transmitting packets and identifies undesirable content, and then uses a blocking or shaping mechanism to prevent or degrade access. Identification of undesirable traffic can occur at the application, transport, or network layer of the IP stack. Censors are almost always concerned with web traffic, so the relevant protocols tend to be filtered in predictable ways. For example, a subversive image would always make it past a keyword filter, but the IP address of the site serving the image may be blacklisted when identified as a provider of undesirable content.

3.2. Application Layer

3.2.1. HTTP Request Header Identification

A HTTP header contains a lot of useful information for traffic identification; although host is the only required field in a HTTP request header, a HTTP method field is necessary to do anything useful. As such, the method and host fields are the two fields used most often for ubiquitous censorship. As a censor, I can sniff traffic and identify a specific domain name (host) and usually a page name (GET /page) as well. This identification technique is usually paired with TCP/IP header identification for a more robust method. Tradeoffs: Request Identification is a technically straight-forward identification method that can be easily implemented at the Backbone or ISP level. The hardware needed for this sort of identification is cheap and easy-to-acquire, making it desirable when budget and scope are a concern. HTTPS will encrypt the relevant request and response fields, so pairing with TCP/IP identification is necessary for filtering of HTTPS. Empirical Examples: Empirical examples of pure

HTTP Request Identification are unusually hard to identify due to the lack of distinguishing charistics. Commercial technologies such as the McAfee SmartFilter and NetSweeper are often purchased by censors [ref-2]. These commercial technologies use a combination of HTTP Request Identification and TCP/IP Header Identification to filter specific URLs. There has not been research conducted to try and identify if only one of these two techniques is being used.

3.2.2. HTTP Response Header Identification

While HTTP Request Header Identification relies on the information contained in the HTTP request from client to server, response identification uses information sent in response by the server to client to identify undesirable content. Usually implemented at the Backbone or ISP level, the technique normally relies on mirroring, or duplicating the packets such that one can provide uninterrupted service while inspecting the duplicates for undesirable content, to prevent QoS degradation [ref-3] - the mirrored traffic is identified by relevant response fields (such as Server or Via). Tradeoffs: As with HTTP Request Header Identification, the techniques used to identify HTTP traffic are well-known, cheap, and relatively easy to implement, but is made useless by HTTPS, because the response in HTTPS is encrypted, including headers. The response fields are also less helpful for identifying content than request fields, as Server could easily be identified using HTTP Request Header identification, and Via is rarely relevant. HTTP Response censorship mechanisms normally let the first n packets through while the mirrored traffic is being processed; this can let a page momentarily load before blocking mechanisms kick in; giving the user a very clear indication that the censor is actively interfering with undesirable content. Empirical Examples: pointing to the "smoking-gun" examples in response header identification is difficult for the same reasons identifying requests is difficult. The best targeted evidence comes from a 2010 study conducted by Jong Park at the University of New Mexico. The study strongly indicates HTTP Response Header Identification was being used as a censorship identification technique in China from August 2008-January 2009 [ref-4].

3.2.3. Search Engine Keyword Identification

While technically similar to a HTTP request filter, the pervasiveness of search engines blacklisting search terms warrants its own attention. Search Engine Keyword Identification differentiates itself from other keyword identification techniques by being controlled by the company managing the search engine. Identification can be regional or worldwide. Implementation is occasionally voluntary, but normally is based on laws and regulations of the country a search engine is operating in. The keyword blacklists are

most likely maintained by the search engine provider. Tradeoffs: Search Engine Keyword Identification is an inconvenience as opposed to a hard block. As around half of all web traffic comes from search [ref-5], disrupting the flow of users to undesirable content is an effective method to redirect non-dedicated, curious users to less subversive content. It is also likely an effective method at encouraging self-censorship (see below) around the blocked content. Empirical Examples: Search Engine Keyword Identification is one of the easiest mechanisms to detect given the clear indicators, such as a specialized or blank results, paired with a trivial enumeration mechanism. China requires search engine providers to "voluntarily" maintain search term blacklists to acquire/keep an ICP license [ref-6]. It is clear these blacklists are maintained by each search engine provider based on the slight variations in the intercepted searches [ref-7][ref-8]. The United Kingdom has been pushing search engines to self censor with the threat of litigation if they don’t do it themselves: Google and Microsoft have agreed to block more than 100,00 queries in U.K. to help combat abuse [ref-9][ref-10].

3.2.4. Deep Packet Inspection (DPI) Identification

Deep Packet Inspection has become computationally feasible as a censorship mechanism in the past 5 years [ref-11]. DPI differs from other filtration techniques in that it examine the application "data" section of traversing packets as opposed to only the header. To prevent substantial QoS impacts, DPI normally works by splitting the traffic, using either a mirror switch or fiber splitter, and analyzing a copy of the traffic. Keyword identification is often times used to flag undesirable content. Tradeoffs: While DPI can be employed across entire networks, it is one of the most expensive technical filtration mechanisms to implement and avoiding a large impact to QoS is difficult [ref-12]. Often times a targeted approach proves itself more feasible. Any encryption on the application level, such as HTTPS, also makes DPI useless as a censorship technique as the content typically analyzed is encrypted in this case. DPI, when paired with a keyword filter, can cause major overblocking problems if used indiscriminately. Empirical Evidence: Identifying deep packet inspection censorship is non-trivial; one must be sure that the undesirable content being filtered isn’t being caught by simpler mechanisms before claiming more advanced DPI techniques are being used. The Tor project claims that China, Iran, Ethiopia, and others must being using DPI to block the obsf2 protocol [ref-13]. Malaysia has been accused of using target DPI, paired with DDoS, to identify and subsequently knockout pro-opposition material [ref-14]. It also seems likely that organizations not so worried about blocking content in real-time could use DPI to sort and categorically search gathered traffic using technologies such as NarusInsight [ref-15].

3.3. Transport Layer

3.3.1. TCP/IP Header Identification

TCP/IP Header Identification is the most pervasive, reliable, and predictable type of identification. TCP/IP headers contain a few invaluable pieces of information that must be transparent for traffic to be successfully routed: destination and source IP address and port. Destination and Source IP are doubly useful, as not only does it allow a cto block undesirable content via IP blacklisting, but also allows a censor to identify the IP of the user making the request. Port is useful for whitelisting certain applications or forcing an HTTP proxy for non-technical users. Trade-offs: This method of filtration is popular due to its simplicity, relative cheapness, and wide availability. It is trivial to implement a filtration mechanism at the Backbone, ISP, or Institutional level that compares the IP address of a packet with a blacklist of IP addresses. IP blocking is relatively crude, often leading to overblocking, and one of the simplest to circumvent via VPN or proxy as those either mask transport protocol within a tunnel or reroute data that might have been blocked otherwise. Port blocking is semi-effective at best. A censor can block communication on the default port of an undesirable application (for example uTorrent defaults to 32459), but almost all applications allow the user to change ports. Port whitelisting, where a censor only allow communication on approved ports, such as 80 for HTTP traffic, is more often used. This identification mechanism is often used in conjunction with HTTP Identification. Empirical Examples: TCP/IP Header Identification is pervasive. Some form of TCP/IP Header Identification is used by most, if not all, ISP and backbone censors. Any time an IP blacklist is being used, TCP/IP Header Identification is probably the technique being used to match the request against the blacklist. The examples of TCP/IP Header Identification are too numerous to enumerate in any meaningful way.

3.3.2. Protocol Identification

Protocol identification is a network analysis technique where one attempts to identify the protocols being used based on a variety of traffic characteristics. There have been a number well documented cases where traffic identification has been used to filter undesirable protocols. A very simple example of traffic identification would be to recognize all TCP traffic over port 80 as HTTP, but much more sophisticated methods, such as analyzing statistical properties of payload data and flow behavior, have been used [ref-16][ref-17]. Trade-offs: Protocol Identification necessarily only provides insight into the way information is traveling, and not the information itself. This can lead to massive

overblocking problems if used with popular protocols. Most often undesirable protocols are those which can be used to transmit information that is otherwise hard to analyze or considered to likely cary undesirable information; VoIP, P2P, SSL, and Tor have all been targets of protocol identification in the past. As statistical analysis is used, the methods tend to be expensive, both computationally and financially, and are occasionally imprecise and under-filter obfuscated protocols. Empirical Examples: Protocol Identification is easy to prove given the ubiquitous nature of the throttling/interruption; If only a specific protocol(s) are being prevented, then Protocol Identification is the most likely culprit. Iran censors have used Protocol Identification to identify and throttle SSH traffic by such a large amount as to make it unusable [ref-18]. The method used by censors in China to identify Tor connections could also be viewed as a type of Protocol Identification[ref-19]. Protocol Identification has also been used by industry from traffic management, such as the 2007 case where Comcast in the United States was using RST injection to interrupt BitTorrent Traffic [ref-20].

4. Technical Prevention

4.1. Packet Dropping

Packet dropping is a simple mechanism to prevent undesirable traffic. The censor identifies undesirable traffic and chooses to not properly forward any packets it sees associated with the traversing undesirable traffic instead of following a normal routing protocol. This can be paired with any of the previously described mechanisms so long as the censor knows the user must route traffic through a controlled router. Trade offs: Packet Dropping is most successful when every traversing packet has transparent information linked to undesirable content, such as a Destination IP. One downside Packet Dropping suffers from is the necessity of overblocking all content from otherwise allowable IP’s based on a single subversive sub-domain; blogging services and github repositories are good examples. China famously dropped all github packets for three days based on a single repository hosting undesirable content [ref-21]. The need to inspect every traversing packet in close to real time also makes Packet Dropping somewhat challenging from a QoS perspective. Empirical Examples: Packet Dropping is a very common form of technical prevention and lends itself to accurate detection given the unique nature of the time-out requests it leaves in its wake. The Great Firewall of China uses packet dropping as one of its primary mechanisms of technical censorship [ref-22]. Iran also uses Packet Dropping as the mechanisms for throttling SSH [ref-23]. These are but two examples of a ubiquitous censorship practice.

4.2. RST Packet Injection

Packet injection, generally, refers to a MITM network interference technique that spoofs packets in an established traffic stream. RST packets are normally used to let one side of TCP connection know the other side has stopped sending information, and thus the receiver should close the connection. RST Packet Injection is a specific type of packet injection attack that is used to interrupt an established stream by sending RST packets to both sides of a TCP connection; as each receiver thinks the other has dropped the connection, the session is terminated. Trade-offs: RST Packet Injection has a few advantages that make it extremely popular is a censorship technique. RST Packet Injection is an out-of-band prevention mechanism, allowing the avoidance of the the QoS bottleneck one can encounter with inline techniques such as Packet Dropping. This out-of-band property allows a censor to inspect a copy of the information, usually mirrored by an optical splitter, making it an ideal pairing for DPI and Protocol Identification[ref-24]. RST Packet Injection also has the advantage of only requiring one of the two endpoints to accept the spoofed packet for the connection to be interrupted[ref-25]. The difficult part of RST Packet Injection is spoofing "enough" correct information to ensure one end-point accepts a RST packet as legitimate; this generally implies a correct IP, port, and sequence number. Sequence number is the hardest to get correct, as RFC 793 specifies an RST Packet should be in-sequence to be accepted, although the RFC also recommends allowing in-window packets as "good enough"[ref-26]. This in-window recommendation is important, as if it is implement it allows for successful Blind RST Injection attacks[ref-27]. When in-window sequencing is allowed, It is trivial to conduct a Blind RST Injection, a blind injection implies the censor doesn’t know any sensitive (encrypted) sequencing information about the TCP stream they are injecting into, they can simply enumerate the ˜70000 possible windows; this is particularly useful for interrupting encrypted/obfuscated protocols such as SSH or Tor. RST Packet Injection relies on a stateful network, making it useless against UDP connections. RST Packet Injection is among the most popular censorship techniques used today given its versatile nature and effectiveness against all types of TCP traffic. Empirical Examples: RST Packet Injection, as mentioned above, is most often paired with identification techniques that require splitting, such as DPI or Protocol Identification. In 2007 Comcast was accused of using RST Packet Injection to interrupt traffic it identified as BitTorrent [ref-28], this later led to a FCC ruling against Comcast [ref-29]. China has also been known to use RST Packet Injection for censorship purposes. This prevention is especially evident in the interruption of encrypted/obfuscated protocols, such as those used by Tor [ref-30].

4.3. DNS Cache Poisoning

DNS Cache Poisoning refers to a mechanism where a censor interferes with the response sent by a DNS resolver to the requesting device by injecting an alternative IP address on the return path. Cache poisoning occurs after the requested site's name servers resolve the request and attempt to forward the IP back to the requesting device; on the return route the resolved IP is recursively cached by each DNS servers that initially forwarded the request. During this caching process if an undesirable keyword is recognized, the resolved IP is poisoned and an alternative IP is returned. These alternative IP's usually direct to a nonsense domain or a warning page[ref-31]. Alternatively, Iranian censorship appears to prevent the communication en-route, preventing a response from ever being sent[ref-32]. Trade-offs: DNS Cache Poisoning is one of the rarer forms of prevention due to a number of shortcomings. DNS Cache Poisoning requires the censor to force a user to traverse a controlled DNS resolver for the mechanism to be effective, it is easily circumvented by a technical savvy user that opts to use alternative DNS resolvers, such as the 8.8.8.8/8.8.4.4 public DNS resolvers provided by Google. DNS Cache Poisoning also implies returning an incorrect IP to those attempting to resolve a domain name, but the site is still technically unblocked if the user has another method to acquire the IP address of the desired site. Blocking overflow has also been a problem, as occasionally users outside of the censors region will be directed through a DNS server controlled by a censor, causing the request to fail. The ease of circumvention paired with the large risk of overblocking and blocking overflow make DNS Cache Poisoning a partial, difficult, and less than ideal censorship mechanism. Empirical Evidence: DNS Cache Poisoning, when properly implemented, is easy to identify based on the shortcomings identified above. Turkey relied on DNS Cache Poisoning for its country-wide block of websites such Twitter and Youtube for almost week in March of 2014 but the ease of circumvention resulted in an increase in the popularity of Twitter until Turkish ISP's implementing an IP blacklist to achieve the governmental mandate[ref-33]. To drive proverbial "nail in the coffin" Turkish ISPs started hijacking all requests to Google and Level 3's international DNS resolvers [ref-34]. DNS Cache Poisoning, when incorrectly implemented, has as has resulted in some of the largest "censorship disasters". In January 2014 China started directing all requests passing through the Great Fire Wall to a single domain, dongtaiwang.com, due to an improperly configured DNS Cache Poisoning attempt; this incident is thought to be the largest internet-service outage in history [ref-35][ref-36]. Countries such as China, Iran, Turkey, and the United States have discussed blocking entire TLDs as well, but only Iran has acted by blocking all Israeli (.il) domains [ref-37].

4.4. Distributed Denial of Service (DDoS)

Distributed Denial of Service attacks are a common attack mechanism used by "hacktivists" and black-hat hackers, but censors have used DDoS in the past for a variety of reasons. There is a huge variety of DDoS attacks[ref-38], but on a high level two possible impacts tend to occur; a flood attack results in the service being unusable while resources are being spent to flood the service, a crash attack aims to crash the service so resources can be reallocated elsewhere without "releasing" the service. Trade-offs: DDoS is an appealing mechanism when a censor would like to prevent all access to undesirable content, instead of only access in their region for a limited period of time, but this is really the only uniquely beneficial feature for DDoS as a censorship technique. The resources required to carry out a successful DDoS against major targets are computationally expensive, usually requiring renting or owning a malicious distributed platform such as a botnet, and imprecise. DDoS is an incredibly crude censorship technique, and appears to largely be used as a timely, easy-to-access mechanism for blocking undesirable content for a limited period of time. Empirical Examples: In 2012 the U.K.’s GCHQ used DDoS to temporarily shutdown IRC chat rooms frequented by members of Anonymous using the Syn Flood DDoS method; Syn Flood exploits the handshake used by TCP to overload the victim server with so many requests that legitimate traffic becomes slow or impossible [ref-39][ref-40]. Dissenting opinion websites are frequently victims of DDoS around politically sensitive events in Burma [ref-41]. Controlling parties in Russia[ref-42], Zimbabwe[ref-43], and Malaysia[ref-44] have been accused of using DDoS to interrupt opposition support and access during elections.

4.5. Network Disconnection or Adversarial Route Announcement

Network Disconnection or Adversarial Route Announcement The crudest of all censorship techniques, there is no more effective way of making sure undesirable information isn’t allowed to propagate on the web than by shutting off the network. The network can be cut off in a region when a censoring body withdraws all of the BGP prefixes routing through the censor’s country. Trade-offs: The impact to a network disconnection in a region is huge and absolute; the censor pays for absolute control over digital information with all the benefits the internet brings; this is never a long-term solution for any rational censor and is normally only used as a last resort in times of substantial unrest. Empirical Examples: Network Disconnections tend to only happen in times of substantial unrest, largely due to the huge social, political, and economic impact such a move has. One of the first, highly covered occurrences was with the Junta in Myanmar employing Network Disconnection to help Junta forces quash a rebellion in 2007 [ref-45]. China disconnected the network

in the Xinjiang region during unrest in 2009 in an effort to prevent the protests from spreading to other regions [ref-46]. The Arab Spring saw the the most frequent usage of Network Disconnection, with events in Egypt and Libya in 2011 [ref-47][ref-48], and Syria in 2012 [ref-49].

5. Non-Technical Aggregation

As the name implies, sometimes manpower is the easiest way to figure out which content to block. Manual Filtering differs from the common tactic of building up blacklists in that is doesn't necessarily target a specific IP or DNS, but instead removes or flags content. Given the imprecise nature of automatic filtering, manually sorting through content and flagging dissenting websites, blogs, articles and other media for filtration can be an effective technique. This filtration can occur on the Backbone/ISP level, China's army of monitors is a good example [ref-50]; more commonly manual filtering occurs on an institutional level. ICP's, such as Google or Weibo, require a business license to operate in China. One of the prerequisites for a business license is an agreement to sign a "voluntary pledge" known as the "Public Pledge on Self-discipline for the Chinese Internet Industry". The failure to " energetically uphold" the pledged values can lead to the ICP's being held liable for the offending content by the Chinese government [ref-51].

6. Non-Technical Prevention

6.1. Self Censorship

Self censorship is one of the most interesting and effective types of censorship; a mix of Bentham's Panopticon, cultural manipulation, intelligence gathering, and meatspace enforcement. Simply put, self censorship is when a censor creates an atmosphere where users censor themselves. This can be achieved through controlling information, intimidating would-be dissidents, swaying public thought, and creating apathy. Self censorship is difficult to document, as when it is implemented effectively the only noticeable tracing is a lack of undesirable content; instead one must look at the tools and techniques used by censors to encourage self-censorship. Controlling Information relies on traditional censorship techniques, or by forcing all users to connect through an intranet, such as in North Korea. Intimidation is often achieved through allowing internet users to post "whatever they want", but arresting those who post about dissenting views, this technique is incredibly common[ref-52][ref-53][ref-54][ref-55][ref-56]. A good example of swaying public thought is China's "50-Cent Party", composed of somewhere between 20,000[ref-57] and 300,000[ref-58] contributors who are paid to "guide public thought" on local and regional issues as

directed by the Ministry of Culture. Creating apathy can be a side-effect of successfully controlling information over time and is ideal for a censorship regime [ref-59].

6.2. Domain Name Reallocation

As Domain Names are resolved recursively, if a TLD deregisters a domain all other DNS resolvers will be unable to properly forward and cache the site. Domain name registration is only really a risk where undesirable content is hosted on TLD controlled by the censoring country, such as .ch or .ru [ref-60].

6.3. Server Takedown

Servers must have a physical location somewhere in the world. If undesirable content is hosted in the censoring country the servers can be physically seized or the hosting provider can be required to prevent access [ref-61].

7. References

[ref-1] Glanville, J., ""The Big Business of Net Censorship"", November 2008 , <http://www.theguardian.com/commentisfree/2008/nov/17/censorship-internet>.

[ref-2] Dalek, J., ""A Method for Identifying and Confirming the Use of URL Filtering Products for Censorship"", October 2013 , <http://www.cs.stonybrook.edu/~phillipa/papers/imc112s-dalek.pdf>.

[ref-3] EF, A., ""EFA Filtering Overview"", May 2009 , <https://www.efa.org.au/main/wp-content/uploads/2009/05/efa-filtering-fact-sheets.pdf>.

[ref-4] Crandall, J., ""Empirical Study of a National-Scale Distributed Intrusion Detection System: Backbone-Level Filtering of HTML Responses in China"", June 2010 , <http://www.cs.unm.edu/~crandall/icdcs2010.pdf >.

[ref-5] Dobie, M., ""Junta Tightens Military Screws"", September 2007 , <http://news.bbc.co.uk/2/hi/asia-pacific/7016238.stm>.

[ref-6] Cheng, J., ""Google stops Hong Kong auto-redirect as China plays hardball"", June 2010, <http://arstechnica.com/tech-policy/2010/06/google-tweaks-china-to-hong-kong-redirect-same-results/>.

[ref-7] Zhu, T., ""An Analysis of Chinese Search Engine Filtering"", July 2011 , <http://arxiv.org/ftp/arxiv/papers/1107/1107.3794.pdf#page=10>.

[ref-8] Whittaker, Z., ""1,168 keywords Skype uses to censor, monitor its Chinese users"", March 2013 , <http://www.zdnet.com/1168-keywords-skype-uses-to-censor-monitor-its-chinese-users-7000012328/>.

[ref-9] News, B., ""Google and Microsoft agree steps to block abuse images"", November 2013 , <http://www.bbc.com/news/uk-24980765>.

[ref-10] Condliffe, J., ""Google Announces Massive New Restrictions on Child Abuse Search Terms"", November 2013 , <http://gizmodo.com/google-announces-massive-new-restrictions-on-child-abus-1466539163>.

[ref-11] Wagner, B., ""Deep Packet Inspection and Internet Censorship: International Convergence on an 'Integrated Technology of Control'"", June 2009 , <http://advocacy.globalvoicesonline.org/wp-content/uploads/2009/06/deeppacketinspectionandinternet-censorship2.pdf>.

[ref-12] Porter, T., ""The Perils of Deep Packet Inspection"", Oct 2010, <http://www.symantec.com/connect/articles/perils-deep-packet-inspection>.

[ref-13] Wilde, T., ""Knock Knock Knockin' on Bridges Doors"", January 2012, <https://blog.torproject.org/blog/knock-knock-knockin-bridges-doors>.

[ref-14] Wagstaff, J., ""In Malaysia, online election battles take a nasty turn"", May 2013, <http://www.reuters.com/article/2013/05/04/uk-malaysia-election-online-idUKBRE94309G20130504>.

[ref-15] EFF, T., ""Hepting vs. ATand T"", Updated December, <https://www.eff.org/cases/hepting>.

[ref-16] Hjelmvik, E., "July 2010 7", "Breaking and, <https://www.iis.se/docs/hjelmvik_breaking.pdf>.

[ref-17] Vine, S., ""Technology Showcase on Traffic Classification: Why Measurements and Freeform Policy Matter"", May 2014, <https://www.sandvine.com/downloads/general/technology/sandvine-technology-showcases/sandvine-technology-showcase-traffic-classification.pdf#page=3 >.

[ref-18] Anonymous, A., ""How to Bypass Comcast's Bittorrent Throttling"", October 2007, <https://torrentfreak.com/how-to-bypass-comcast-bittorrent-throttling-071021>.

[ref-19] Lee, T., ""Here's how Iran censors the Internet"", August 2013, <http://www.washingtonpost.com/blogs/the-switch/wp/2013/08/15/heres-how-iran-censors-the-internet/>.

[ref-20] Winter, P., ""How China is Blocking Tor"", April 2012, <http://arxiv.org/pdf/1204.0447v1.pdf21>.

[ref-21] Anonymous, A., ""GitHub blocked in China - how it happened, how to get around it, and where it will take us"", January 2013, <https://en.greatfire.org/blog/2013/jan/github-blocked-china-how-it-happened-how-get-around-it-and-where-it-will-take-us>.

[ref-22] Ensafi, R., ""Detecting Intentional Packet Drops on the Internet via TCP/IP Side Channels"", December 2013, <http://arxiv.org/pdf/1312.5739v1.pdf>.

[ref-23] Aryan*, A., ""Internet Censorship in Iran: A First Look"", August 2013 , <https://jhalderm.com/pub/papers/iran-foci13.pdf>.

[ref-24] Weaver, S., ""Detecting Forged TCP Packets"", June 2009 , <http://www.icir.org/vern/papers/reset-injection.ndss09.pdf>.

[ref-25] Weaver, S., ""Detecting Forged TCP Packets"", June 2009 , <http://www.icir.org/vern/papers/reset-injection.ndss09.pdf>.

[ref-26] Weaver, S., ""Detecting Forged TCP Packets"", June 2009 , <http://www.icir.org/vern/papers/reset-injection.ndss09.pdf>.

[ref-27] Anonymous, A., ""TCP-RST Injection"", June 210 , <http://www.blackhatlibrary.net/TCP-RST_Injection >.

[ref-28] Schoen, S., ""EFF tests agree with AP: Comcast is forging packets to interfere with user traffic"", October 19th,, <https://www.eff.org/deeplinks/2007/10/eff-tests-agree-ap-comcast-forging-packets-to-interfere>.

[ref-29] VonLohmann, F., ""FCC Rules Against Comcast for BitTorrent Blocking"", August 3rd,, <https://www.eff.org/deeplinks/2008/08/fcc-rules-against-comcast-bit-torrent-blocking>.

[ref-30] Phillip Winter, S., ""How China Is Blocking Tor"", April 2nd,, <http://arxiv.org/pdf/1204.0447v1.pdf#page=5>.

[ref-31] DNS, V., ""DNS Cache Poisoning in the People's Republic of China"", September 6th, <http://viewdns.info/research/dns-cache-poisoning-in-the-peoples-republic-of-china/>.

[ref-32] Aryan*, A., ""Internet Censorship in Iran: A First Look"", August 2013 , <https://jhalderm.com/pub/papers/iran-foci13.pdf#page=5>.

[ref-33] Zmijewki, E., ""Turkish Internet Censorship Takes a New Turn"", March 2014, <http://www.renesys.com/2014/03/turkish-internet-censorship/>.

[ref-34] Zmijewki, E., ""Turkish Internet Censorship Takes a New Turn"", March 2014, <http://www.renesys.com/2014/03/turkish-internet-censorship/>.

[ref-35] AFP, .A., ""China Has Massive Internet Breakdown Reportedly Caused By Their Own Censoring Tools"", January 2014, <http://www.businessinsider.com/chinas-internet-breakdown-reportedly-caused-by-censoring-tools-2014-1>.

[ref-36] Anonymous, A., ""The Collateral Damage of Internet Censorship by DNS Injection"", July 2012 , <http://www.sigcomm.org/sites/default/files/ccr/papers/2012/July/2317307-2317311.pdf>.

[ref-37] Albert, K., ""DNS Tampering and the new ICANN gTLD Rules"", June 2011, <https://opennet.net/blog/2011/06/dns-tampering-and-new-icann-gtld-rules>.

[ref-38] Anonymous, A., ""Denial of Service Attacks (Wikipedia)"", N/A N/A, <http://en.wikipedia.org/wiki/Denial-of-service_attack#Methods_of_attack>.

[ref-39] Esposito, S., ""Snowden Docs Show UK Spies Attacked Anonymous, Hackers"", February 2014, <http://www.nbcnews.com/feature/edward-snowden-interview/exclusive-snowden-docs-show-uk-spies-attacked-anonymous-hackers-n21361>.

[ref-40] CMU, .C., ""TCP SYN Flooding and IP Spoofing Attacks"", November 2000, <http://www.cert.org/historical/advisories/CA-1996-21.cfm>.

[ref-41] Villeneuve, N., ""Open Access: Chapter 8, Control and Resistance, Attacks on Burmese Opposition Media"", December 2011 , <http://access.opennet.net/wp-content/uploads/2011/12/accesscontested-chapter-08.pdf>.

[ref-42] Kravtsova, Y., ""Cyberattacks Disrupt Opposition's Election"", October 2012, <http://www.themoscowtimes.com/news/article/cyberattacks-disrupt-oppositions-election/470119.html>.

[ref-43] Orion, E., ""Zimbabwe election hit by hacking and DDoS attacks"", August 2013, <http://www.theinquirer.net/inquirer/news/2287433/zimbabwe-election-hit-by-hacking-and-ddos-attacks>.

[ref-44] Muncaster, P., ""Malaysian election sparks web blocking/ DDoS claims"", May 2013, <http://www.theregister.co.uk/2013/05/09/malaysia_fraud_elections_ddos_web_blocking/>.

[ref-45] Dobie, M., ""Junta tightens media screw"", September 2007, <http://news.bbc.co.uk/2/hi/asia-pacific/7016238.stm>.

[ref-46] Heacock, R., ""China Shuts Down Internet in Xinjiang Region After Riots"", July 2009, <https://opennet.net/blog/2009/07/china-shuts-down-internet-xinjiang-region-after-riots>.

[ref-47] Cowie, J., ""Egypt Leaves the Internet"", January 2011, <http://www.renesys.com/2011/01/egypt-leaves-the-internet/>.

[ref-48] Cowie, J., ""Libyan Disconnect"", February 2011, <http://www.renesys.com/2011/02/libyan-disconnect-1/>.

[ref-49] Thomson, I., ""Syria Cuts off Internet and Mobile Communication"", November 2012, <http://www.theregister.co.uk/2012/11/29/syria_internet_blackout/>.

[ref-50] News, B., ""China employs two million microblog monitors state media say"", October 2013, <http://www.bbc.com/news/world-asia-china-24396957>.

[ref-51] MacKinnon, R., ""'Race to the Bottom' Corporate Complicity in Chinese Internet Censorship"", August 2006 , <http://www.hrw.org/reports/2006/china0806/china0806web.pdf>.

[ref-52] Calamur, K., ""Prominent Egyptian Blogger Arrested"", November 2013, <http://www.npr.org/blogs/thetwo-way/2013/11/29/247820503/prominent-egyptian-blogger-arrested>.

[ref-53] Press, A., ""Sattar Beheshit, Iranian Blogger, Was Beaten In Prison According To Prosecutor"", December 2012, <http://www.huffingtonpost.com/2012/12/03/sattar-beheshit-iran_n_2233125.html>.

[ref-54] Hopkins, C., ""Communications Blocked in Libya, Qatari Blogger Arrested: This Week in Online Tyranny"", March 2011, <http://readwrite.com/2011/03/03/communications_blocked_in_libya_this_week_in_onlin>.

[ref-55] Gaurdian, T., ""Chinese blogger jailed under crackdown on 'internet rumours'"", April 2014, <http://www.theguardian.com/world/2014/apr/17/chinese-blogger-jailed-crackdown-internet-rumours-qin-zhihui>.

[ref-56] Johnson, L., ""Torture feared in arrest of Iraqi blogger"", Febuary 2010, <http://seattlepostglobe.org/2010/02/05/torture-feared-in-arrest-of-iraqi-blogger/>.

[ref-57] Bristow, M., ""China's internet 'spin doctors'"", November 2013, <http://news.bbc.co.uk/2/hi/asia-pacific/7783640.stm>.

[ref-58] Fareed, M., ""China joins a turf war"", September 2008, <http://www.theguardian.com/media/2008/sep/22/chinathemedia.marketingandpr>.

[ref-59] Gao, H., ""Tiananmen, Forgotten"", June 2014, <http://www.nytimes.com/2014/06/04/opinion/tiananmen-forgotten.html>.

[ref-60] Anderson, R., ""Access Denied: Tools and Technology of Internet Filtering"", December 2011 , <http://access.opennet.net/wp-content/uploads/2011/12/accessdenied-chapter-3.pdf#page=8>.

[ref-61] Murdoch, S., ""Access Denied: Tools and Technology of Internet Filtering"", December 2011 , <http://access.opennet.net/wp-content/uploads/2011/12/accessdenied-chapter-3.pdf#page=8>.

Authors' Addresses

Joeseph L. Hall
Center for Democracy and Technology

Email: jhall@cdt.org

Michael D. Aaron
Center for Democracy and Technology

Email: maaron@cdt.org

NTP Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 26, 2015

D. Sibold
PTB
S. Roettger
Google Inc
K. Teichel
PTB
R. Housley
Vigil Security
October 23, 2014

Protecting Network Time Security Messages with the Cryptographic Message Syntax (CMS)

draft-ietf-ntp-cms-for-nts-message-00.txt

Abstract

This document describes a convention for using the Cryptographic Message Syntax (CMS) to protect the messages in the Network Time Security (NTS) protocol. NTS provides authentication of time servers as well as integrity protection of time synchronization messages using Network Time Protocol (NTP) or Precision Time Protocol (PTP).

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

Table of Contents

1. Introduction

This document provides detail on how to construct NTS messages in practice. NTS provides secure time synchronization with time servers using Network Time Protocol (NTP) [RFC5905] or Precision Time Protocol (PTP) [IEEE1588]. Among other things, this document

describes a convention for using the Cryptographic Message Syntax (CMS) [RFC5652] to protect messages in the Network Time Security (NTS) protocol. Encryption is used to provide confidentiality of secrets, and digital signatures are used to provide authentication and integrity of content.

Sometimes CMS is used in an exclusively ASN.1 [ASN1] environment. In this case, the NTS message may use any syntax that facilitates easy implementation.

2. CMS Conventions for NTS Message Protection

Regarding the usage of CMS we differentiate between four archetypes according to which the NTS message types can be structured:

NTS-Plain: This archetype is used for actual time synchronization messages (explicitly, the message types: time_request, time_response, server_broad, see [I-D.ietf-ntp-network-time-security], section 6) as well as for the very first messages of a unicast or a broadcast exchange (client_assoc or client_bpar, respectively) and the broadcast keycheck exchange (client_keycheck and server_keycheck). This archetype does not make use of any CMS structures.

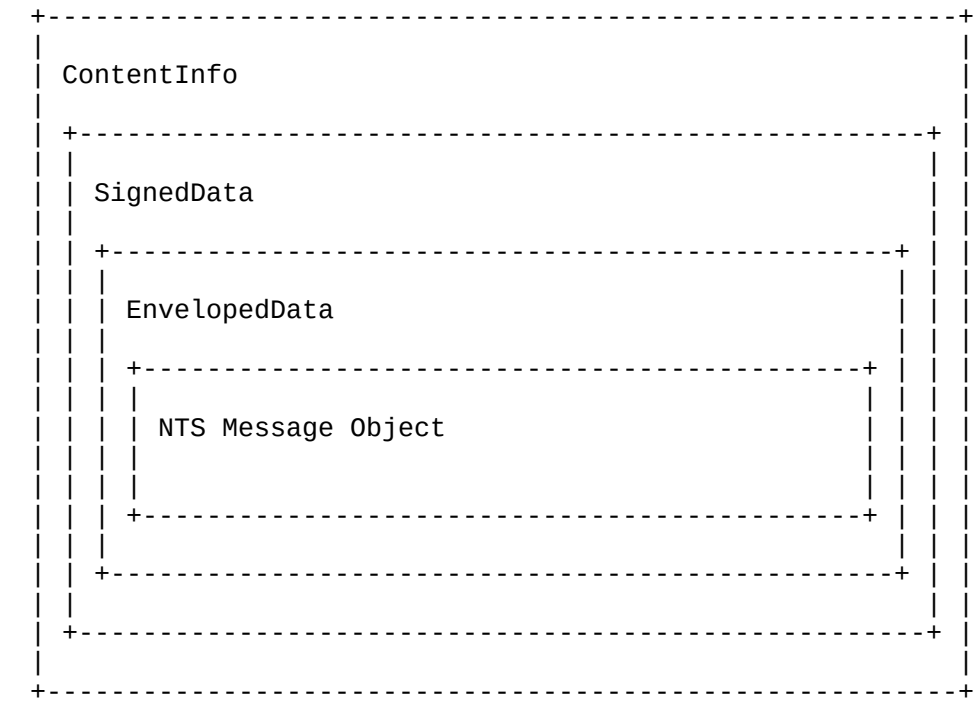
NTS-Signed-and-Encrypted: This archetype is used for secure transmission of the cookie (only for the server_cook message type, see [I-D.ietf-ntp-network-time-security], section 6). For this, the following CMS structure is used:

First, the NTS message MUST be encrypted using the EnvelopedData content type. EnvelopedData supports nearly any form of key management. In the NTS protocol the client provides a certificate in an unprotected message, and the public key from this certificate, if it is valid, will be used to establish a pairwise symmetric key for the encryption of the protected NTS message.

Second, the EnvelopedData content MUST be digitally signed using the SignedData content type. SignedData supports nearly any form of digital signature, and in the NTS protocol the server will include its certificate within the SignedData content type.

Third, the SignedData content type MUST be encapsulated in a ContentInfo content type.

Figure 1 illustrates this structure.

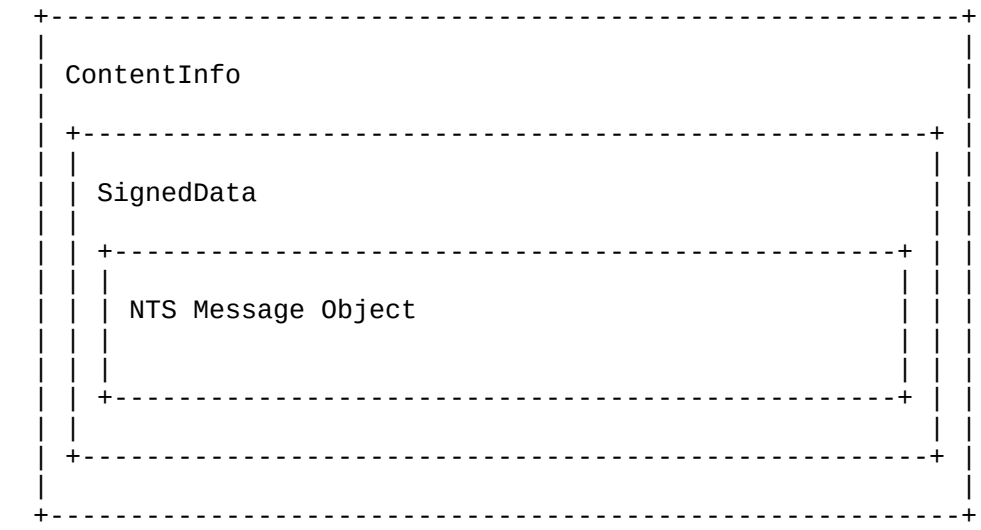


NTS-Signed: This archetype is used for server_assoc and server_bpar message types. It uses the following CMS structure:

First, the NTS message object MUST be wrapped in a SignedData content type. The messages MUST be digitally signed, and certificates included. SignedData supports nearly any form of digital signature, and in the NTS protocol the server will include its certificate within the SignedData content type.

Second, the SignedData content type MUST be encapsulated in a ContentInfo content type.

Figure 2 illustrates this structure.



NTS-Certified: This archetype is used for the client_cook message type. It uses a CMS structure much like the NTS-Signed archetype (see Figure 2), with the only difference being that messages SHOULD NOT be digitally signed. This archetype employs the CMS structure merely in order to transport certificates.

Whichever archetype is used, the resulting structure is always transported in an extension field of an NTP packet. In the case of messages that also need to carry time synchronization data, this data is written into the regular fields of the NTP packet.

2.1. Fields of the employed CMS Content Types

Overall, three CMS content types are used for NTS messages: ContentInfo, SignedData and EnvelopedData. The following is a description of how the fields of those content types are used in detail.

2.1.1. ContentInfo

The ContentInfo content type is used in all four archetypes. The fields of the SignedData content type are used as follows:

contentType -- indicates the type of the associated content. For the archetype NTS-Plain, it MUST identify the NTS message object that is included. For all other archetypes (NTS-Certified, NTS-Signed and NTS-Signed-and-Encrypted), it MUST contain the object identifier for the SignedData content type:

```
id-signedData OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) rsadsi(113549) pkcs(1) pkcs7(7) 2 }
```

content is the associated content. For the NTS-Plain archetype, it MUST contain the DER encoded NTS message object. For all other archetypes, it MUST contain the DER encoded SignedData content type.

2.1.2. SignedData

The SignedData content type is used in the NTS-Certified, NTS-Signed and NTS-Signed-and-Encrypted archetypes but not in the NTS-Plain archetype. The fields of the SignedData content type are used as follows:

version -- the appropriate value depends on the optional items that are included. In the NTS protocol, the signer certificate MUST be included, and other items MAY be included. The instructions in [RFC5652] section 5.1 MUST be followed to set the correct value.

digestAlgorithms -- is a collection of message digest algorithm identifiers. In the NTS protocol, there MUST be exactly one algorithm identifier present. The instructions in Section 5.4 of [RFC5652] MUST be followed.

encapContentInfo -- this structure is always present. In the NTS protocol, it MUST follow these conventions:

eContentType -- is an object identifier. In the NTS protocol, for the NTS-Certified and NTS-Signed archetypes, it MUST identify the type of the NTS message that was encapsulated. For the NTS-Signed-and-Encrypted archetype, it MUST contain the object identifier for the EnvelopedData content type:

```
id-envelopedData OBJECT IDENTIFIER ::= { iso(1) member-body(2)
       us(840) rsadsi(113549) pkcs(1) pkcs7(7) 3 }.
```

eContent is the content itself, carried as an octet string. For the NTS-Certified and NTS-Signed archetypes, it MUST contain the DER encoded encapsulated NTS message object. The instructions in Section 6.3 of [RFC5652] MUST be followed. For the NTS-Signed-and-Encrypted archetype, it MUST contain the DER encoded EnvelopedData content type.

certificates -- is a collection of certificates. In the NTS protocol, it MUST contain the DER encoded certificate [RFC5280] of the sender. It is intended that the collection of certificates be

sufficient for the recipient to construct a certification path from a recognized "root" or "top-level certification authority" to the certificate used by the sender.

crls -- is a collection of revocation status information. In the NTS protocol, it MAY contain one or more DER encoded CRLs [RFC5280]. It is intended that the collection contain information sufficient to determine whether the certificates in the certificates field are valid.

signerInfos -- is a collection of per-signer information. In the NTS protocol, for the NTS-Certified archetype, this SHOULD be left out. For both the NTS-Signed and the NTS-Signed-and-Encrypted archetypes, there MUST be exactly one SignerInfo structure present. The details of the SignerInfo type are discussed in Section 5.3 of [RFC5652]. In the NTS protocol, it MUST follow these conventions:

version -- is the syntax version number. In the NTS protocol, the SignerIdentifier is subjectKeyIdentifier, therefore the version MUST be 3.

sid -- identifies the signer’s certificate. In the NTS protocol, the sid field contains the subjectKeyIdentifier from the signer’s certificate.

digestAlgorithm -- identifies the message digest algorithm, and any associated parameters, used by the signer. In the NTS protocol, the identifier MUST match the single algorithm identifier present in the digestAlgorithms.

signedAttrs -- is a collection of attributes that are signed. In the NTS protocol, it MUST be present, and it MUST contain the following attributes:

Content Type -- see Section 11.1 of [RFC5652].

Message Digest -- see Section 11.2 of [RFC5652].

In addition, it MAY contain the following attributes:

Signing Time -- see Section 11.3 of [RFC5652].

Binary Signing Time -- see Section 3 of [RFC5652].

signatureAlgorithm -- identifies the signature algorithm, and any associated parameters, used by the signer to generate the digital signature.

signature is the result of digital signature generation, using the message digest and the signer's private key. The instructions in Section 5.5 of [RFC5652] MUST be followed.

unsignedAttrs -- is an optional collection of attributes that are not signed. In the NTS protocol, the it MUST be absent.

2.1.3. EnvelopedData

The EnvelopedData content type is used only in the NTS-Signed-and-Encrypted archetype. The fields of the EnvelopedData content type are used as follows:

version -- the appropriate value depends on the type of key management that is used. The instructions in [RFC5652] section 6.1 MUST be followed to set the correct value.

originatorInfo -- this structure is present only if required by the key management algorithm. In the NTS protocol, it MUST be present when a key agreement algorithm is used, and it MUST absent when a key transport algorithm is used. The instructions in Section 6.1 of [RFC5652] MUST be followed.

recipientInfos -- this structure is always present. In the NTS protocol, it MUST contain exactly one entry that allows the client to determine the key used to encrypt the NTS message. The instructions in Section 6.2 of [RFC5652] MUST be followed.

encryptedContentInfo -- this structure is always present. In the NTS protocol, it MUST follow these conventions:

contentType -- indicates the type of content. In the NTS protocol, it MUST identify the type of the NTS message that was encrypted.

contentEncryptionAlgorithm -- identifies the content-encryption algorithm, and any associated parameters, used to encrypt the content.

encryptedContent -- is the encrypted the content. In the NTS protocol, it MUST contain the encrypted NTS message. The instructions in Section 6.3 of [RFC5652] MUST be followed.

unprotectedAttrs -- this structure is optional. In the NTS protocol, it MUST be absent.

3. Certificate Conventions

The syntax and processing rules for certificates are specified in [RFC5652]. In the NTS protocol, the server certificate MUST contain the following extensions:

Subject Key Identifier -- see Section 4.2.1.2 of [RFC5652].

Key Usage -- see Section 4.2.1.3 of [RFC5652].

Extended Key Usage -- see Section 4.2.1.22 of [RFC5652].

The Extended Key Usage extension MUST include the id-kp-NTSserver object identifier. When a certificate issuer includes this object identifier in the extended key usage extension, it provides an attestation that the certificate subject is a time server that supports the NTS protocol.

The id-kp-NTSserver object identifier is:

```
id-kp-NTSserver OBJECT IDENTIFIER ::= { TBD }
```

4. Implementation Notes: ASN.1 Structures and Use of the CMS

This section gives some hints on the structures of the NTS message objects for the different message types when one wishes to implement the protocol.

4.1. Preliminaries

The following ASN.1 coded data type "NTSNonce" is needed for other types used below for NTS messages. It specifies a 128 bit nonce as required in several message types:

```
NTSNonce ::= OCTET STRING (SIZE(16))
```

4.2. Unicast Messages

4.2.1. Association Messages

4.2.1.1. Message Type: "client_assoc"

This message is structured according to the NTS-Plain archetype. It is realized as an NTP packet with an extension field which holds all the data relevant for NTS. Explicitly, the extension field contains an ASN.1 object of type "ClientAssocData", which is structured as follows:

```
ClientAssocData ::= SEQUENCE {
    clientId         SubjectKeyIdentifier,
    digestAlgos      DigestAlgorithmIdentifiers,
    keyEncAlgos      KeyEncryptionAlgorithms,
    contentEncAlgos  ContentEncryptionAlgorithms
}
```

4.2.1.2. Message Type: "server_assoc"

This message is structured according to the NTS-Signed archetype. The NTS message object in this case is an ASN.1 object of type "ServerAssocData", which is structured as follows:

```
ServerAssocData ::= SEQUENCE {
    clientId              SubjectKeyIdentifier,
    choiceDigestAlgo      DigestAlgorithmIdentifier,
    choiceKeyEncAlgo      KeyEncryptionAlgorithmIdentifier,
    choiceContentEncAlgo  ContentEncryptionAlgorithmIdentifier
}
```

4.2.2. Cookie Messages

4.2.2.1. Message Type: "client_cook"

This message is structured according to the NTS-Certified archetype. The NTS message object is a "ClientCookieData" type ASN.1 object, structured as follows:

```
ClientCookieData ::= SEQUENCE {
    nonce       NTSNonce,
    signAlgo    SignatureAlgorithmIdentifier,
    digestAlgo  DigestAlgorithmIdentifier,
    encAlgo     ContentEncryptionAlgorithmIdentifier,
    keyEncAlgo  KeyEncryptionAlgorithmIdentifier
}
```

It is identified by the following object identifier (fictional values):

```
id-clientCookieData OBJECT IDENTIFIER ::=
    {nts(??) cookie(3) clientcookiedata(1)}
```

4.2.2.2. Message Type: "server_cook"

This message is structured according to the "NTS-Signed-and-Encrypted" archetype. The NTS message object is a "ServerCookieData" object, specified as:

```
ServerCookieData ::= SEQUENCE {
    nonce     NTSNonce,
    cookie    OCTET STRING (SIZE(16))
}
```

It is identified by the following object identifier (fictional values):

```
id-serverCookieData OBJECT IDENTIFIER ::=
    {nts(??) cookie(3) servercookiedata(2)}
```

4.2.3. Time Synchronization Messages

4.2.3.1. Message Type: "time_request"

This message is structured according to the "NTS-Plain" archetype. It is realized as an NTP packet which actually contains regular NTP time synchronization data, as an unsecured NTP packet from a client to a server would. Furthermore, the packet has an extension field which contains an ASN.1 object of type "TimeRequestSecurityData", whose structure is as follows:

```
TimeRequestSecurityData ::=
SEQUENCE {
    nonce_t           NTSNonce,
    digestAlgo        DigestAlgorithmIdentifier,
    hashOfClientCert  BIT STRING
}
```

4.2.3.2. Message Type: "time_response"

This message is also structured according to "NTS-Plain". It is realized as an NTP packet which, like "time_request", contains regular NTP time synchronization data, as an unsecured NTP packet from a server back to a client would. The packet also has an extension field which contains an ASN.1 object of type "TimeResponseSecurityData", with the following structure:

```
TimeResponseSecurityData ::=
SEQUENCE {
    nonce_t  NTSNonce,
}
```

Finally, this NTP packet has a MAC field which contains a Message Authentication Code generated over the whole packet (including the extension field).

4.3. Broadcast Messages

4.3.1. Broadcast Parameter Messages

4.3.1.1. Message Type: "client_bpar"

This first broadcast message is structured according to the NTS-Plain archetype. It is realized as an NTP packet which is empty except for an extension field which contains ans ASN.1 object of type "BroadcastParameterRequest", which is structured as follows:

```
BroadcastParameterRequest ::=
SEQUENCE {
    clientId  SubjectKeyIdentifier
}
```

4.3.1.2. Message Type: "server_bpar"

This message is structured according to "NTS-Signed". It is realized as an NTP packet whose extension field carries the necessary CMS structure. The NTS message object in this case is an ASN.1 object of type "BroadcastParameterResponse", with the following structure:

```
BroadcastParameterRequest ::=
SEQUENCE {
    oneWayAlgo1         DigestAlgorithmIdentifier,
    oneWayAlgo2         DigestAlgorithmIdentifier,
    lastKey             OCTET STRING (SIZE (16)),
    intervalDuration    BIT STRING,
    disclosureDelay     INTEGER,
    nextIntervalTime    BIT STRING,
    nextIntervalIndex   INTEGER
}
```

4.3.2. Broadcast Time Synchronization Message

4.3.2.1. Message Type: "server_broad"

This message is structured according to the "NTS-Plain" archetype. Its realization works via an NTP packet which carries regular NTP broadcast time data as well as an extension field, which contains an ASN.1 object of type "BroadcastTime". It has the following structure:

```
BroadcastTime ::=
SEQUENCE {
    thisIntervalIndex   INTEGER,
    disclosedKey        OCTET STRING (SIZE (16)),
}
```

In addition, this packet has a MAC field which contains a Message Authentication Code generated over the whole packet (including the extension field).

4.3.3. Broadcast Keycheck

4.3.3.1. Message Type: "client_keycheck"

This message is structured according to the "NTS-Plain" archetype. It is realized as an NTP packet with an extension field, which contains an ASN.1 object of type "ClientKeyCheckSecurityData", whose structure is as follows:

```
ClientKeyCheckSecurityData ::=
SEQUENCE {
    nonce_k           NTSNonce,
    interval_number   INTEGER,
    digestAlgo        DigestAlgorithmIdentifier,
    hashOfClientCert  BIT STRING
}
```

4.3.3.2. Message Type: "server_keycheck"

This message is also structured according to "NTS-Plain". It is also realized as an NTP packet with an extension field, which contains an ASN.1 object of type "ServerKeyCheckSecurityData", with the following structure:

```
ServerKeyCheckSecurityData ::=
SEQUENCE {
    nonce_t           NTSNonce,
    interval_number   INTEGER
}
```

Additionally, this NTP packet has a MAC field which contains a Message Authentication Code generated over the whole packet (including the extension field).

5. IANA Considerations

IANA needs to assign an object identifier for id-kp-NTSserver key purpose and another one for the ASN.1 module in the appendix.

6. Security Considerations

To be written.

7. References

7.1. Normative References

[ASN1] International Telecommunication Union, "Abstract Syntax Notation One (ASN.1): Specification of basic notation", ITU-T Recommendation X.680, November 2008.

[IEEE1588] IEEE Instrumentation and Measurement Society. TC-9 Sensor Technology, "IEEE standard for a precision clock synchronization protocol for networked measurement and control systems", 2008.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, September 2009.

[RFC5905] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.

7.2. Informative References

[I-D.ietf-ntp-network-time-security] Sibold, D., Roettger, S., and K. Teichel, "Network Time Security", draft-ietf-ntp-network-time-security-04 (work in progress), July 2014.

Appendix A. ASN.1 Module

The ASN.1 module contained in this appendix defines the id-kp-NTSserver object identifier.

```
      NTSserverKeyPurpose
        { TBD }

      DEFINITIONS IMPLICIT TAGS ::=
      BEGIN

      id-kp-NTSserver OBJECT IDENTIFIER ::= { TBD }

      END
```

Authors' Addresses

Dieter Sibold
Physikalisch-Technische Bundesanstalt
Bundesallee 100
Braunschweig D-38116
Germany

Phone: +49-(0)531-592-8420
Fax: +49-531-592-698420
Email: dieter.sibold@ptb.de

Stephen Roettger
Google Inc

Email: stephen.roettger@googlemail.com

Kristof Teichel
Physikalisch-Technische Bundesanstalt
Bundesallee 100
Braunschweig D-38116
Germany

Phone: +49-(0)531-592-8421
Email: kristof.teichel@ptb.de

Russ Housley
Vigil Security

NTP Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 26, 2015

D. Sibold
PTB
S. Roettger
Google Inc
K. Teichel
PTB

October 23, 2014

Network Time Security
draft-ietf-ntp-network-time-security-05.txt

Abstract

This document describes the Network Time Security (NTS) protocol that enables secure time synchronization with time servers using Network Time Protocol (NTP) or Precision Time Protocol (PTP). Its design considers the special requirements of precise timekeeping, which are described in Security Requirements of Time Protocols in Packet Switched Networks [RFC7384].

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 26, 2015.

Table of Contents

1. Introduction

Time synchronization protocols are increasingly utilized to synchronize clocks in networked infrastructures. The reliable performance of such infrastructures can be degraded seriously by successful attacks against the time synchronization protocol. Therefore, time synchronization protocols have to be secured if they are applied in environments that are prone to malicious attacks. This can be accomplished by utilization of external security protocols like IPsec or by intrinsic security measures of the time synchronization protocol.

The two most popular time synchronization protocols, the Network Time Protocol (NTP) [RFC5905] and the Precision Time Protocol (PTP) [IEEE1588], currently do not provide adequate intrinsic security precautions. This document specifies security measures for NTP and PTP which enable these protocols to verify authenticity of the time server and integrity of the time synchronization protocol packets.

The protocol is specified with the prerequisite in mind that precise timekeeping can only be accomplished with stateless time synchronization communication, which excludes the utilization of standard security protocols like IPsec or TLS for time synchronization messages. This prerequisite corresponds with the requirement that a security mechanism for timekeeping must be designed in such a way that it does not degrade the quality of the time transfer [RFC7384].

Note:

The intent is to formulate the protocol to be applicable to NTP and also PTP. In the current state the specification focuses on the application to NTP.

2. Security Threats

A profound analysis of security threats and requirements for NTP and PTP can be found in the "Security Requirements of Time Protocols in Packet Switched Networks" [RFC7384].

3. Objectives

The objectives of the NTS specification are as follows:

- Authenticity: NTS enables the client to authenticate its time servers.
- Integrity: NTS protects the integrity of time synchronization protocol packets via a message authentication code (MAC).
- Confidentiality: NTS does not provide confidentiality protection of the time synchronization packets.
- Modes of operation: All operational modes of NTP are supported.
- Operational modes of PTP should be supported as far as possible.
- Hybrid mode: Both secure and insecure communication modes are possible for NTP servers and clients, respectively.
- Compatibility:
  - Unsecured NTP associations shall not be affected.
  - An NTP server that does not support NTS shall not be affected by NTS authentication requests.

4. Terms and Abbreviations

MITM Man In The Middle

NTP Network Time Protocol [RFC5905]

NTS Network Time Security

PTP Precision Time Protocol [IEEE1588]

TESLA Timed Efficient Stream Loss-Tolerant Authentication

5. NTS Overview

5.1. Symmetric and Client/Server Mode

NTS applies X.509 certificates to verify the authenticity of the time server and to exchange a symmetric key, the so-called cookie. This cookie is then used to protect authenticity and integrity of the subsequent time synchronization packets by means of a Message Authentication Code (MAC), which is attached to each time synchronization packet. The calculation of the MAC includes the whole time synchronization packet and the cookie which is shared between client and server. The cookie is calculated according to:

cookie = MSB_128 (HMAC(server seed, H(certificate of client))),

with the server seed as key, where H is a hash function, and where the function MSB_128 cuts off the 128 most significant bits of the result of the HMAC function. The server seed is a 128 bit random value of the server, which has to be kept secret. The cookie never changes as long as the server seed stays the same, but the server seed has to be refreshed periodically in order to provide key freshness as required in [RFC7384]. See Section 8 for details on the seed refresh and Section 7.1.1 for the client’s reaction to it.

The server does not keep a state of the client. Therefore it has to recalculate the cookie each time it receives a request from the client. To this end, the client has to attach the hash value of its certificate to each request (see Section 6.3).

5.2. Broadcast Mode

Just as in the case of the client server mode and symmetric mode, authenticity and integrity of the NTP packets are ensured by a MAC, which is attached to the NTP packet by the sender. Verification of the packets’ authenticity is based on the TESLA protocol, in particular on its "not re-using keys" scheme, see section 3.7.2 of

[RFC4082]. TESLA uses a one-way chain of keys, where each key is the output of a one-way function applied to the previous key in the chain. The last element of the chain is shared securely with all clients. The server splits time into intervals of uniform duration and assigns each key to an interval in reverse order, starting with the penultimate. At each time interval, the server sends an NTP broadcast packet appended by a MAC, calculated using the corresponding key, and the key of the previous disclosure interval. The client verifies the MAC by buffering the packet until the disclosure of the key in its associated disclosure interval. In order to be able to verify the validity of the key, the client has to be loosely time synchronized to the server. This has to be accomplished during the initial client server exchange between broadcast client and server. In addition, NTS uses another, more rigorous check to what is used in the TESLA protocol. For a more detailed description of how NTS employs and customizes TESLA, see Appendix C.

6. Protocol Messages

This section describes the types of messages needed for secure time synchronization with NTS.

For some guidance on how these message types can be realized in practice, for use with existing time synchronization protocols, see [I-D.ietf-ntp-cms-for-nts-messages], a companion document for NTS. Said document describes ASN.1 encodings for those message parts that have to be added to a time synchronization protocol for security reasons as well as CMS (Cryptographic Message Syntax, see [RFC5652]) conventions that can be used to get the cryptographic aspects right.

Note that currently, the companion document describes realizations of NTS messages only for utilization with NTP, in which the NTS specific data are enclosed in extension fields on top of NTP packets. A specification of NTS messages for PTP will have to be developed accordingly.

The steps described in Section 6.1 - Section 6.3 belong to the unicast mode, while Section 6.4 and Section 6.5 explain the steps involved in the broadcast mode of NTS.

6.1. Association Messages

In this message exchange, the hash and encryption algorithms that are used throughout the protocol are negotiated. Also, the client receives the certification chain up to a trusted anchor. With the established certification chain the client is able to verify the

server’s signatures and, hence, authenticity of future NTS messages from the server is ensured.

6.1.1. Message Type: "client_assoc"

The protocol sequence starts with the client sending an association message, called client_assoc. This message contains

- the NTS message ID "client_assoc",
- the version number of NTS that the client wants to use (this SHOULD be the highest version number that it supports),
- the hostname of the client,
- a selection of accepted hash algorithms, and
- a selection of accepted encryption algorithms.

6.1.2. Message Type: "server_assoc"

This message is sent by the server upon receipt of client_assoc. It contains

- the NTS message ID "server_assoc",
- the version number used for the rest of the protocol (which SHOULD be determined as the minimum over the client’s suggestion in the client_assoc message and the highest supported by the server),
- the hostname of the server, and
- the server’s choice of algorithm for encryption and for cryptographic hashing, all of which MUST be chosen from the client’s proposals.
- a signature, calculated over the data listed above, with the server’s private key and according to the signature algorithm which is also used for the certificates which are included (see below),
- a chain of certificates, which starts at the server and goes up to a trusted authority, and each certificate MUST be certified by the one directly following it.

6.2. Cookie Messages

During this message exchange, the server transmits a secret cookie to the client securely. The cookie will be used for integrity protection during unicast time synchronization.

6.2.1. Message Type: "client_cook"

This message is sent by the client, upon successful authentication of the server. In this message, the client requests a cookie from the server. The message contains

- the NTS message ID "client_cook",
- the negotiated version number,
- the negotiated signature algorithm,
- the negotiated encryption algorithm,
- a 128-bit nonce,
- the negotiated hash algorithm H,
- the client’s certificate.

6.2.2. Message Type: "server_cook"

This message is sent by the server, upon receipt of a client_cook message. The server generates the hash of the client’s certificate, as conveyed during client_cook, in order to calculate the cookie according to Section 5.1. This message contains

- the NTS message ID "server_cook"
- the version number as transmitted in client_cook,
- a concatenated datum, which is encrypted with the client’s public key, according to the encryption algorithm transmitted in the client_cook message. The concatenated datum contains
  - the nonce transmitted in client_cook, and
  - the cookie.
- a signature, created with the server’s private key, calculated over all of the data listed above. This signature MUST be

calculated according to the transmitted signature algorithm from the client_cook message.

6.3. Unicast Time Synchronisation Messages

In this message exchange, the usual time synchronization process is executed, with the addition of integrity protection for all messages that the server sends. This message can be repeatedly exchanged as often as the client desires and as long as the integrity of the server's time responses is verified successfully.

6.3.1. Message Type: "time_request"

This message is sent by the client when it requests time exchange. It contains

- the NTS message ID "time_request",
- the negotiated version number,
- a 128-bit nonce,
- the negotiated hash algorithm H,
- the hash of the client's certificate under H.

6.3.2. Message Type: "time_response"

This message is sent by the server, after it received a time_request message. Prior to this the server MUST recalculate the client's cookie by using the hash of the client's certificate and the transmitted hash algorithm. The message contains

- the NTS message ID "time_response",
- the version number as transmitted in time_request,
- the server's time synchronization response data,
- the 128-bit nonce transmitted in time_request,
- a MAC (generated with the cookie as key) for verification of all of the above data.

6.4. Broadcast Parameter Messages

In this message exchange, the client receives the necessary information to execute the TESLA protocol in a secured broadcast association. The client can only initiate a secure broadcast association after a successful unicast run, see Section 7.1.2.

See Appendix C for more details on TESLA.

6.4.1. Message Type: "client_bpar"

This message is sent by the client in order to establish a secured time broadcast association with the server. It contains

- the NTS message ID "client_bpar",
- the version number negotiated during association in unicast mode,
- the client's hostname, and
- the signature algorithm negotiated during unicast.

6.4.2. Message Type: "server_bpar"

This message is sent by the server upon receipt of a client_bpar message during the broadcast loop of the server. It contains

- the NTS message ID "server_bpar",
- the version number as transmitted in the client_bpar message,
- the one-way functions used for building the key chain, and
- the disclosure schedule of the keys. This contains:
  - the last key of the key chain,
  - time interval duration,
  - the disclosure delay (number of intervals between use and disclosure of a key),
  - the time at which the next time interval will start, and
  - the next interval's associated index.
- The message also contains a signature signed by the server with its private key, verifying all the data listed above.

6.5. Broadcast Messages

Via this message, the server keeps sending broadcast time synchronization messages to all participating clients.

6.5.1. Message Type: "server_broad"

This message is sent by the server over the course of its broadcast schedule. It is part of any broadcast association. It contains

- the NTS message ID "server_broad",
- the version number that the server’s broadcast mode is working under,
- time broadcast data,
- the index that belongs to the current interval (and therefore identifies the current, yet undisclosed key),
- the disclosed key of the previous disclosure interval (current time interval minus disclosure delay),
- a MAC, calculated with the key for the current time interval, verifying
  - the message ID,
  - the version number, and
  - the time data.

6.6. Broadcast Key Check

This message exchange is performed for an additional check of packet timeliness in the course of the TESLA scheme, see Appendix C.

6.6.1. Message Type: "client_keycheck"

A message of this type is sent by the client in order to initiate an additional check of packet timeliness for the TESLA scheme. It contains

- the NTS message ID "client_keycheck",
- the version number chosen for the broadcast,
- a 128-bit nonce,

- an interval number from the TESLA disclosure schedule,
- the hash algorithm H negotiated in unicast mode, and
- the hash of the client's certificate under H.

6.6.2. Message Type: "server_keycheck"

A message of this type is sent by the server upon receipt of a client_keycheck message during the broadcast loop of the server. Prior to this the server MUST recalculate the client's cookie by using the hash of the client's certificate and the transmitted hash algorithm. It contains

- the NTS message ID "server_keycheck"
- the version number that the server's broadcast mode is working under,
- the 128-bit nonce transmitted in the client_keycheck message,
- the interval number transmitted in the client_keycheck message, and
- a MAC (generated with the cookie as key) for verification of all of the above data.

7. Protocol Sequence

7.1. The Client

7.1.1. The Client in Unicast Mode

For a unicast run, the client performs the following steps:

1. It sends a client_assoc message to the server. It MUST keep the transmitted values for version number and algorithms available for later checks.
2. It waits for a reply in the form of a server_assoc message. After receipt of the message it performs the following checks:
   * The client checks that the message contains a conform version number.
   * It also verifies that the server has chosen the encryption and hash algorithms from its proposal sent in the client_assoc message.

* Furthermore, it performs authenticity checks on the certificate chain and the signature for the version number.

If one of the checks fails, the client MUST abort the run.
Discussion:

Note that by performing the above message exchange and checks, the client validates the authenticity of its immediate NTP server only. It does not recursively validate the authenticity of each NTP server on the time synchronization chain. Recursive authentication (and authorization) as formulated in [RFC7384] depends on the chosen trust anchor.

3. Next, it sends a client_cook message to the server. The client MUST save the included nonce until the reply has been processed.

4. It awaits a reply in the form of a server_cook message; upon receipt it executes the following actions:

   * It verifies that the received version number matches the one negotiated before.

   * It verifies the signature using the server’s public key. The signature has to authenticate the encrypted data.

   * It decrypts the encrypted data with its own private key.

   * It checks that the decrypted message is of the expected format: the concatenation of a 128 bit nonce and a 128 bit cookie.

   * It verifies that the received nonce matches the nonce sent in the client_cook message.

   If one of those checks fails, the client MUST abort the run.

5. The client sends a time_request message to the server. The client MUST save the included nonce and the transmit_timestamp (from the time synchronization data) as a correlated pair for later verification steps.

6. It awaits a reply in the form of a time_response message. Upon receipt, it checks:

   * that the transmitted version number matches the one negotiated before,

* that the transmitted nonce belongs to a previous time_request message,

* that the transmit_timestamp in that time_request message matches the corresponding time stamp from the synchronization data received in the time_response, and

* that the appended MAC verifies the received synchronization data, version number and nonce.

If at least one of the first three checks fails (i.e. if the version number does not match, if the client has never used the nonce transmitted in the time_response message or if it has used the nonce with initial time synchronization data different from that in the response), then the client MUST ignore this time_response message. If the MAC is invalid, the client MUST do one of the following: abort the run or go back to step 5 (because the cookie might have changed due to a server seed refresh). If both checks are successful, the client SHOULD continue time synchronization by going back to step 7.

The client's behavior in unicast mode is also expressed in Figure 1.

7.1.2. The Client in Broadcast Mode

To establish a secure broadcast association with a broadcast server, the client MUST initially authenticate the broadcast server and securely synchronize its time to it up to an upper bound for its time offset in unicast mode. After that, the client performs the following steps:

1. It sends a client_bpar message to the server. It MUST remember the transmitted values for version number and signature algorithm.

2. It waits for a reply in the form of a server_bpar message after which it performs the following checks:

   * The message must contain all the necessary information for the TESLA protocol, as listed in Section 6.4.2.

   * Verification of the message's signature.

   If any information is missing or the server's signature cannot be verified, the client MUST abort the broadcast run. If all checks are successful, the client MUST remember all the broadcast parameters received for later checks.

3. The client awaits time synchronization data in the form of a server_broadcast message. Upon receipt, it performs the following checks:

    1. Proof that the MAC is based on a key that is not yet disclosed (packet timeliness). This is achieved via a combination of checks. First the disclosure schedule is used, which requires the loose time synchronization. If this is successful, the client gets a stronger guarantee via a key check exchange: it sends a client_keycheck message and waits for the appropriate response. Note that it needs to memorize the nonce and the time interval number that it sends as a correlated pair. For more detail on both of the mentioned timeliness checks, see Appendix Appendix C.4. If its timeliness is verified, the packet will be buffered for later authentication. Otherwise, the client MUST discard it. Note that the time information included in the packet will not be used for synchronization until its authenticity could also be verified.

    2. The client checks that it does not already know the disclosed key. Otherwise, the client SHOULD discard the packet to avoid a buffer overrun. If verified, the client ensures that the disclosed key belongs to the one-way key chain by applying the one-way function until equality with a previous disclosed key is shown. If falsified, the client MUST discard the packet.

    3. If the disclosed key is legitimate, then the client verifies the authenticity of any packet that it received during the corresponding time interval. If authenticity of a packet is verified it is released from the buffer and the packet's time information can be utilized. If the verification fails, then authenticity is no longer given. In this case the client MUST request authentic time from the server by means of a unicast time request message.

    See RFC 4082[RFC4082] for a detailed description of the packet verification process.

The client MUST restart the broadcast sequence with a client_bpar message Section 6.4.1 if the one-way key chain expires.

The client's behavior in broadcast mode can also be seen in Figure 2.

7.2. The Server

7.2.1. The Server in Unicast Mode

To support unicast mode, the server MUST be ready to perform the following actions:

- Upon receipt of a client_assoc message, the server constructs and sends a reply in the form of a server_assoc message as described in Section 6.1.2.

- Upon receipt of a client_cook message, the server checks whether it supports the given cryptographic algorithms. It then calculates the cookie according to the formula given in Section 5.1. With this, it MUST construct a server_cook message as described in Section 6.2.2.

- Upon receipt of a time_request message, the server re-calculates the cookie, then computes the necessary time synchronization data and constructs a time_response message as given in Section 6.3.2.

The server MUST refresh its server seed periodically (see Section 8.1).

7.2.2. The Server in Broadcast Mode

A broadcast server MUST also support unicast mode, in order to provide the initial time synchronization which is a precondition for any broadcast association. To support NTS broadcast, the server MUST additionally be ready to perform the following actions:

- Upon receipt of a client_bpar message, the server constructs and sends a server_bpar message as described in Section 6.4.2.

- Upon receipt of a client_keycheck message, the server looks up if it has already disclosed the key associated with the interval number transmitted in that message. If it has not disclosed it, it constructs and sends the appropriate server_keycheck message as described in Section 6.6.2. For more detail, see also Appendix C.

- The server follows the TESLA protocol in all other aspects, by regularly sending server_broad messages as described in Section 6.5.1, adhering to its own disclosure schedule.

It is also the server’s responsibility to watch for the expiration date of the one-way key chain and generate a new key chain accordingly.

8. Server Seed Considerations

The server has to calculate a random seed which has to be kept secret. The server MUST generate a seed for each supported hash algorithm, see Section 9.1.

8.1. Server Seed Refresh

According to the requirements in [RFC7384] the server MUST refresh each server seed periodically. As a consequence, the cookie memorized by the client becomes obsolete. In this case the client cannot verify the MAC attached to subsequent time response messages and has to respond accordingly by re-initiating the protocol with a cookie request (Section 6.2).

8.2. Server Seed Algorithm

8.3. Server Seed Lifetime

9. Hash Algorithms and MAC Generation

9.1. Hash Algorithms

Hash algorithms are used at different points: calculation of the cookie and the MAC, and hashing of the client's certificate. Client and server negotiate a hash algorithm H during the association message exchange (Section 6.1) at the beginning of a unicast run. The selected algorithm H is used for all hashing processes in that run.

In broadcast mode, hash algorithms are used as pseudo random functions to construct the one-way key chain. Here, the utilized hash algorithm is communicated by the server and non-negotiable.

The list of the hash algorithms supported by the server has to fulfill the following requirements:

- it MUST NOT include SHA-1 or weaker algorithms,
- it MUST include SHA-256 or stronger algorithms.

Note

Any hash algorithm is prone to be compromised in the future. A successful attack on a hash algorithm would enable any NTS client to derive the server seed from their own cookie. Therefore, the server MUST have separate seed values for its different supported hash algorithms. This way, knowledge gained from an attack on a

hash algorithm H can at least only be used to compromise such clients who use hash algorithm H as well.

9.2. MAC Calculation

For the calculation of the MAC, client and server are using a Keyed-Hash Message Authentication Code (HMAC) approach [RFC2104]. The HMAC is generated with the hash algorithm specified by the client (see Section 9.1).

10. IANA Considerations

11. Security Considerations

11.1. Initial Verification of the Server Certificates

The client has to verify the validity of the certificates during the certification message exchange (Section 6.1.2). Since it generally has no reliable time during this initial communication phase, it is impossible to verify the period of validity of the certificates. Therefore, the client MUST use one of the following approaches:

- The validity of the certificates is preconditioned. Usually this will be the case in corporate networks.
- The client ensures that the certificates are not revoked. To this end, the client uses the Online Certificate Status Protocol (OCSP) defined in [RFC6277].
- The client requests a different service to get an initial time stamp in order to be able to verify the certificates' periods of validity. To this end, it can, e.g., use a secure shell connection to a reliable host. Another alternative is to request a time stamp from a Time Stamping Authority (TSA) by means of the Time-Stamp Protocol (TSP) defined in [RFC3161].

11.2. Revocation of Server Certificates

According to Section 8.1, it is the client's responsibility to initiate a new association with the server after the server's certificate expires. To this end the client reads the expiration date of the certificate during the certificate message exchange (Section 6.1.2). Besides, certificates may also be revoked prior to the normal expiration date. To increase security the client MAY verify the state of the server's certificate via OCSP periodically.

11.3. Usage of NTP Pools

The certification based authentication scheme described in Section 6 is not applicable to the concept of NTP pools. Therefore, NTS is not able to provide secure usage of NTP pools.

11.4. Denial-of-Service in Broadcast Mode

TESLA authentication buffers packets for delayed authentication. This makes the protocol vulnerable to flooding attacks, causing the client to buffer excessive numbers of packets. To add stronger DoS protection to the protocol, client and server use the "not re-using keys" scheme of TESLA as pointed out in section 3.7.2 of RFC 4082 [RFC4082]. In this scheme the server never uses a key for the MAC generation more than once. Therefore the client can discard any packet that contains a disclosed key it knows already, thus preventing memory flooding attacks.

Note that an alternative approach to enhance TESLA’s resistance against DoS attacks involves the addition of a group MAC to each packet. This requires the exchange of an additional shared key common to the whole group. This adds additional complexity to the protocol and hence is currently not considered in this document.

11.5. Delay Attack

In a packet delay attack, an adversary with the ability to act as a MITM delays time synchronization packets between client and server asymmetrically [RFC7384]. This prevents the client to measure the network delay, and hence its time offset to the server, accurately [Mizrahi]. The delay attack does not modify the content of the exchanged synchronization packets. Therefore cryptographic means do not provide a feasible way to mitigate this attack. However, several non-cryptographic precautions can be taken in order to detect this attack.

1. Usage of multiple time servers: this enables the client to detect the attack provided that the adversary is unable to delay the synchronizations packets between the majority of servers. This approach is commonly used in NTP to exclude incorrect time servers [RFC5905].

2. Multiple communication paths: The client and server are utilizing different paths for packet exchange as described in the I-D [I-D.shpiner-multi-path-synchronization]. The client can detect the attack provided that the adversary is unable to manipulate the majority of the available paths [Shpiner]. Note that this approach is not yet available, neither for NTP nor for PTP.

3. Usage of an encrypted connection: the client exchanges all packets with the time server over an encrypted connection (e.g. IPsec). This measure does not mitigate the delay attack but it makes it more difficult for the adversary to identify the time synchronization packets.

4. For the unicast mode: Introduction of a threshold value for the delay time of the synchronization packets. The client can discard a time server if the packet delay time of this time server is larger than the threshold value.

Additional provision against delay attacks has to be taken in the broadcast mode. This mode relies on the TESLA scheme which is based on the requirement that a client and the broadcast server are loosely time synchronized. Therefore, a broadcast client has to establish time synchronization with its broadcast server before it maintains time synchronization by utilization of the broadcast mode. To this end it initially establishes a unicast association with its broadcast server until time synchronization and calibration of the packet delay time is achieved. After that it establishes a broadcast association to the broadcast server and utilizes TESLA to verify integrity and authenticity of any received broadcast packets.

An adversary who is able to delay broadcast packets can cause a time adjustment at the receiving broadcast clients. If the adversary delays broadcast packets continuously, then the time adjustment will accumulate until the loose time synchronization requirement is violated, which breaks the TESLA scheme. To mitigate this vulnerability the security condition in TESLA has to be supplemented by an additional check in which the client, upon receipt of a broadcast message, verifies the status of the corresponding key via a unicast message exchange with the broadcast server (see section Appendix C.4 for a detailed description of this check). Note, that a broadcast client should also apply the above mentioned precautions as far as possible.

12. Acknowledgements

The authors would like to thank Russ Housley, Steven Bellovin, David Mills and Kurt Roeckx for discussions and comments on the design of NTS. Also, thanks to Harlan Stenn for his technical review and specific text contributions to this document.

13. References

13.1. Normative References

[IEEE1588] IEEE Instrumentation and Measurement Society. TC-9 Sensor Technology, "IEEE standard for a precision clock synchronization protocol for networked measurement and control systems", 2008.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3161] Adams, C., Cain, P., Pinkas, D., and R. Zuccherato, "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)", RFC 3161, August 2001.

[RFC4082] Perrig, A., Song, D., Canetti, R., Tygar, J., and B. Briscoe, "Timed Efficient Stream Loss-Tolerant Authentication (TESLA): Multicast Source Authentication Transform Introduction", RFC 4082, June 2005.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, September 2009.

[RFC5905] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.

[RFC6277] Santesson, S. and P. Hallam-Baker, "Online Certificate Status Protocol Algorithm Agility", RFC 6277, June 2011.

13.2. Informative References

[I-D.shpiner-multi-path-synchronization] Shpiner, A., Tse, R., Schelp, C., and T. Mizrahi, "Multi-Path Time Synchronization", draft-shpiner-multi-path-synchronization-03 (work in progress), February 2014.

[Mizrahi] Mizrahi, T., "A game theoretic analysis of delay attacks against time synchronization protocols", in Proceedings of Precision Clock Synchronization for Measurement Control and Communication, ISPCS 2012, pp. 1-6, September 2012.

[RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

[RFC7384] Mizrahi, T., "Security Requirements of Time Protocols in Packet Switched Networks", RFC 7384, October 2014.

[Shpiner] Shpiner, A., Revah, Y., and T. Mizrahi, "Multi-path Time Protocols", in Proceedings of Precision Clock Synchronization for Measurement Control and Communication, ISPCS 2013, pp. 1-6, September 2013.

Appendix A. Flow Diagrams of Client Behaviour

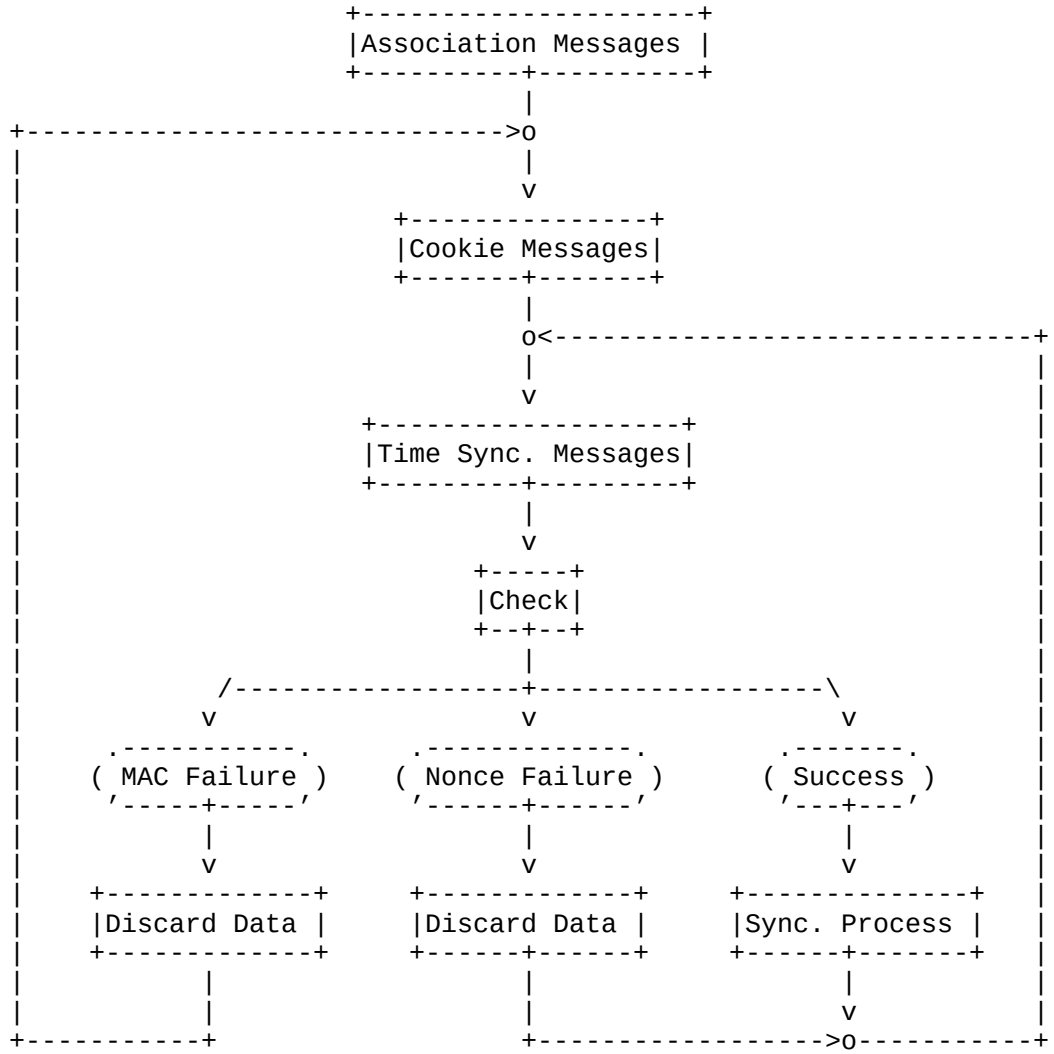


Figure 1: The client's behavior in NTS unicast mode.

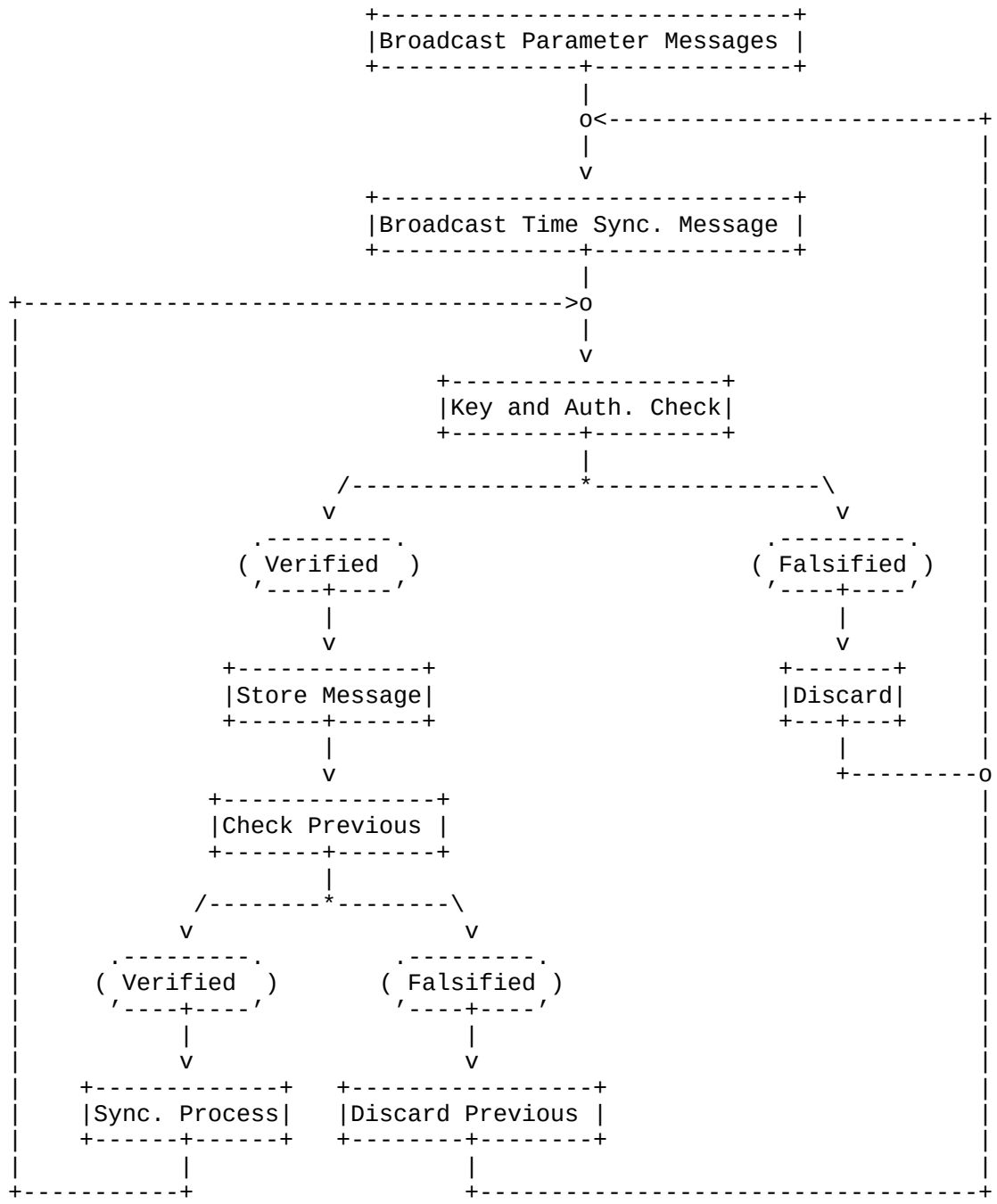


Figure 2: The client’s behaviour in NTS broadcast mode.

Appendix B. TICTOC Security Requirements

The following table compares the NTS specifications against the TICTOC security requirements [RFC7384].

Section	Requirement from I-D tictoc security-requirements-05	Requirement level	NTS
5.1.1	Authentication of Servers	MUST	OK
5.1.1	Authorization of Servers	MUST	OK
5.1.2	Recursive Authentication of Servers (Stratum 1)	MUST	OK
5.1.2	Recursive Authorization of Servers (Stratum 1)	MUST	OK
5.1.3	Authentication and Authorization of Slaves	MAY	-
5.2	Integrity protection.	MUST	OK
5.4	Protection against DoS attacks	SHOULD	OK
5.5	Replay protection	MUST	OK
5.6	Key freshness.	MUST	OK
	Security association.	SHOULD	OK
	Unicast and multicast associations.	SHOULD	OK
5.7	Performance: no degradation in quality of time transfer.	MUST	OK
	Performance: lightweight computation	SHOULD	OK
	Performance: storage, bandwidth	SHOULD	OK
5.7	Confidentiality protection	MAY	NO
5.9	Protection against Packet Delay and Interception Attacks	SHOULD	NA*)

5.10	Secure mode	MUST	-
	Hybrid mode	SHOULD	-

*) See discussion in section Section 11.5.

Comparison of NTS sepecification against TICTOC security requirements.

Appendix C. Broadcast Mode

For the broadcast mode, NTS adopts the TESLA protocol with some customizations. This appendix provides details on the generation and usage of the one-way key chain collected and assembled from [RFC4082]. Note that NTS is using the "not re-using keys" scheme of TESLA as described in section 3.7.2. of [RFC4082].

C.1. Server Preparations

Server setup:

1. The server determines a reasonable upper bound B on the network delay between itself and an arbitrary client, measured in milliseconds.

2. It determines the number n+1 of keys in the one-way key chain. This yields the number n of keys that are usable to authenticate broadcast packets. This number n is therefore also the number of time intervals during which the server can send authenticated broadcast messages before it has to calculate a new key chain.

3. It divides time into n uniform intervals I_1, I_2, ..., I_n. Each of these time intervals has length L, measured in milliseconds. In order to fulfill the requirement 3.7.2. of RFC 4082 the time interval L has to be smaller than the time interval between the broadcast messages.

4. The server generates a random key K_n.

5. Using a one-way function F, the server generates a one-way chain of n+1 keys K_0, K_1, ..., K_{n} according to

   K_i = F(K_{i+1}).

6. Using another one-way function F', it generates a sequence of n+1 MAC keys K'_0, K'_1, ..., K'_{n-1} according to

K’_i = F’(K_i).

7. Each MAC key K’_i is assigned to the time interval I_i.

8. The server determines the key disclosure delay d, which is the number of intervals between using a key and disclosing it. Note that although security is provided for all choices d>0, the choice still makes a difference:

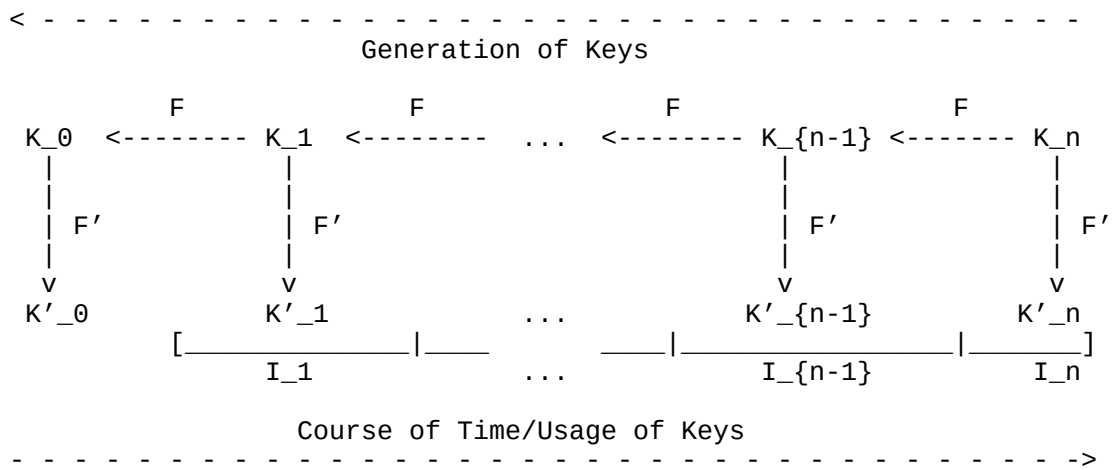
   * If d is chosen too short, the client might discard packets because it fails to verify that the key used for their MAC has not been yet disclosed.

   * If d is chosen too long, the received packets have to be buffered for a unnecessarily long time before they can be verified by the client and subsequently be utilized for time synchronization.

   The server SHOULD calculate d according to

      d = ceil(2*B / L) + 1,

   where ceil gives the smallest integer greater than or equal to its argument.



A Schematic explanation on the TESLA protocol’s one-way key chain

C.2. Client Preparation

A client needs the following information in order to participate in a TESLA broadcast.

- One key K_i from the one-way key chain, which has to be authenticated as belonging to the server. Typically, this will be K_0.
- The disclosure schedule of the keys. This consists of:
  - the length n of the one-way key chain,
  - the length L of the time intervals I_1, I_2, ..., I_n,
  - the starting time T_i of an interval I_i. Typically this is the starting time T_1 of the first interval;
  - the disclosure delay d.
- The one-way function F used to recursively derive the keys in the one-way key chain,
- The second one-way function F' used to derive the MAC keys K'_0, K'_1, ... , K'_n from the keys in the one-way chain.
- An upper bound D_t on how far its own clock is "behind" that of the server.

Note that if D_t is greater than (d - 1) * L, then some authentic packets might be discarded. If D_t is greater than d * L, then all authentic packets will be discarded. In the latter case, the client should not participate in the broadcast, since there will be no benefit in doing so.

C.3. Sending Authenticated Broadcast Packets

During each time interval I_i, the server sends one authenticated broadcast packet P_i. This packet consists of:

- a message M_i,
- the index i (in case a packet arrives late),
- a MAC authenticating the message M_i, with K'_i used as key,
- the key K_{i-d}, which is included for disclosure.

C.4. Authentication of Received Packets

When a client receives a packet P_i as described above, it first checks that it has not received a packet with the same disclosed key before. This is done to avoid replay/flooding attacks. A packet that fails this test is discarded.

Next, the client begins to check the packet's timeliness by ensuring that, according to the disclosure schedule and with respect to the upper bound D_t determined above, the server cannot have disclosed the key K_i yet. Specifically, it needs to check that the server's clock cannot read a time that is in time interval I_{i+d} or later. Since it works under the assumption that the server's clock is not more than D_t "ahead" of the client's clock, the client can calculate an upper bound t_i for the server's clock at the time when P_i arrived. This upper bound t_i is calculated according to

$$t_i = R + D_t,$$

where R is the client's clock at the arrival of P_i. This implies that at the time of arrival of P_i, the server could have been in interval I_x at most, with

$$x = \mathrm{floor}((t_i - T_1) / L) + 1,$$

where floor gives the greatest integer less than or equal to its argument. The client now needs to verify that

$$x < i+d$$

is valid (see also section 3.5 of [RFC4082]). If falsified, it is discarded.

If the check above is successful, the client performs another more rigorous check: it sends a key check request to the server (in the form of a client_keycheck message), asking explicitly if K_i has already been disclosed. It remembers the timestamp t_check of the sending time of that request as well as the nonce it used correlated with the interval number i. If it receives an answer from the server stating that K_i has not yet been disclosed and it is able to verify the HMAC on that response, then it deduces that K_i was undisclosed at t_check and therefore also at R. In this case, the clients accepts P_i as timely.

Next the client verifies that a newly disclosed key K_{i-d} belongs to the one-way key chain. To this end it applies the one-way function F to K_{i-d} until it can verify identity with an earlier disclosed key (see Clause 3.5 in RFC 4082, item 3).

Next the client verifies that the transmitted time value s_i belongs to the time interval I_i, by checking

T_i =< s_i, and

s_i < T_{i+1}.

If falsified, the packet MUST be discarded and the client MUST reinitialize the broadcast mode with a unicast association (because a falsification of this check yields that the packet was not generated according to protocol, which suggests an attack).

If a packet P_i passes all tests listed above, it is stored for later authentication. Also, if at this time there is a package with index i-d already buffered, then the client uses the disclosed key K_{i-d} to derive K'_{i-d} and uses that to check the MAC included in package P_{i-d}. On success, it regards M_{i-d} as authenticated.

Appendix D. Random Number Generation

At various points of the protocol, the generation of random numbers is required. The employed methods of generation need to be cryptographically secure. See [RFC4086] for guidelines concerning this topic.

Authors' Addresses

Dieter Sibold
Physikalisch-Technische Bundesanstalt
Bundesallee 100
Braunschweig D-38116
Germany

Phone: +49-(0)531-592-8420
Fax: +49-531-592-698420
Email: dieter.sibold@ptb.de

Stephen Roettger
Google Inc

Email: stephen.roettger@googlemail.com

Kristof Teichel
Physikalisch-Technische Bundesanstalt
Bundesallee 100
Braunschweig D-38116
Germany

Phone: +49-(0)531-592-8421
Email: kristof.teichel@ptb.de

INTERNET-DRAFT
Intended Status: Informational
Expires: April 30, 2015

H. Song
N. Zong
Huawei

October 27, 2014

A Threat Model for Router Backdoor
draft-song-router-backdoor-00

Abstract

This document elaborates a threat model for inherent backdoor in a telecom router. We assume a malicious router can have inherent backdoor with an interest in eavesdropping or disabling the functioning of the router or the whole network. It is intended to demonstrate to the system designers and network administrators how the backdoor works, so as to assist in the security evaluation of the routers, and especially the standard design that is immune to inherent backdoors.

Table of Contents

1 Introduction

In recent years, telecom routers sometimes might be doubted having backdoors, and the main suspicion is the equipment might be used for eavesdropping, because telecom routers are the key equipments for packet forwarding, it handles huge amount of traffic forwarding all the time. So it might have the opportunity to take its network position advantage to analyze information for unknown purposes. But equipment vendors always claim they have no backdoors. Usually there is no evidence for it, but this kind of distrust among each other harms the industry. This document is going to introduce a threat model of telecom routers in detail.

In one aspect, vendors would like to verify its innocence. Now they usually would like to find a third party organization to evaluate and assign a certificate to authenticate their products. With this authentication, it more or less helps to setup a trust between each other. Sometimes they are required to open their source code to the regulators, but most vendors consider their source code as their business secret, and the key to achieve their business success. So they often would not like the idea of opening source code.

In another aspect, operators/regulators would like to make sure the equipment is secure. But there lack of standard mechanisms to evaluate whether a backdoor exists in the router or not. And of course, the operators would not like to spend a lot of manpower to evaluate the source code of the router either. They usually also trust the evaluation of third party. But now third party can only provide the service to authenticate if the equipment is secure under some common attacks, or if it abides some secure programming rules , there is no way for a third party to guarantee the non-existence of backdoor.

The motivation is to address the aforementioned problem from both sides. One direction is prevention. That means, a well designed standard solution/guidance can be found to prevent/avoid the occurrence of back doors. For example, standard design specifications that can prevent backdoors.

Another potential way might be running time detection. Some designed tools can be running to detect the malicious behavior of the router immediately, just like people run anti-virus software (e.g. 360 or Symantec) in their computers. The potential challenge is that the malicious behavior is unknown. But it is still effective if the detector can detect and then block such malicious behaviors.

The third possible way might be analysis afterwards. But which needs huge storage space (when you consider a router of 40Gbps, if you

store the raw data, then it needs about 18T bytes storage space per hour), and might become useless if the malicious behaviors have already happened. It is helpful in a slow manner for people to adopt measures after some detected accidents.

With the above efforts, there can be three results.

Result 1: No backdoor. Then it can certify the innocence of vendors. The operators and the regulators are also glad to dismiss the suspicion to the vendors.

Result 2: Yes, there is backdoor. Then in the opposite aspect, it helps the administrators to detect it.

Result 3: Still NOT Sure. It has not detected the malicious behavior, but it can mitigate the distrust between each other, because both parties agree on a solution.

The problem space of this document includes the threat models of inherent router backdoors, but leaves the solutions of prevention, detection and afterwards analysis for future study. And anything related to third party implanted backdoors or system vulnerabilities are out of scope, as well as anything related for protection against attacks to the routers.

2 Terminology

Backdoor: A backdoor is a method of bypassing normal authentication, securing unauthorized remote access to a equipment, obtaining access to functioning components or enabling hidden functions, while attempting to remain undetected.

Inherent backdoor: An on-purpose designed backdoor in an equipment when a customer gets the equipment from the provider, and it is not a backdoor implanted by any third party after the customer operates the equipment, implemented by either software or hardware. The assumption is that the software and hardware of the equipment is not changed during the delivery chain.

3 Backdoor Classification

3.1 Implementation classification

From the implementation perspective, we classify the backdoor into hardware and software. For hardware backdoors, they can be specific designed transistor, or shadow circuit. For software backdoors, it

could be hidden software functions triggered by specific designed packets, or hidden ports, for example, the notorious TCP 32764 backdoor.

3.2 Purpose Classification

From the purpose perspective of a backdoor, we can classify the backdoor into classes with following purposes.

One purpose of the backdoor is for traffic eavesdropping, which is mainly suspected in various cases. The traffic eavesdropping can have a definite target (a person, a line or a user account), or can be pervasive.

Another purpose of backdoor might be to make the equipment malfunction. An adversary can get the root control of the router, and can control over time, location, component, and in which behavior to make the router malfunction.

A possible purpose of backdoor could also be for management and operation of the device, for example, for the update of the device. But un-documented method to access the device must also be seen as a backdoor attack.

4 Behaviors of Traffic Eavesdropping

The main suspected behavior is traffic eavesdropping. An easiest way that a spying router can do is to encapsulate the original user packet (no matter targeted or pervasive) and send to another destination for information collection and analysis. The pervasive monitoring cannot be done during the network traffic peak time, as it will produce too much traffic from the device. But the targeted user/line packet replication and monitoring can be done at any time. Note that in this way, there are new eavesdropping packets generated by the router. And the source IP address could be of the router itself or any fake IP address. And the destination of the eavesdropping packet could be a malicious NMS or any other controlled destination. The eavesdropping packets can be encrypted.

Another way for traffic eavesdropping is to use an existing session instead of a new session from the router. A spying router monitors user packets information, and then encapsulates that information to an existing e2e session that was designed for eavesdropping. Please note there is no new packet from the router in this scenario, due to its utilization of an existing session. It is very hard to find it with traffic monitoring in the router interfaces. And of course, the eavesdropping packets can be encrypted. This kind of eavesdropping is

hard to be used for pervasive monitoring due to the capability of a spying session.

A more complicated way for traffic eavesdropping of a router is that the spying router monitors and analyzes user packets, and the extracted information is sent to the adversary when needed, either through router to NMS messages, or a new session/an existing session. In this case, there are no continuous eavesdropping messages. Eavesdropping messages can also be encrypted. But this method requires the malicious router to have a powerful analysis tool for big data, which might be not so easy to hide.

A spying router can also have a backdoor of storage, and provides access to it through manual or remote control access. A spying router can leave illegal root control to the adversary, and the information is only accessed when needed.

The functioning of the eavesdropping function can be triggered by special designed packets or other means.

5 Behavior of Equipment Malfunctioning

A back door can make the router malfunction. With enabling the backdoor in a router located in the key path in a network topology, it can even destroy the functioning of a whole network.

Usually, the adversary gets root control over the router, and then can operate the router as its will. The malfunctioning behaviors include but not limited to: packet dropping, illegal routing table modification, illegal packet modification, or turning off the router.

6 Backdoor of Black Platform

The back door in a router can provide a platform, so that the adversary can implant various other unlawful plug-ins functions secretarially. The platform is just like an engine for any future risks. The malicious plug-in can be installed or uninstalled from the platform freely. The adversary gets broad and extensible control over the router. The adversary can develop new malicious plug-in for new services when needed, or new plug-ins to protect other malicious functions from being detected. It can also uninstall the plug-in from the router after it completes its task so as to avoid detection.

7 Potential Solutions

The main purpose of this document is about the treat model instead of solution guidance. This section generally discusses the direction of solution.

As introduced in section 1, the prevention solution may include: (a) Source code examination (which could be done by using open source code) and (b)Authoritative third party authentication and certification.

And the running time detection may include an anti-virus like backdoor-detection application in the router, or outside of the router but to monitor the traffic in and out of the router, so as to check if there is abnormal traffic patterns. There is also method to trace the code running in the machine, and report any suspicious behaviors.

The afterwards analysis needs big data capability, to gather all related information from the router, including those reported from the router or monitored information from other tools. The big data analysis should take both data plane and control plane in scope.

8 Security Considerations

This document explores the security threats of network forwarding equipments inherent backdoors, It does not provide any detailed specifications on how to avoid or detect such backdoors. But it hopes the standard development organizations can work on the solutions.

9 IANA Considerations

There is no IANA consideration with this specification.

10 Acknowledgements

The authors would like to thank the following people for their support and comments with the discussion of this problem: Stephen Farrell, Melinda Shore, Jari Akro, Dacheng Zhang.

11 References

11.1 Informative References

[RFC4108] Housley, R., "Using Cryptographic Message Syntax (CMS) to Protect Firmware Packages", RFC 4108, August 2005.

[I-D.trammell-perpass-ppa] Trammell, B., Borkmann, D., and C. Huitema, "A Threat Model for Pervasive Passive Surveillance", draft-trammell-perpass-ppa-01, November, 2013.

Authors' Addresses

Haibin Song
Huawei Technologies, Co. Ltd
Nanjing, China

EMail: haibin.song@huawei.com

Ning Zong
Huawei Technologies, Co. Ltd
Nanjing, China

Email: zongning@huawei.com