DETECTION OF DATA EXFILTRATION USING ENTROPY AND ENCRYPTION CHARACTERISTICS OF NETWORK TRAFFIC

T. W. Fawcett*, C. G. Boncelet Jr., C. J. Cotton, and W. D. Sincoskie
University of Delaware
Newark, DE 19716

ABSTRACT

This paper describes the use of entropy characteristics combined with protocol analysis to ascertain whether egress network traffic from computer systems is malicious. For entropy, we conclude the inspection of egress traffic should be performed at the session level rather than the packet level. We describe a detection scheme capable of distinguishing data exfiltration from benign traffic and incorporate this detection scheme into a tool called ExFILD, which identifies suspicious traffic by comparing its observed state of encryption to its expected state of encryption.

1 INTRODUCTION

Many security tools and strategies emphasize the prevention and detection of intrusion into a system. Intrusion prevention has the obvious benefit of protecting the system before damage occurs. Another line of defense can be constructed by paying attention to the egress traffic, because no intrusion detection system is perfect. Additionally, certain system compromises may not involve a detectable intrusion, e.g. in cases of insider attack or supply chain attacks where the intrusion occurs before the system is placed into service.

An attacker that slips past the intrusion detection system could potentially steal information from a system indefinitely if there is no inspection of the outgoing traffic. The problem is commonplace, including organizations as sophisticated as large enterprises and nation states. The data breach of the Joint Strike Fighter project is an excellent example of an enormous amount of data leaving a network before being noticed. According to (Cole et al., 2009), attackers were able to exfiltrate several terabytes of data about the Pentagon's Joint Strike Fighter project.

2 RELATED WORK

Researchers from the University of California at Berkley created a system named Glavlit (Schear et al., 2006) to prevent data from being exfiltrated. Their system uses a whitelisting approach. It has two main parts to the system that are referred to as a guard system and a warden system. The guard prevents any traffic from exiting the network without approval from the warden. The warden is in charge of approving all of the traffic. The approval process can be automated with content matching or system administrators can manually approve each piece of data by hand.

Researchers from University of California at Davis and Sandia National Laboratories developed a framework to detect data exfiltration called SIDD, Sensitive Information Dissemination Detection (Liu et al., 2009). SIDD has three major components: application identification, content detection, and covert channel detection. The components are inline and each component can cause an object to exit the chain and assign some action to be taken.

The application identification component tries to determine the application of the traffic and uses a policy to determine if it should be allowed. The content detection component checks traffic for data that has been labeled as sensitive. The searching for the sensitive data is signature-based. The last component handles covert channel detection. The covert channel detector focuses on digital audio channels. Steganalysis described in (Liu et al., 2009) is used to generate characteristics of data and decide whether a covert channel exists. The work present in this paper is complimentary to the prior work described in this section.

3 ENTROPY EXPERIMENT

This section describes the calculations performed to extract the entropy information of egress network traffic. We discuss the standard entropy calculation (Section 3.1) and our scaled version of the entropy calculation (Section 3.2).

3.1 Entropy Calculation

We assume that all highly random data is either encrypted or compressed. Data encrypted through good encryption algorithms should appear random when observed at the bit level. Entropy is used to characterize the randomness of data. Data that is very structured has low entropy, while data produced by a good encryption algorithm has

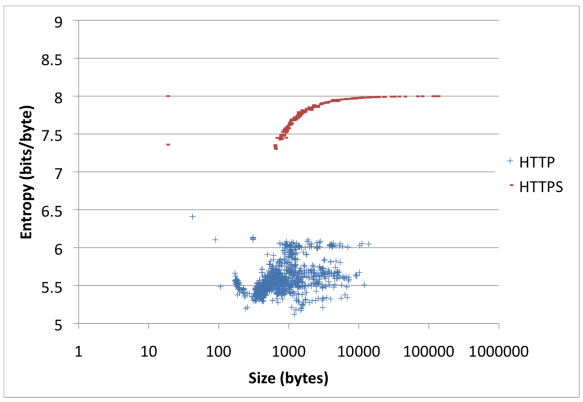

Figure 1: HTTP and HTTPS sessions

higher entropy. By comparing the payloads of HTTP and HTTPS sessions, we obtain a good demonstration of the difference in entropies between encrypted and unencrypted network traffic. Figure 3.1 shows the separation between the entropies of the HTTP and HTTPS sessions. The separation between the session entropies allows for us to observe the state of encryption using a simple threshold.

Equation 1 is used to calculate the entropy of data (in our case a session's payload), which was derived by C. E. Shannon (Shannon, 1951). Conceptually, it produces a histogram of possible bit combinations. If the distribution of the data values is even across all of the possible combinations, then the data has high entropy. Conversely, an uneven distribution of the possible combinations results in a lower entropy value.

$$-\sum_{i=1}^{N} p\left(x\right) \log_2 p\left(x\right) \tag{1}$$

In Equation 1, N is the number of possible combinations. N is 256 for this work, because we inspect 8 bits at a time, hence there are 2^8 or 256 possible combinations. The calculation is performed 8 bits, or one byte, at a time, because most computers communicate in multiples of bytes. Also, human readable text is common in computer communications, each character being represented by an 8-bit code. The observed probability of each of the possible bit combinations is represented by the function p(x). This entropy calculation can range from 0 to 8.

3.2 Scaling of the Entropy Calculation

The maximum entropy value for payloads that are not multiples of 256 bytes in length is less than 8. The maximum possible entropy for payloads that are less than 256 bytes in length can be significantly less than 8. The maximum possible values for the range of 0 to 256 bytes in length increase from 0 to 8 as the size increases. Since entropy values are used to determine whether traffic is encrypted, and we wish to have a fixed threshold, it is necessary for the entropy values to be scaled.

Two solutions were proposed with only one achieving the goal. The first solution was to initialize the number of occurrences for each possible combination to a number larger than 0. This solution normalized the maximum entropy over all possible payload sizes, but it raised the minimum values to start at 8 and then they decreased as the size of the payload increases.

The other solution is to scale the entropy with respect to the size of the payload. This solution normalizes the maximum entropy while maintaining the minimum entropy for all payload sizes. The scaling of the entropy is performed by multiplying it with the simple scalar shown by Equation 2.

$$scalar = \frac{8}{\log_2 \left(payload\_size\right)} \tag{2}$$

4 DESIGN

This section describes the design of ExFILD with the two design components detailing encryption characteristics (Section 4.1) and the decision-making tree (Section 4.2).

4.1 Encryption

Encryption is a standard for people who want to ensure the privacy of their data. Encryption obfuscates data in order to prevent unauthorized parties from being able to read it. When encryption is implemented correctly it is an effective method of protecting data. Although encryption is very useful it can also be used against people. For example, attackers can encrypt data they are exfiltrating in order to hide it from system administrators. Egress network traffic can automatically be considered suspicious once encryption is observed, because the payloads' contents are unknown to the administrator. The administrator may be able to whitelist some of the egress network traffic given basic knowledge about it. For example, a person that uses Gmail's web-based email client typically trusts encrypted traffic destined for Google web servers over the port 443 (HTTPS).

4.1.1 Expected Encryption

Knowing whether network traffic is encrypted is not enough to make a determination of whether the outgoing network traffic is benign or malicious. It is important to know if encryption is expected. The discrepancy between

the expected and observed state of encryption can be a good discriminator of suspicious traffic.

The most specific protocol associated with the payload is used to determine if encryption is expected. The application layer protocol, which is determined by the IANA assigned ports (IANA, 2010), is used a majority of the time, but other layers' protocols may be used in absence of application layer protocol. If the expected state of encryption is unknown for a protocol it is assumed that unencrypted traffic is expected.

4.1.2 Observed Encryption

Given our expectation of the traffic's state of encryption, we now need to determine whether traffic is encrypted. The state of encryption for a session is ascertained using the scaled entropy from Section 3.2. A comparison is performed between the entropy values and a threshold.

The comparison between the session entropies and the threshold is trivial; however, setting the threshold is not. As with all thresholds, careful attention should be placed to prevent setting it too high or too low. A threshold that is set too low or high results in traffic's state of encryption being incorrectly labeled. Incorrectly labeling the traffic leads to ExFILD producing incorrect results, specifically false positives and negatives. Within ExFILD, false positives cause time and resources to be wasted investigating benign traffic, while false negatives result in suspicious traffic exiting the system unnoticed.

The initial threshold for the program was set using the traffic from the Control data set later described in Section 5. HTTP and HTTPS were used as the basis for setting the threshold. The reason these two protocols were used is that they perform the same function, but HTTP is not encrypted while HTTPS is encrypted. An important observation to take away from Figure 3.1 is the separation between the HTTP and HTTPS entropies. The separation between encrypted and unencrypted traffic is the attribute of entropy that allows for the determination of whether a payload is encrypted.

The result of averaging the entropies for all of the HTTP and HTTPS sessions was used to set the threshold. The entropies have enough separation that the average of the sessions is a good value to delineate between the two. The threshold was calculated to be 6.51237, and was approximated to 6.5 for this paper.

4.2 Decision-Making Tree

ExFILD's decision-making tree is responsible for determining whether egress traffic is suspicious. The detection of data being exfiltrated is based on the comparison between the expected and observed state of encryption. It includes other checks to filter out known benign and malicious activities.

The tree is split into four branches when the comparison between the expected and observed state of encryption is performed. The four branches are expect and observe unencrypted, expect unencrypted but observe encrypted, expect and observe encrypted, and expect encrypted but observe unencrypted. The branches where the expected and observed states match represent traffic as expected, therefore they are not typically interesting. The branches with more interesting results are the two where the expected and observed states do not match. These discrepancies in the two branches result in the traffic being considered suspicious.

4.2.1 Additional Checks

The knowledge of the encryption characteristics provides a sound foundation to base whether or not the traffic is suspicious. Additional checks are made to customize the detection scheme for the network and to remove payloads that contain compressed data and unencrypted data.

All traffic is checked against a whitelist and blacklist. These checks are added to allow for IP addresses that are always or never trusted to be ignored or alerted without further examination. The main purpose of these checks is to allow a user to whitelist an application that continues to cause alerts and the user does not believe the exfiltrated data is inappropriate.

Skype provides a good example of an application a user may want to whitelist. Skype communicates over encrypted sessions with many random IP addresses on non-standard ports, so its outgoing traffic appears suspicious to a user. A Skype user would want to whitelist the application's traffic to remove the unnecessary alerts. Many of these alerts can be removed by whitelisting any IP address maintained by Skype. Other alerts can be whitelisted using a host-specific port that can be set in Skype's preferences. Without a whitelist check many sessions cause alerts that can be considered false positives to the user.

Traffic having high entropy is inspected to see if it is a compressed file being served by a web server. Compressed data also has high entropy, which allows for the possibility that it could be confused with encrypted traffic. The check for compression uses the magic file numbers for known compression algorithms to decide if the traffic contains a compressed file. More thorough discrimination is being planned as future work to find more compressed files as well as to decompress them and inspect the files' con-

tents for confidential or encrypted data. (Typically, data is compressed before encryption. However, steganographic exfiltration might be attempted by encrypting before compression. Decompression and an entropy calculation can be used to detect some of these attempts.)

The traffic observed to be unencrypted is inspected for censored strings. The payloads are searched for strings that are determined to be confidential. The confidential strings are defined by strings or regular expressions stored in a file. Examples of confidential data can be credit card numbers, social security numbers, or files containing the phrases "confidential" or "top secret." Many techniques exist for inspecting unencrypted traffic, so further checks are beyond the scope of this work.

5 EVALUATION

This section discusses the results from experiments used to evaluate ExFILD and provides an overview of the data sets used (Section 5.1). We then briefly discuss the choice to use session alerts rather than packet alerts (Session 5.2). Finally, we provide the performance results for the detection schemes used within ExFILD (Section 5.3).

5.1 Experiments

ExFILD was run against eight different data sets. The data sets include a control, sets created by basic users, and sets including traffic from malware samples. The control and the user created data sets provide a good mixture of encrypted and unencrypted traffic. The malware samples provide real examples of how data can be exfiltrated from a host. The packet captures for the malware were obtained from (Mu Dynamics, 2010) and (Bejtlich and Cummings, 2010). The packet captures are verified to contain malware traffic in (Fawcett, 2010).

The Control, Kraken, Zeus, and Blackworm data sets all contain data being exfiltrated. The Control data set contains traffic from a variety of applications as well as multiple examples of data exfiltration using FTP and FTPS, which were intentionally included. The Kraken data set contains samples of data being exfiltrated on a nonstandard port using a custom encryption protocol. The Zeus data sets contain data exfiltration using HTTP POST commands. The Blackworm data set provides a sample of data exfiltration using the NETBIOS suite.

5.2 Packet Versus Session Alerts

It was shown in (Fawcett, 2010) that egress traffic should be processed at the session level rather than the packet level when using entropy as an indicator. The entropy calculation made on smaller packets, specifically packets smaller than 256 bytes, can be misleading when determining whether a packet is encrypted or not. Inspecting the egress traffic at the session level allows information used for the same purpose to be aggregated and allow for more accurate entropy calculations, thus less false positives and negatives. The aggregation of the packets into their respective sessions also reduces the number of alerts, which simplifies incident response actions.

5.3 Detection Results

The results from running the 8 data sets from Section 5.1 are shown in Table 1. For this work positives and negatives directly correspond to whether ExFILD alerts on a session, with an alert being a positive. The definition for true and false is a little more complex. A true is any session containing information in its payload about or from the host that a user considers private and does not want leaving the network. However, information leaving the network used to establish or control the outbound connection is not private and does not cause a true. A session containing any private data is considered a true, while a negative is any session that contains no private data leaving the network.

5.3.1 Control Data Set

Forty-three unique sessions were alerted on from the Control data set. Skype caused 2 of the alerts. In this work Skype is considered to be trusted, so its IP addresses and incoming connection port (defined on a host-by-host basis) are whitelisted. However, the fact that Skype's source code has not been released, combined with its use of encrypted sessions connected to many random hosts, a consequence of its peer-to-peer model, make it suspect to security conscious administrators, since it is indeed exfiltrating encrypted data to random destinations. It was shown in (Fawcett, 2010) that without the whitelisting it results in 84 session alerts. Since the Skype traffic was supposed to be whitelisted, these two sessions causing alerts are considered false positives. A better way to handle Skype needs to be developed.

The controlled exfiltration added into the Control data set caused 41 session alerts. There were 14 file transfers using FTPS on a nonstandard port, with one of the transfers failing. Each of the FTPS transfers resulted in 2 sessions for setup and control and 1 session for the data transfer. The 13 sessions responsible for the encrypted transfer of the file caused alerts due to them being encrypted on an unknown port. These 13 sessions actually contain private exfiltrated data therefore they are true positives.

The other 28 alerts resulted from the sessions used to initiate and control the FTPS session. These sessions do

Table 1: Session Alert Metrics

Data Set	Outgoing Sessions	True Positives	False Positives	True Negatives	False Negatives
Control	2,779	13	30	2,736	0
One	9,789	0	0	9,789	0
Two	3,348	0	12	3,336	0
Kraken	53	16	0	37	0
Zeus 1	72	3	0	59	10
Zeus 2	5	3	0	2	0
Zeus 3	5	3	0	2	0
Blackworm	10	1	0	9	0

not contain private data, and for that reason they should not cause an alert. The sessions are labeled as false positives. Decoding the application layer protocols may help with this problem and allow for the control sessions to be identified and ignored.

5.3.2 Data Set #2

Twelve alerts resulted from running ExFILD against data set #2. All of these alerts resulted from single packet NETBIOS sessions. The sessions had payloads containing only 5 bytes of data each. These false positives are due to their extremely small payloads. For example, a payload of only 5 bytes containing the word "hello" has an entropy value of 6.621 and thus appears to be encrypted. This is a much larger problem at the packet level, and is one of the reasons to alert at the session level. A better solution needs to be developed to handle sessions with very small payloads. A possibility is to aggregate small sessions together using less specific host and destination pairs of the IP addresses and ports.

5.3.3 Kraken Data Set

ExFILD performed very well against the Kraken data set. It alerted against all of the sessions that contained exfiltrated data, and did not falsely alert against any other sessions. The sessions were alerted due to the encrypted communication over a non-standard port. Note that the fact that the packet capture containing the malware samples are small does not affect the results from ExFILD. ExFILD processes each session individually and the results from each session are independent of any other sessions in the data set.

5.3.4 Zeus Data Sets

The Zeus data sets caused 9 total alerts. The data sets contain a total of 19 sessions with HTTP POST commands exfiltrating data that should have caused alerts. The 9 sessions characterized as false negatives were observed to be unencrypted, which lead to the inspection of these sessions by hand. The HTTP header information was removed and only the payloads were used to calculate the entropy. The entropies for these payloads are higher than 7.5 bits/byte, which is much higher than the threshold. The decoding of the application layer protocols would mitigate this problem, and improve the entropy accuracy for all sessions. It is discussed more in Section 6.2.

5.3.5 Blackworm Data Set

The Blackworm data set caused a single session alert. The session contained the Blackworm worm spreading to a target host. The data leaving the host is the Blackworm executable being transferred from the infected host to the target host. It was alerted on because the packed executables have high entropy, therefore are labeled encrypted. ExFILD was not originally expected to alert on this, but it is a nice side effect since worms spread using peer-to-peer methods that involve the executable being transferred from an infected host to the target system, which in many cases is on the local network.

5.4 Detection Performance

The true and false positive rates are shown in Table 2. Data sets #1 and #2 did not have any true positives or false negatives, so their true positive rates could not be calculated. The true positive rate of Zeus #1 and the false positive rate of the Control data set can be improved with the decoding of the application layer protocol. The false positive rate of data set #2 can be improved with a better solution for the entropy calculation for small session payloads.

6 FUTURE WORK

ExFILD has known limitations that can be mitigated with future work. The limitations include hiding exfiltrated

Table 2: True and False Positive Rates for Each Data Set

Data Set	True Positive Rate	False Positive Rate
Control	1	0.0108
One	N/A	0
Two	N/A	0.0036
Kraken	1	0
Zeus 1	0.2308	0
Zeus 2	1	0
Zeus 3	1	0
Blackworm	1	0

data in protocols being used as expected, including application layer header information within the entropy calculation, and only supporting a single host. Section 6.1 describes future work to handle an entire network and Section 6.2 describes the decoding the application layer protocol, which mitigates the first two limitations.

6.1 Handling A Network

The program currently only inspects the outgoing traffic of a single host at a time. Under normal circumstances administrators are more interested in all of the traffic leaving a network, not just a single host. They only check a single host when its activities have been identified as suspicious. If an administrator wants to look at multiple hosts' outgoing traffic, ExFILD needs to be run multiple times with different IP addresses as the inputs. It would be useful to add the capability of looking at an entire network's outgoing traffic.

6.2 Application Layer Decoding

One of the improvements that can increase the accuracy of the entropy calculations is to decode the sessions to the application layer. For example, ExFILD extracts the payload of a HTTP sessions with the headers, since the HTTP protocol itself is not decoded. The HTTP cleartext headers are large enough to influence the entropy of the payload to appear unencrypted, when it is actually encrypted. Decoding the application layer protocols could remove the headers from the data and perform the entropy calculation only on the payload.

The decoding of the application layer protocol could allow for the discovery of the data being exfiltrated over normal channels. For example, an attacker could be exfiltrating data over an encrypted channel on port 443 (standard HTTPS port) using a custom encryption protocol. Decoding the session could show that in reality it is not using HTTPS and perform additional processing on the session.

CONCLUSION

This paper shows that normalized entropy can be used to effectively discern encrypted traffic from unencrypted traffic. When combined with rudimentary protocol analysis, a mechanism for detection of exfiltrated data is achieved.

ExFILD has proven to be capable of detecting private data being exfiltrated from a host, specifically from those infected with malware. It has demonstrated a proof of concept for an effective detection algorithm that could be extended by decoding of the application layer protocols of the egress traffic and improving the handling of small sessions.

REFERENCES

Bejtlich, R. and J. Cummings, 2010: Openpacket. `http://www.openpacket.org/`.

Cole, A., Y. Dreazen, and S. Gorman, 2009: Computer Spies Breach Fighter-Jet Project. *The Wall Street Journal*, `http://online.wsj.com/article/SB124027491029837401.html`.

Fawcett, T., 2010: ExFILD: A Tool for The Detection of Data Exfiltration Using Entropy and Encryption Characteristics of Network Traffic. M.S. thesis, University of Delaware.

IANA, 2010: Port Numbers. `http://www.iana.org/assignments/port-numbers`.

Liu, Y., C. Corbett, R. Archibald, B. Mukherjee, and D. Ghosal, 2009: SIDD: A Framework for Detecting Sensitive Data Exfiltration by an Insider Attack. *Proceedings of the 42nd Annual Hawaii International Conference on System Science*, 1–10.

Mu Dynamics, 2010: pcapr. `http://www.pcapr.net/`.

Schear, N., C. Kintana, Q. Zhang, and A. Vahdat, 2006: Glavlit: Preventing Exfiltration at Wire Speed. *Proceedings of the 5th ACM Workshop on Hot Topics in Networks (HotNets-V)*, Irvine, CA.

Shannon, C. E., 1951: Prediction and Entropy of Printed English. *The Bell System Technical Journal*, **30**, 50–64.