

A Covert Data Transport Protocol

Yu Fu*, Zhe Jia[†], Lu Yu*, Xingsi Zhong*, and Richard Brooks*

* The Holcombe Department of Electrical and Computer Engineering
 {fu2, lyu, xingsiz, rrb}@g.clemson.edu

[†]Department of Physics and Astronomy
 zhej@g.clemson.edu
 Clemson University, Clemson, SC, 29634, USA

Abstract

Both enterprise and national firewalls filter network connections. For data forensics and botnet removal applications, it is important to establish the information source. In this paper, we describe a data transport layer which allows a client to transfer encrypted data that provides no discernible information regarding the data source. We use a domain generation algorithm (DGA) to encode AES encrypted data into domain names that current tools are unable to reliably differentiate from valid domain names. The domain names are registered using (free) dynamic DNS services. The data transmission format is not vulnerable to Deep Packet Inspection (DPI).

1 Introduction

Protocol obfuscation is widely used for evading censorship and surveillance, and hiding criminal activity. Most firewalls use DPI to analyze network packets and filter out sensitive information. But if the source protocol is obfuscated or transformed into a different protocol, detection techniques that worked well with the source protocol will either detect nothing sensitive, or detect something which is far from the real information. When encrypted connections draw attention or are blocked [1], protocol obfuscation is a solution for covert communication. For example, consider a botnet mothership trying to diffuse instructions to its infected zombies, obfuscating encrypted data streams is very useful.

In general, protocol obfuscation can be divided into two categories: (1) protocol mimicry, and (2) protocol tunnelling. Protocol mimicry is to make protocol A look like protocol B, by tampering some features (packet syntax and

statistical features) of protocol A. Protocol tunnelling is to take the contents of packets from protocol A and put them into the payload of protocol B. In this way, protocol B is a carrier and masquerade of protocol A. As Houmansadr [2] pointed out, most protocol mimicry fails to be completely unobservable even without the attacker resorting to correlating multiple network flows or performing sophisticated traffic analysis. He concluded that mimicking the protocol in its entirety, including its reaction to errors, typical traffic and artifacts, is difficult. Protocol tunnelling, on the other hand, is easy to implement and there are many tools available. However, it is vulnerable to statistical analysis [3, 4].

The covert data transport protocol we present transforms arbitrary network traffic into legitimate DNS traffic. The server encodes the message into a list of domain names and register them to a randomly chosen IP address. The client does a reverse-DNS lookup on the IP address and decodes the domain names to retrieve the message. Different from DNS tunneling, this doesn't use uncommon record types (TXT records) or carry suspiciously large volume of traffic as DNS payloads. On the contrary, the resulting traffic will be normal DNS lookup/reverse-lookup traffic, which will not attract attention. The data transmission is not vulnerable to DPI.

The structure of this paper is as follows. In Section 2, protocol obfuscation techniques are discussed. Section 3 provides the background for this work. Section 4 proposes the transport protocol. Section 5 concludes the paper and suggests future work.

2 Related Work

2.1 Protocol obfuscation techniques

As one of the most famous anonymization tools, the Onion Router (Tor) [5] provides an infrastructure for anonymous communication over a public network. Obfsproxy [6] circumvents censorship by camouflaging Tor traffic between client and bridge nodes. It supports multiple protocols, called pluggable transports, which specify how traffic is transformed. ScrambleSuit [7] is a network protocol to obfuscate the transported application data to defend against active probing and protocol fingerprinting. SkypeMorph [8] is a Tor pluggable transport to reshape Tor packets to resemble Skype calls. StegoTorus [9] first uses chopping to change packet sizes and timing information, and then uses steganography to disguise Tor traffic as a message in an innocuous cover protocol, such as HTTP. Format-Transforming Encryption (FTE) [10] evades regular-based deep packet inspection (DPI) technologies by transforming Tor traffic into a predefined format. Dust [11] provides blocking resistance against the most common packet filtering techniques.

Tor has been used by botnets to help hide their command and control (C&C) nodes (motherships) [12, 13, 14]. In August 2013, Tor had seen a rapid spike of clients, which turned out to be click-fraud botnet running its C&C as a Tor Hidden Service [15, 16].

Other protocol obfuscation techniques including randomized flushing of data streams and random padding [17] are widely used by P2P software, for example, Skype, BitTorrent, and eMule. Decoy routing [18] makes it possible for a client to connect to any unblocked host/service as a middle point and finally connect to a blocked destination without cooperation with the host. Similar to protocol tunneling, Houmansadr [2] suggests running the actual protocol to embed data rather than mimicking the protocol. It is not certain that this would be practical, since the resulting combined protocol would contain side channels that could be modeled using the cross-product of two Markov models [19].

Obfuscation techniques in the last paragraph hide the protocol being used, but do not hide the address of the communication partner. Tor hides the address of the communication partners, and may hide the fact that Tor is being used. Mainly, Tor hides the communication partners by adding two extra network hops and three encryptions. Many countries block Tor used by stopping access to all Tor entrance IP addresses. The approach we present hides the identity of the partner, hides the protocol being used, and produces common DNS traffic.

2.2 DNS-based steganographic channels

Since our paper focuses on transforming arbitrary network traffic into DNS traffic, it is necessary to look at DNS-based steganographic channels. The Feederbot malware [20] implements a transmission scheme by encoding messages into DNS TXT records, with which the bots query DNS servers for C&C commands. PlugX [21], a remote access tool providing remote control and surveillance capabilities, uses DNS as a carrier protocol for C&C communication. Thyer [22] and Altalhi [23] use DNS 16-bit identification (ID) in the IP header as a covert channel to transmit data, which only encodes two characters of message in each packet. Ngadi [24] uses DNS packet length covert channel, which is suitable for short message transfer. Most DNS-based steganographic channels suffer from low throughput or side-channel analysis. For example, DNS TXT-based steganographic channels have different payload length (probably longer) than normal DNS traffic, which are subject to side-channel analysis in packet length. DNS packet length-based steganographic channels have different packet length distribution than normal DNS traffic, which are subject to side-channel analysis in the entropy of the packet length distribution. Since our approach produces real DNS responses, it generates real DNS traffic, which is not subject to side-channel analysis.

3 Background

3.1 Hidden Markov Models (HMMs)

A Markov model is a tuple $G = (S, T, P)$ where S is a set of states of a model, T is a set of directed transitions between the states, and $P = \{p(s_i, s_j)\}$ is a probability matrix associated with transitions from state s_i to s_j such that:

$$\sum_{s_j \in S} p(s_i, s_j) = 1, \forall s_i \in S \quad (1)$$

A Markov model satisfies the Markov property, where the next state only depends on the current state. An HMM is a Markov model with unobservable states. A standard HMM [25, 26] has two sets of random processes: one for state transition and the other for symbol outputs. A deterministic HMM [27, 28, 29] is used in this paper, which only has one random processes for state transitions. Different output symbols are associated with transitions with different probability. This representations is equivalent to the standard HMM [30].

HMMs are widely used for pattern recognition and detection. Chen [28] used HMMs to detect Zeus botnet zombies and achieved 94.7% true positive rate (tpr) and 0.7% false positive rate (fpr). Inter-packet delay of Zeus packets

were collected and represented by an HMM, which was inferred with the zero-knowledge HMM inference algorithm [31]. Zhong [32] conducted a side-channel attack on Phasor Measurement Units (PMUs) in an electric power grid and used HMMs to differentiate data source in an encrypted tunnel.

3.2 Domain Generation Algorithms (DGAs)

Botnets are groups of compromised computers that botmasters use to launch attacks over the Internet. In modern botnets, fast-flux is used to change the mapping between IP addresses and DNS names of the C&C server periodically to avoid detection. DGAs are used to generate large numbers of DNS names as the rendezvous for the covert channel between bots and the botmaster. DGAs are widely used by botnets including Zeus [33], Conficker [34], Kraken[35], Srizbi [36], Torpig [37] etc.

Our previous work developed an HMM-based DGA, which evaded the DGA detection metrics (Kullback-Leibler distance, Edit distance, and Jaccard Index) in the literature [38] and two cutting-edge DGA-detection systems (BotDigger [39] and Pleiades [40]). In our previous HMM-based DGA work, we inferred an HMM that represents the linguistic features of legitimate domain names collected from the entire IPv4 space [41]. Domain names generated by the HMM will be similar to, but not conflicting with legitimate domain names statistically. The inferred HMM will be used in this paper.

3.3 Format-Transforming Encryption (FTE)

FTE [10] extends conventional symmetric encryption by formatting the ciphertext. Arbitrary application-layer network traffic can be transformed into a target protocol using FTE. It is used to evade regular expression based protocol identification in Internet censorship and surveillance, because the original protocols are obfuscated. Zhong [42] improved FTE to mimic arbitrary network protocols without learning their regular expressions. This work is similar to FTE, because we transform data into domain names, which also obfuscates the original protocol with DNS protocol.

4 Proposed Data Transport Protocol

The covert data transport protocol is based on two-way communications between a client and a server. Figure 1 shows that the flow chart of the protocol. Before communication starts, the client and server will share the following information out of band: (1) AES key, (2) pseudo-random number generator (PRG), (3) PRG seed, and (4) an HMM

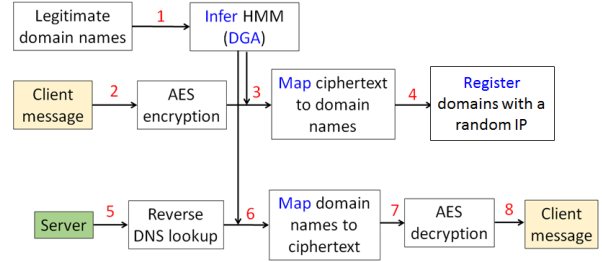


Figure 1: Flow chart of the covert data transport protocol

that represents the statistical model of legitimate IPv4 domain names (step 1) and start state.

The HMM guarantees that domain names generated by the HMM will have the same statistical features as, but not conflict with, legitimate domain names. The steps of communication from the client to the server are:

- The client prepares the message ('client message') to send to the server.
- The message is AES encrypted into ciphertext (step 2).
- The client maps ciphertext to domain names with the HMM-encoding algorithm described below (step 3).
- The client registers the generated domain names with the pseudo-randomly chosen IP address (step 4).
- The server reverse DNS-lookups the chosen IP address and retrieves the domain names (step 5).
- The server maps domain names back to ciphertext (step 6).
- Ciphertext is AES decrypted into client message (step 7 and 8).

4.1 HMM-encoding/decoding algorithm

HMM-encoding maps data strings (ciphertext in our work) to domain names. Since the HMM is a probabilistic regular grammar with transitions associated with different probability, an intuitive idea is to find a path in HMM on one side, which can be recovered on the other side. This requires both sides to choose a start state and the same encoding/decoding algorithm. We choose the state with the largest asymptotic probability as the starting state because it is the state occurring with the largest probability as time goes to infinity.

To encode the message into the path of HMM, we round transition probabilities to the closest $\frac{1}{2^n}$, where n is an integer. Rounding keeps the statistical features of legitimate domain names, while finding a way to encode and decode. Since the original probabilities of all transitions going out of a state sum up to 1, it is possible to round them to the closest $\frac{1}{2^n}$. The set of transition probabilities associated with state S_i is denoted as $\{P_{i1}, P_{i2}, \dots, P_{ik}\}$, where k is the to-

- 1: **procedure** BINARIZATION($\{RP_{im}\}$)
- 2: order $\{RP_{im}\}$ from the largest to the smallest into $\{ORP_{i1}, ORP_{i2}, \dots, ORP_{ik}\}$
- 3: start with the largest value ORP_{i1} , and $BIN_ORP_{i1} = \underbrace{0\dots0}_{\log_2 \frac{1}{ORP_{i1}}}$
- 4: initiate $j = 2$, $last_input = ORP_{i1}$, $last_output = BIN_ORP_{i1}$, and $num = \log_2 \frac{1}{ORP_{i1}}$
- 5: **while** $j \leq k$ **do** \triangleright not all values are assigned
- 6: **if** $ORP_{ij} == last_input$: **then**
- 7: $BIN_ORP_{ij} = last_output + 1$
- 8: **else**
- 9: calculate $diff = \log_2 \frac{1}{ORP_{ij}} - num$
- 10: $BIN_ORP_{ij} = (last_output + 1) * 2^{diff}$
- 11: update $num = \log_2 \frac{1}{ORP_{ij}}$
- 12: **end if**
- 13: update $last_input = ORP_{ij}$
- 14: update $last_output = (last_output + 1) * 2^{diff}$
- 15: $j = j + 1$
- 16: **end while**
- 17: From the mapping between $\{RP_{ij}\}$ to $\{ORP_{ij}\}$, obtain $\{BIN_RP_{ij}\}$ from $\{BIN_ORP_{ij}\}$
- 18: **return** $\{BIN_RP_{ij}\}$
- 19: **end procedure**

Figure 2: Proability binarization algorithm

tal number of transitions leaving state S_i and $\sum_{n=1}^k P_{in} = 1$. The set of rounded transition probabilities associated with state S_i is denoted as $\{RP_{i1}, RP_{i2}, \dots, RP_{ik}\}$, where k is the total number of transitions leaving state S_i and $\sum_{n=1}^k RP_{in} = 1$.

With the rounded transition probabilities of state S_i , we associate a binary representation for each transition. This allows us to encode binary data (converted from the original ‘client message’) into the HMM. Figure 2 shows the binarization algorithm. The input is the rounded probability ($\{RP_{im}\}$, $m = 1, \dots, k$), and the output is the binary representation ($\{BIN_RP_{im}\}$) for the corresponding transitions. Note that any string can be encoded with the binary representations.

After binarization, each transition of the HMM has a binary representation. Given the binary data, there is a unique path going through the HMM that encodes the binary data string. However, there are two technical difficulties:

- The output symbols in HMM includes ‘a-z’, ‘0-9’, ‘-’ and ‘_’ (space). Normally, domain names generated from the HMM are separated with space (for example ‘ab_cd_ef’). But the receiver will not be able to put space in the correct spot (only put space after ‘ab’ and ‘cd’, but not ‘ef’).
- With receiving the domain names (‘ab’, ‘cd’ and ‘ef’),

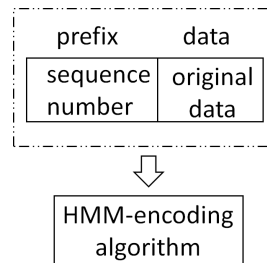


Figure 3: The new data structure

the receiver doesn’t know the order.

The solution is to add the prefix to the original data. The new data structure is in Figure 3. The prefix is the sequence number, which guarantees the correct order of the message on the receiver side.

After the client message is converted into multiple domain names, we register them as sub-domain names with the server IP address using dynamic DNS service (<https://freedns.afraid.org/> for example).

To decode the domain names into client message, the server will reverse-lookup a randomly chosen IP address and retrieve a list of sub-domain names. After doing the inverse of the encoding, several message pieces are recovered. Using the sequence number, the original client message is obtained.

Since both client and server will register the same randomly chosen IP address to make the system work, we have to consider:

- The probability of choosing a collided domain name is almost zero if using IPv6 domain names. The IPv6 address space range contains 2^{128} entries, which is more than one entry for every three atoms in the universe. If an address is chosen at random, the chance of a collision is essentially zero. If the entire IPv4 space were to choose IPv6 addresses at random, the probability of at least one collision is approximately 0.0155.
- If collision happens, the system will work fine. Because the HMM decoding won’t work on the existing domain names. By simply ignoring the domain names that fail to pass HMM decoding, we can separate the desired domain names from the existing ones.

Besides the two-way communication, the idea can also be used in botnets where the botnet C&C encodes the message into domain names, and the bot retrieves the message. This will make the botnet traffic look like pure DNS traffic.

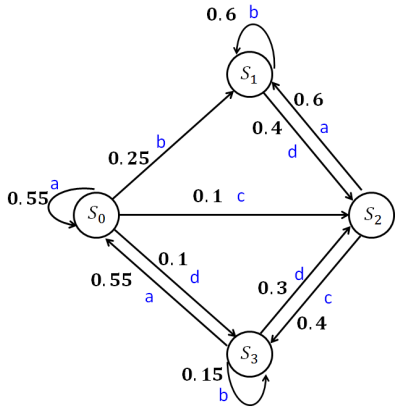


Figure 4: An example HMM for illustration

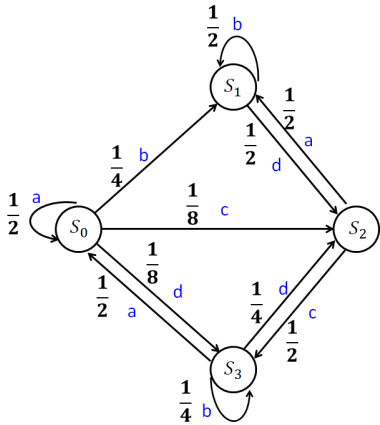


Figure 5: HMM after probability rounding algorithm

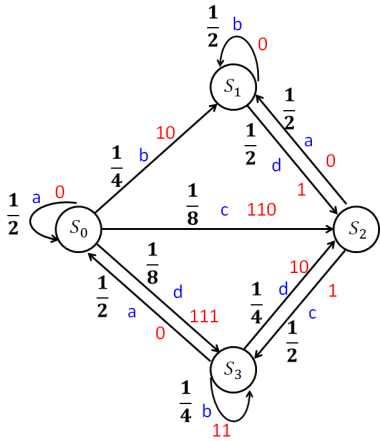


Figure 6: HMM after binarization algorithm

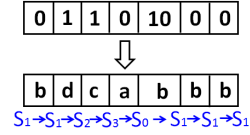


Figure 7: An example mapping

Table 1: Performance of HMM encoding/decoding

Input String Length	AESed String Length	Encode Execution Time	Decode Execution Time	Output Domain Names
11	24	267.98s	114.04s	lviazea01; lviahgeurakk-04; lvia79djqb02; lviamhlayae2-; lviabpi62f03; lvieIm13owul; lvienh103; lvwlei14j0volg002; lvwltrti102
22	44	460.03s	188.05s	lviaudzsud zssic20e0002; lvia523rajc10j02; lvia76rptu16q603; lviabpm01; lvwleg186bczp; lvienb-8702; lvwlejo01; lvwt-28d6 gyopur24-05; lvwlynaks; lvwl07bsjm- lvusfoe-1ofw23 ucm29c58-05; lvwlcko1-02
34	64	781.06s	191.32s	lviauket99jd104; lviaha102; lvia78h; lviamiltathybpr5102; lvienbpo-04; lvwlep003; lvwltb5eud01; lvwl1d1601; lvwlyunt9103; lvwlxmanrovp; lvwlcl67ds01; lvw6l0v30ge- nsigoyed6gy-01; luctgold99q; luctgjnbu9jv5wk; luctiwaa; luctigh5; luctia-thne55x001; luctin01; luctivbusovalriq5ilf 215lld-z2tchl0lcl2201

4.2 An illustrative example

The HMM representing the real legitimate domain names is very complex (it has 2995 states and each state has at most 38 outgoing transitions). We use an example HMM (Figure 4) to illustrate the algorithm. In the example HMM, the asymptotic probability of state S_i ($i = 0, 1, 2, 3$) is calculated as: $P(S_0) = 0.15$, $P(S_1) = 0.46$, $P(S_2) = 0.25$ and $P(S_3) = 0.14$. So S_1 is chosen as the starting state for both the client and the server.

After the probability rounding algorithm, all transition probabilities are rounded to the closest $\frac{1}{2^n}$ (Figure 5). After the binarization algorithm, each transition is associated with a binary representation (Figure 6).

Suppose that the ‘client message’ is ‘h’, its binary representation is ‘01101000’. The starting state is S_1 . Following Figure 6, the state transition is $S_1 \rightarrow S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_0 \rightarrow S_1 \rightarrow S_1 \rightarrow S_1$. The mapping is in Figure 7. With the HMM-encoding algorithm, the input ‘h’ is converted into ‘bdcabbb’.

4.3 Performance

We implemented the proof-of-concept idea on a laptop (OS: Windows 10, CPU: i5-5300U, RAM: 8G). We measure the encoding and decoding execution time on various lengths of input string. Table 1 shows the performance of the transparent protocol. For AES encryption/decryption, we use block size equal to 16. This will make the length of AESed-string 24 (when length of the input string is between 1-15), 44 (when length is between 16-31), 64 (when length is between 32 to 47) bytes etc. So the performance of the implementation is more dependent on the length of AESed-string than the length of the input string. Note that the input string can be any length.

From Table 1, we can see that the execution time is proportional with the length of AESed-string. Note that, all output domain names start with ‘l’, which is resulted from the same start state (the state with the largest asymptotic probability). How to pick different start states where the client and server agree on will be an interesting topic as the future work.

Figure 8 shows the network traffic sniffed by Wireshark. The server registers the domain names with a randomly chosen IP address (192.168.1.72). The client (192.168.1.80) is trying to retrieve a message from the server. All communication happens between the client to the local DNS server (192.168.1.254). And the DNS server cannot distinguish the DNS traffic from the others. This proves the destination IP address stays hidden and the idea works!

4.4 Security Analysis

To further validate the possibility of the implementation, we conduct security empirical analysis in terms of confidentiality, differentiability and communication.

- **Confidentiality:** Since the arbitrary network traffic is masquerading as normal DNS traffic, it is hard to filter it out from other DNS traffic. Even if a man-in-the-middle (MITM) filters out the traffic and tried to decode the DNS packets, he/she has to have all pre-shared information: AES key, PRG, PRG seed, and the HMM. This is infeasible unless he has the software package.
- **Differentiability:** Except for normal DNS queries, the application involves reverse-DNS lookup and DNS registration traffic. Reverse-DNS lookup is widely used by common security tools [43] including network troubleshooting tools, anti-spam techniques, and system monitoring tools. For example, email anti-spam software checks the domain names using reverse-DNS lookup to see if the source is a dynamically assigned address, which is unlikely used by a legitimate mail server. Web browsers use reverse-DNS lookup to verify the same origin of the requests to avoid DNS re-binding attacks [44]. In terms of DNS registration traffic, although large companies like Google and Amazon keep registering domain names in a round-robin fashion, it remains unclear that how unusual the DNS registration traffic is. This will be an interesting topic for future work.
- **Communication:** If there is an ISP-level surveillance tool looking for this application, it is impossible to pick up the related domain names. Even if it is possible, we can use Tor at the client side (who generates the domain names) to further hide the source IP address. Also, the registered IP address associated with the generated domain names is randomly selected, so there is no way to trace the communication parties using the IP address.

5 Conclusion and Future Work

This paper proposes the structure of a new covert data transport protocol. It would be suitable for botnet C&C. The advantages over existing protocol obfuscation tools are that network traffic is real and normal DNS lookup traffic on benign-looking domain names, which is protected against DPI detection. One possible disadvantage is the low throughput, which is more suitable for delay-tolerant communication than real-time chatting, such as twitter-like social networks, sensitive information retrieval tools etc. We hope this paper will provide a new insight into protocol obfuscation.

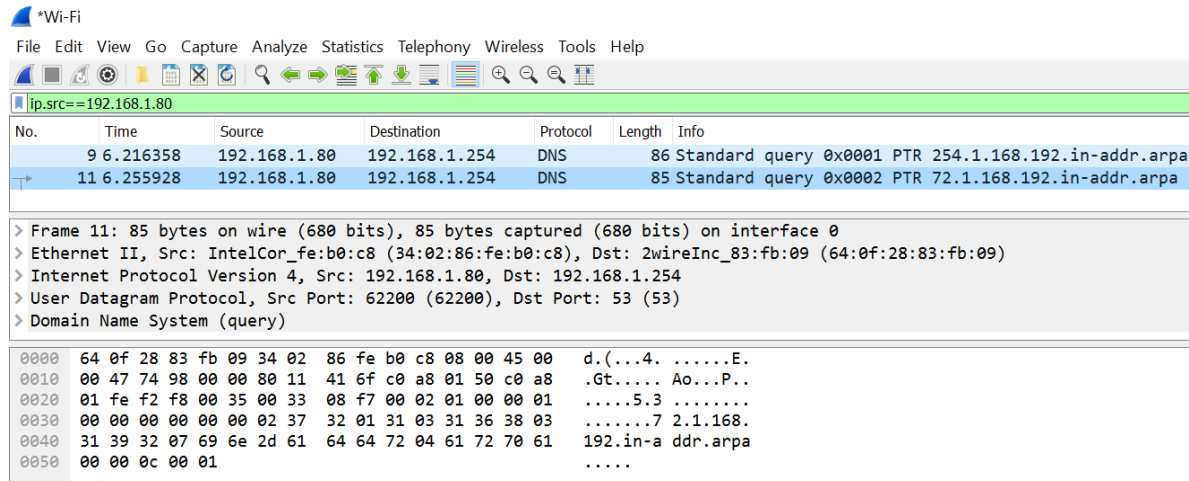


Figure 8: Wireshark screenshot

Future work can tune parameters to optimize the throughput including re-designing HMM encoding/decoding algorithm, better data structure to improve the performance etc.

The proposed concept will be helpful in moving botnet countermeasures to being pro-active. It will change the routines of botnets innovating, anti-virus vendors reverse-engineering and finding countermeasures [45]. Instead, we need to develop better technology than the enemy and know how to counter botnets before they deploy innovations. Up to now, the botnet implementers have the advantage of creating the innovations. By designing countermeasures to better botnet designs in advance, it would make botnet deployment less profitable.

Acknowledgment

This material is based in whole or in part upon work supported by the National Science Foundation under Grant numbers ACI-1547164 and CNS-1544910. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation

References

- [1] "Iran reportedly blocking encrypted internet traffic," <http://arstechnica.com/tech-policy/2012/02/iran-reportedly-blocking-encrypted-internet-traffic/>, [Last visited: 05-July-2016].
- [2] A. Houmansadr, C. Brubaker, and V. Shmatikov, "The parrot is dead: Observing unobservable network communications," in *Security and Privacy (SP), 2013 IEEE Symposium on*. IEEE, 2013, pp. 65–79.
- [3] K. Born and D. Gustafson, "Detecting dns tunnels using character frequency analysis," *arXiv preprint arXiv:1004.4358*, 2010.
- [4] G. Farnham and A. Atlasis, "Detecting dns tunneling," *SANS Institute InfoSec Reading Room*, pp. 1–32, 2013.
- [5] "Tor," <https://www.torproject.org/>, [Last visited: 06-Aug-2015].
- [6] T. T. Project, "obfsproxy," <https://www.torproject.org/projects/obfsproxy.html>, 2015, [Last visited: 06-Aug-2015].
- [7] P. Winter, T. Pulls, and J. Fuss, "Scramblesuit: A polymorphic network protocol to circumvent censorship," in *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*. ACM, 2013, pp. 213–224.
- [8] H. Mohajeri Moghaddam, B. Li, M. Derakhshani, and I. Goldberg, "Skypemorph: Protocol obfuscation for tor bridges," in *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012, pp. 97–108.
- [9] Z. Weinberg, J. Wang, V. Yegneswaran, L. Briesemeister, S. Cheung, F. Wang, and D. Boneh, "Stegotorus: a camouflage proxy for the tor anonymity system," in *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012, pp. 109–120.
- [10] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Protocol misidentification made easy with format-transforming encryption," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 61–72.
- [11] B. Wiley, "Dust: A blocking-resistant internet transport protocol," *Technical report*. <http://blanu.net/Dust.pdf>, 2011.
- [12] D. Brown, "Resilient botnet command and control with tor," 2010. [Online]. Available: <https://www.defcon.org/images/defcon-18/dc-18-presentations/D.Brown/DEFCON-18-Brown-TorCnC.pdf>

- [13] M. Casenove and A. Miraglia, "Botnet over tor: The illusion of hiding," in *Cyber Conflict (CyCon 2014), 2014 6th International Conference On*. IEEE, 2014, pp. 273–282.
- [14] M. Mimoso, "Shedding new light on tor-based malware," 2014. [Online]. Available: <https://threatpost.com/shedding-new-light-on-tor-based-malware/104651/>
- [15] K. J. Higgins, "Botnet behind mysterious spike in tor traffic," 2013. [Online]. Available: <http://www.darkreading.com/attacks-breaches/botnet-behind-mysterious-spike-in-tor-traffic/d/d-id/1140422>
- [16] N. Hopper, "Protecting tor from botnet abuse in the long term," Tech. Rep. 2013-11-001, The Tor Project, Tech. Rep., 2013.
- [17] E. Hjelmvik and W. John, "Breaking and improving protocol obfuscation," *Chalmers University of Technology, Tech. Rep.*, vol. 123751, 2010.
- [18] J. Karlin, D. Ellard, A. W. Jackson, C. E. Jones, G. Lauer, D. Mankins, and W. T. Strayer, "Decoy routing: Toward unblockable internet communication." in *FOCI*, 2011.
- [19] H. Bhanu, "Timing side-channel attacks on ssh," 2010.
- [20] C. J. Dietrich, C. Rossow, F. C. Freiling, H. Bos, M. Van Steen, and N. Pohlmann, "On botnets that use dns for command and control," in *Proceedings of European Conference on Computer Network Defense*, 2011, pp. 9–16.
- [21] E. Cole, "Hiding in plain sight," *Steganography and the Art of Covert Communication*, Wiley, 2003.
- [22] J. Thyer, "Covert data storage channel using ip packet headers," *SANS Institute*, 2008.
- [23] A. H. Altalhi, M. A. Ngadi, S. N. Omar, and Z. M. Sidek, "Dns id covert channel based on lower bound steganography for normal dns id distribution," *International Journal of Computer Science Issues(IJCSI)*, vol. 8, no. 6, 2011.
- [24] M. Ngadi, S. Omar, and I. Ahmedy, "Indirect dns covert channel based on base 16 matrix for stealth short message transfer," *International Journal of Computer Science Issues(IJCSI)*, vol. 8, no. 6, 2011.
- [25] S. R. Eddy, "Hidden markov models," *Current opinion in structural biology*, vol. 6, no. 3, pp. 361–365, 1996.
- [26] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.
- [27] C. Lu, J. M. Schwier, R. M. Craven, L. Yu, R. R. Brooks, and C. Griffin, "A normalized statistical metric space for hidden markov models," *IEEE transactions on cybernetics*, vol. 43, no. 3, pp. 806–819, 2013.
- [28] C. Lu, "Network traffic analysis using stochastic grammars," 2012.
- [29] J. Schwier, "Pattern recognition for command and control data systems," 2009.
- [30] B. Vanluyten, J. C. Willems, and B. De Moor, "Equivalence of state representations for hidden markov models," *Systems & Control Letters*, vol. 57, no. 5, pp. 410–419, 2008.
- [31] L. Yu, J. M. Schwier, R. M. Craven, R. R. Brooks, and C. Griffin, "Inferring statistically significant hidden markov models," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 7, pp. 1548–1558, 2013.
- [32] X. Zhong, A. Ahmadi, R. Brooks, G. K. Venayagamoorthy, L. Yu, and Y. Fu, "Side channel analysis of multiple pmu data in electric power systems," in *Power Systems Conference (PSC), 2015 Clemson University*. IEEE, 2015, pp. 1–6.
- [33] D. Andriessse, C. Rossow, B. Stone-Gross, D. Plohmann, and H. Bos, "Highly resilient peer-to-peer botnets are here: An analysis of gameover zeus," in *Malicious and Unwanted Software: The Americas (MALWARE), 2013 8th International Conference on*. IEEE, 2013, pp. 116–123.
- [34] S. Shin, G. Gu, N. Reddy, and C. P. Lee, "A large-scale empirical study of conficker," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 676–690, 2012.
- [35] P. Amini and C. Pierce, "Kraken botnet infiltration," 2008.
- [36] J. Wolf, "Technical details of srizbis domain generation algorithm," 2008.
- [37] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydowski, R. Kemmerer, C. Kruegel, and G. Vigna, "Your botnet is my botnet: analysis of a botnet takeover," in *Proceedings of the 16th ACM conference on Computer and communications security*. ACM, 2009, pp. 635–647.
- [38] S. Yadav, A. K. K. Reddy, A. N. Reddy, and S. Ranjan, "Detecting algorithmically generated domain-flux attacks with dns traffic analysis," *IEEE/Acm Transactions on Networking*, vol. 20, no. 5, pp. 1663–1677, 2012.
- [39] H. Zhang, M. Gharaibeh, S. Thanasoulas, and C. Papadopoulos, "Botdigger: Detecting dga bots in a single network," 2016.
- [40] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, and D. Dagon, "From throw-away traffic to bots: detecting the rise of dga-based malware," in *Presented as part of the 21st USENIX Security Symposium (USENIX Security 12)*, 2012, pp. 491–506.
- [41] R. team, "Welcome to project sonar!" 2013, <https://community.rapid7.com/community/infosec/sonar/blog/2013/09/26/welcome-to-project-sonar>.
- [42] X. Zhong, Y. Fu, L. Yu, R. Brooks, and G. K. Venayagamoorthy, "Stealthy malware traffic-not as innocent as it looks," in *2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*. IEEE, 2015, pp. 110–116.
- [43] Wikipedia, "Reverse dns lookup," 2016. [Online]. Available: https://en.wikipedia.org/wiki/Reverse_DNS_lookup
- [44] J. K. N. T. L. T. Savolainen, Nokia and N. Inc., "Improved recursive dns server selection for multi-interfaced nodes," 2012. [Online]. Available: <https://tools.ietf.org/html/rfc6731#page-24>
- [45] Y. Fu, B. Husain, and R. R. Brooks, "Analysis of botnet counter-counter-measures," in *Proceedings of the 10th Annual Cyber and Information Security Research Conference*. ACM, 2015, p. 9.