2015 7th International Conference on Cyber Conflict:
Architectures in Cyberspace
M.Maybaum, A.-M.Osula, L.Lindström (Eds.)
2015 © NATO CCD COE Publications, Tallinn

Exfiltrations Using Polymorphic Blending Techniques: Analysis and Countermeasures

Matteo Casenove
Vrije Universiteit
Amsterdam, The Netherlands
m.casenove@gmail.com

Abstract: Cyber espionage campaigns and cyber attacks make use of data exfiltration on a regular basis causing damages for billions of dollars. Nowadays, they represent one of the primary threats, and they are performed by criminals, companies and states. Normally, data exfiltration uses classic application-layer protocols (e.g. *FTP* or *HTTP*) in combination with very basic obfuscation mechanisms. Even though in most cases these techniques are effective enough, this paper describes how instead they can be detected using properly configured *IDSs*. Moreover, we introduce a novel approach named *polymorphic blending exfiltration* that serves to avoid detection from signature-based as well as anomaly-based *IDSs*. This technique permits to blend the exfiltrated data in the normal and legitimate traffic. We show how *IDSs* can be easily improved in order to be able to detect such exfiltration. Finally, we conclude presenting different evasion techniques that can be included in the polymorphic blending exfiltration to keep providing a safe undetectable exfiltration.

Keywords: *cyber-espionage, exfiltration, obfuscation, IDS*

1. INTRODUCTION

Over the last ten years cyber security has been dealing with the major threat of data loss due to cyber espionage campaigns and cyber attacks. Besides the trivial technical security implications, it also has a substantial economic impact on companies and states; therefore, nowadays, it sits on top of the list of the most dangerous cyber threats. The action commonly associated with stealing data is called *data exfiltration* [1] and it corresponds with moving data without authorisation from a compromised machine to an external drop-zone controlled by the attacker. Security experts strive to secure the internal network from the external one, often overlooking the threats coming from the internal and more trusted network. Within an

organization, or a company, information is a critical resource as it carries personal client data, classified company data or any other information that could cause substantial damages to its owner if not adequately protected. Due to its criticality it is called *sensitive information* and it represents the target of the exfiltration activity.

During the exfiltration process, it is crucial that the activity does not raise any suspicion and, most importantly, it is not itself detected. In fact, as soon as the exfiltration is detected, security personnel can stop the attacker's operation and enhance the security level so that the possibility of security breaches decrease. In a computer system, the last actor of the security chain is the *Intrusion Detection System (IDS)* [2], which performs traffic inspection in order to detect malicious activity and - in case - raises an alert. Obviously, in a computer system there can be many other security solutions but in this work we only focus our study on IDSs. Knowing the presence of these security systems, the attacker makes use of different techniques for the purpose of avoiding detection during an exfiltration, such as social engineering, steganography and encryption, or common protocols [3][4]. Many of the detected malware and espionage campaigns have been found to be using a single or a combination of these exfiltration methods [5][6].

In this work we argue that these techniques can be detected not by the conventional *signature-based IDSs* but instead, by the more *advanced anomaly-based IDSs*. Moreover, we propose and implement a more advanced exfiltration technique named *Polymorphic Blending Exfiltration (PBE)* based on the classic *Polymorphic Blending Techniques (PBT)* [7] in order to evade the *anomaly-based IDSs* as well. This technique tries to emulate the normal behaviour of the network to blend the exfiltration in the normal traffic.

The contribution of this work is threefold: a) it shows that *IDSs* can be evaded by using our new polymorphic blending technique, b) it presents a tool that uses this technique successfully against state of the art *IDSs*, and c) it shows that the exfiltration tool can take advantages of the *traffic feature tolerance* allowed by the *IDS* in order to avoid high false-positive rate.

The remainder of the paper is organised as follows. In Section 2 we hand over the very limited literature about exfiltration. Section 3 describes the exfiltration problem and the polymorphic blending technique. In Section 4, the paper presents the exfiltration tool and the tests performed to evaluate its exfiltration performances. Section 5 discusses countermeasures that *IDSs* can apply in order to detect our exfiltration and then what we can improve in our exfiltration technique. Finally, Section 6 contains the conclusions of our work and Section 7 paves the way for future research.

2. RELATED WORKS

The Polymorphic Blending Technique was first addressed by Fogla in [7] and applied only for avoiding detection when sending exploits. The first phase of this technique collects the traffic features and it creates the traffic profile, while it is in the second phase that the real attack happens and the traffic manipulation comes to be. Perdisci *et al*. in [8] presents McPAD,

an anomaly-based intrusion detection system able to detect the polymorphic blending attack introduced by Fogla. In our work, we use the polymorphic blending technique by inverting the direction of the attack: instead of sending the attack from the hacker's machine to the target machine, we apply the *PBT* for exfiltrating data from the infected machine to the hacker's device. One of our goals in this work is to test our exfiltration against *McPAD* in order to determine whether it is still able to detect our implementation of the technique.

Amit in [9] and Antwerp in [1] present the most common exfiltration techniques that have been seen in the wild, such as *HTTP Post, FTP, DNS tunnelling, VoIP etc*. The two works describe these methods and how they can be detected. Antwerp in particular provides a framework where to use these methods and where to test the network security.

Wendzel *et al*. in [10] present the concept of Network Steganography. It is the same technique that we call blending: hiding information in the network traffic. They compiled a state-of-the-art survey on several techniques, which use well-known protocols in order to hide information in the network traffic. They use common high-level protocols like *VoIP, P2P*, and *Google* search queries as well as low-level steganography such using *WLANs* padding frames or cross-virtual machine information leakage for Cloud Computing. Moreover, Wandzel in his PhD Thesis [11] provides a very complete and detailed picture on the field of Covert Channels. He describes different techniques and uses of covert channels as well as few solutions able to detect or stop the leakage of information that exploits these channels. Our technique can be seen as a covert channel but, differently to the ones proposed by Wandzel, we do not exploit the protocols injecting extra data in the existing communication, instead we create a new exfiltration connection emulating the content of the packets of the legitimate connections.

Yarochkin *et al*. in [12] introduced a so-called *Network Environment Learning* phase used by covert channel in order to detect the legitimate protocol to use. This learning phase permits to identify the peers of the communication and which are the protocols that can be used as covert channels. In our work, the same technique is used in our collecting phase in order to create the profile for the legitimate traffic.

Khattak *et al*. in [13] discuss the problem of passing through censorship-resistant systems. They present the exfiltration techniques that can be applied to avoid the censorship monitors - in particular the *Great Firewall of China* - by using known *NIDSs* vulnerabilities. They exploit flaws in the *TCP* and vulnerabilities in the *IDSs*.

Houmansadr *et al*. in [14] study the vulnerabilities of censorship-resistant communication systems. These systems just partially implement well-known communication protocols like *Skype* trying to look like legitimate traffic. The research shows the lengthy list of requirements these censorship-resistant communication systems must respect in order to avoid detection and, due to this conspicuous list, the work concludes by stating the failure of the *"unobservability by imitation"* approach.

Fawcett in [15] proposes a possible solution to fill the gap between the advanced exfiltration techniques and the ability of detecting them. He uses the entropy characteristics of network

traffic and the observation of the traffic state of encryption in order to distinguish data leakage from benign data. He contributes to the implementation of the detection tool called *ExFILD*, which is able to detect exfiltration from normal traffic by using heuristics on the traffic entropy. In our case, this approach is not effective since with polymorphic blending we tend to maintain the same entropy as the normal traffic.

Bolzoni *et al*. in [16] present *ATLANTIDES*: an architecture for automatic alert verification in network intrusion-detection systems. Using *ATLANTIDES* they intend to reduce the number of false positive by using correlations between the input and output traffic. Unfortunately, the system requires a training phase where it records the normal output traffic per host. In our case, the server represents the host and it is contacted for the first time for the exfiltration and consequently *ATLANTIDES* cannot create any profile.

3. EXFILTRATION

Nowadays, we observe an enormous activity of information theft such as espionage campaigns, credential thefts or intellectual property thefts, and each of them shares a common denominator: the action of extracting sensitive data from an infected machine. This action is called data exfiltration and it targets *sensitive information*, which is defined as information to which an unauthorised loss, misuse, access, modification, or disclosure may produce an adversely security effect [17].

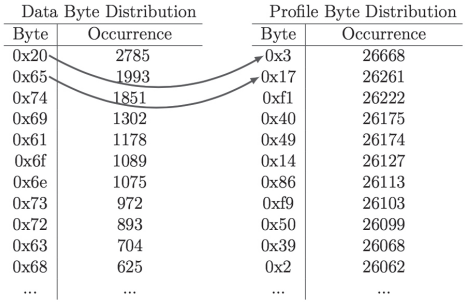
The one and only concern of an exfiltration is to avoid detection of all the security systems placed in the computer system. Among all of them, we only focus on a subset of them such as the so-called Intrusion Detection Systems (*IDSs*). These are monitoring systems designed to detect malicious activities and, upon detection, raise alerts. In order to evade *IDSs*, the attacker uses different methods for exfiltrating data [1]. She can use standard high-level protocols such as *HTTP Post* [18], *DNS Tunnelling* [19], *FTP, Skype* [20], *etc*. trying to make the exfiltration look like a legitimate traffic and avoid suspicions, or she can use transport layer protocols (*TCP* or *UDP*) applying encryption or obfuscation in order to make the *IDSs* deep-packet inspection useless. In the first case, by using standard protocols, the attacker blends the data in the normal traffic, while in the second case she manipulates the data so that it can be unrecognisable. In this work, we use the Polymorphic Blending Technique (*PBT*), which combines the two previously described cases.

A. The Polymorphic Blending Technique

The *PBT* was used for the first time by Fogla in [7] to avoid exploits detection during an attack. The main goal of *PBT* is to perform obfuscation and blend the data in the normal traffic. The polymorphic part of the technique is put in when the obfuscation is applied and it is used for avoiding the detection of *signature-based IDSs (SIDS)*. The blending part instead is used to evade the more advanced *anomaly-based IDSs (AIDS)*. In order to complete these two parts, *PBT* is divided in two phases: the collecting and the blending phases. The preliminary phase of collecting is used to create the traffic profile of the normal traffic. The normal network activities of the infected machine are recorded and analysed with the purpose of creating a traffic profile

to emulate. During this phase, only the most significant network features are recorded and those are exactly the same ones used by the *IDSs* to create their profiles. Afterwards, this profile is used in the blending phase to alter the traffic and to make it as similar as possible to legitimate traffic. The blending manipulates the traffic features as well as the payload of the packets by using byte substitution. The profile contains the bytes distribution of the traffic that is the number of occurrences of each byte within the same connection. After that, *PBT* substitutes the bytes of the target data according to the bytes distribution of the profile as shown in Figure 1. In this way, when this data is sent, the payload statistics remain almost the same as the one recorded in the profile.

FIGURE 1: BYTE SUBSTITUITION



Data Byte Distribution	
Byte	Occurrence
0x20	2785
0x65	1993
0x74	1851
0x69	1302
0x61	1178
0x6f	1089
0x6e	1075
0x73	972
0x72	893
0x63	704
0x68	625
...	...

Profile Byte Distribution	
Byte	Occurrence
0x3	26668
0x17	26261
0xf1	26222
0x40	26175
0x49	26174
0x14	26127
0x86	26113
0xf9	26103
0x50	26099
0x39	26068
0x2	26062
...	...

Byte substitution is a really easy way to obfuscate data and it is also polymorphic since it changes every time according to the collected traffic.

4. UNDETECTABLE EXFILTRATION

Almost all the latest pieces of malware and attacks have data exfiltration capabilities and the information security world is facing big challenges to stop them. As previously said, at the moment data exfiltration is applied by making use of renown classical methods, but they can be detected by properly tuned *IDSs*. In this work, we present a tool that uses *PBT* for data exfiltration. Moreover, we evaluate the tool against the most common and widely used *IDSs* to test what they are able to detect and under which circumstances.

We assume a scenario where we are able to infect a machine inside the private network without detection from the security administrator. Hence, we study the scenario after the infection. We mainly focus the study on the Network Intrusion Detection Systems *(NIDS)* due to the polymorphic blending technique we intend to use, which is specifically designed to evade network-monitoring systems. In this scenario, sensitive data is represented by confidential files stored in the infected machine. Everything with access to the machine has also access to these files. Finally, we assume that our view of the network is consistent with the IDS' view. This means that the tool must have enough permissions to be able to sniff on the local interface. Otherwise it cannot collect the traffic information.

A. The Exfiltration Tool

The goal of the exfiltration tool is to send data from a compromised machine to a remote server, which is outside the compromised network and under the control of the attacker, while avoiding the security measures that may be in place. In this particular case, these security measures are represented by the *IDS*s. The exfiltration tool is composed of two main entities: *the collector and the blender*. The data is sent to an external part of the tool called *BlackHole*. It identifies the drop-zone of the malware, which is where all the exfiltrated data is collected. The structural design of the tool is represented in Figure 2.

The collector collects network traffic by sniffing on the local network interface and it stores the statistics on a shared *statistic table*. The collected features are described in Figure 3 and they are divided by the three different layers of extraction such as *transport, payload* and *host*. The general tag describes whether the feature is common for every connection or depending on the single one.

FIGURE 2: TOOL ARCHITECTURE

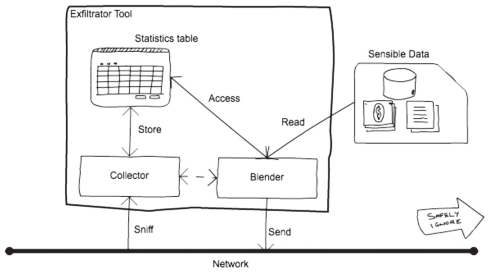


FIGURE 3: TRAFFIC FEATURES

Layer	General	Name
Transport	✓	Rate of Connection Established (*RCE*)
	✓	Range of hours of activities (*TimeFrame*)
	✓	Frequency of source ports
	✓	Frequency of destination ports
	✓	Frequency of destination ports
	✓	Inter Packet Delay (*IPD*)
		Protocol Name
		Bandwidth
		ID Connection (Source *IP* - Source Port - Destination *IP* - Destination Port)
Payload		Packets size
		Entropy
		Byte Value Distribution
Host	✓	*CPU* Load
	✓	Memory Load

The statistic table contains data that is continuously updated as well as features for completed connections. For every new connection, the collector stores the statistics and the byte distribution of the single connection to a temporary table and only when the connection is closed this data is moved to the statistics table. The statistic table is implemented as a hash table and it represents the network profile in the tool.

The blender is in charge of the real exfiltration. It sends out sensitive data shaping the traffic according to the normal profile calculated by the collector. Observing the statistics, it checks whether the right conditions for the exfiltration are satisfied: for example, it controls if the time frame allows new connections or if the workload of the infected machine is below a suspicious threshold. If the number of closed connections stored in the statistic table is above a certain threshold, we can start the exfiltration otherwise we wait. It selects a connection from the statistics table that has the entropy as close as possible to the file to exfiltrate, and then it starts to exfiltrate. The blender supports three different obfuscation methods: *XOR, Cesar13*[1], or byte substitution. For the first two methods, the blender applies basic obfuscation without extra traffic manipulation and they are used only for the sake of the tests. The first packet the blender sends to the server is the conversion table which is the table used to deobfuscate the following packets. The conversion table contains the type of obfuscation used: in case of *XOR*, it contains only the obfuscation key, on the contrary in case of byte substitution, it contains the decryption table which is the reversed byte mapping table to be used for the deobfuscation. The byte mapping table or encryption table is created by combining the selected connection with the file byte distribution, so that the most used byte in the file is mapped with the most used byte in the connection. The blender sends the sensitive data in chunks, each of them obfuscated and respecting the traffic statistics of the selected connection. In fact, the blender also sends the packets with the same bandwidth and the same packet size of the connection, so as it is able to exfiltrate data by imitating the recorded traffic.

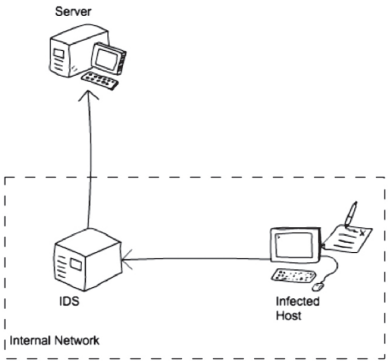
The *BlackHole* is the external entity used by the tool to drop the exfiltrated file. It runs one *TCP* and one *UDP* server and it waits for a transmission to begin. It just needs the first packet with the deobfuscation information in order to perform its task, which is to receive and reconstruct the file.

B. Test Environment

We tested our tool against the following most used *IDSs* divided per type: as signature-based *IDS* we used *Snort (Version 2.9.2 IPv6 GRE (Build 78))*[21], as anomaly-based *IDS* we used *SnortAD (Version 2.9.2.3 IPv6 GRE (Build 205) and AnomalyDetection Version 3.1)* [22] and *McPAD* (site version) [8], and finally as hybrid solution we used *Suricata (Vesion 2.0.2)* [23] and *Bro (Version 2.3-124)* [24]. These were used with the default configurations. I only added the signatures of the sensitive files for Snort and Suricata. They were created by using the first bytes of each sensitive file, which identify the type. For the *anomaly-based IDS*, the profile was created by using from 5 to 10 days of traffic recording. It was normal traffic recorded during a working day then repeated many times. This is the same traffic used to train the Collector.

1 The classic *Cesar13* (or *Rot13*) encryption is limited to alphabet characters so in our work we implemented an extended version which uses all the 256 UNICODE characters. In the whole paper this extended version will be referred as *Cesar13*.

FIGURE 4: EXFILTRATION ENVIRONMENT



The test was conducted in an artificial environment as represented in Figure 4. The *IDS* in figure is configured as network gateway, so all the traffic coming from the client goes through the *IDS*. It listens to the internal interface sniffing all the traffic. The client using *tcpreplay* replayed the recorded traffic, which was used to train the *IDS* as well as during the execution of the tool.

C. Evaluation

During the evaluation, we wanted to test the detection capabilities of different *IDSs* against exfiltration and especially exfiltration using *PBT*. For this purpose, we selected different types of files to act as sensitive files in order to be more realistic in our tests. They are listed in Figure 6 along with their entropy.

FIGURE 5: EXFILTRATION TIME PER SENSITIVE FILE USING PBT

Type	Size (Byte)	Exfiltration time	Normal Transfer
pptx	3.6M	6.81 min	1.0 sec
jpg	98K	10.98 sec	0.3 sec
pdf	1.7M	3.1 min	0.8 sec
zip	210K	3.6 min	0.4 sec
mp3	3.4M	6.43 min	0.9 sec
exe	2.8M	40.56 min	0.9 sec
txt	19K	16 sec	0.1 sec
iso	911M	30.11 hours	90.4 sec

FIGURE 6: SENSITIVE FILES

Type	Size (Byte)	Entropy	Entropy Scale
pptx	3.6M	0.99911088065	HIGH
jpg	98K	0.997354080948	HIGH
pdf	1.7M	0.996589737052	HIGH
zip	210K	0.985803020077	MEDIUM
mp3	3.4M	0.982546526443	MEDIUM
exe	2.8M	0.873447456656	LOW
txt	19K	0.723637261467	LOW

During the tests, we exfiltrated all types of files using all the three obfuscation techniques. *XOR* and *Cesar13* do not perform any traffic manipulation, it is only implemented by *PBT*. We were

able to exfiltrate our testing sensitive files within one hour time. We also tested the exfiltration of a larger file (around *911Mb*) taking around *30* hours. Figure 5 shows the detailed exfiltration time per file, even though it is important to emphasize that these times are strongly dependent on the traffic profile and on the connection selected in the profile. These times are only related to *PBT* exfiltrations. We wanted to test the ability of an *IDS* to detect an exfiltration, so our results are expressed in terms of success (✓) when the *IDS* raised an alert in front of an exfiltration, failure (×) otherwise.

FIGURE 7: EXFILTRATION WITHOUT DETECTION WITH SNORT, SURICATA, AND BRO.

Snort/Suricata/Bro

		pptx	*pdf*	*jpg*	*zip*	*exe*	*mp3*	*txt*
No Obfuscation	TCP	*X*	*X*	*X*	*X*	*X*	*X*	*X*
	UDP	*X*	*X*	*X*	*X*	*X*	*X*	*X*
PBT	TCP	✓	✓	✓	✓	✓	✓	✓
	UDP	✓	✓	✓	✓	✓	✓	✓
XOR	TCP	✓	✓	✓	✓	✓	✓	✓
	UDP	✓	✓	✓	✓	✓	✓	✓
Cesar13	TCP	✓	✓	✓	✓	✓	✓	✓
	UDP	✓	✓	✓	✓	✓	✓	✓

FIGURE 8: EXFILTRATION WITHOUT DETECTION WITH SNORTAD.

SnortAD

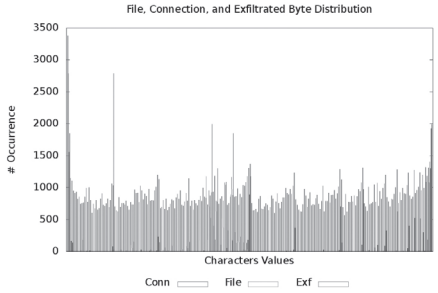
		pptx	*pdf*	*jpg*	*zip*	*exe*	*mp3*	*txt*
No Obfuscation	TCP	*X*	*X*	*X*	*X*	*X*	*X*	*X*
	UDP	*X*	*X*	*X*	*X*	*X*	*X*	*X*
PBT	TCP	✓	✓	✓	✓	✓	✓	✓
	UDP	✓	✓	✓	✓	✓	✓	✓
XOR	TCP	*X*	*X*	*X*	*X*	*X*	*X*	*X*
	UDP	*X*	*X*	*X*	*X*	*X*	*X*	*X*
Cesar13	TCP	*X*	*X*	*X*	*X*	*X*	*X*	*X*
	UDP	*X*	*X*	*X*	*X*	*X*	*X*	*X*

FIGURE 9: EXFILTRATION WITHOUT DETECTION WITH MCPAD.

McPAD

		pptx	*pdf*	*jpg*	*zip*	*exe*	*mp3*	*txt*
No Obfuscation	TCP	✓	✓	✓	✓	✓	✓	✓
	UDP	✓	✓	✓	✓	✓	✓	✓
PBT	TCP	✓	✓	✓	✓	✓	✓	✓
	UDP	✓	✓	✓	✓	✓	✓	✓
XOR	TCP	✓	✓	✓	✓	✓	✓	✓
	UDP	✓	✓	✓	✓	✓	✓	✓
Cesar13	TCP	✓	✓	✓	✓	✓	✓	✓
	UDP	✓	✓	✓	✓	✓	✓	✓

FIGURE 10: BYTE DISTRIBUTION WITH EXFILTRATTED TRAFFIC.



Snort and *Suricata* were able to detect only not obfuscated exfiltrations as shown in Figure 7 since, as it could be expected, signature-based *IDSs* failed against any form of obfuscation. Even with *Bro* we had the same results (Figure 7). In fact, we could not find any feature in the exfiltration that could stand out such that they could be described in its very advanced scripting language.

SnortAD increases the rate of detected exfiltrations as shown in Figure 8. Since the obfuscation using *XOR* and *Cesar13* does not apply any kind of blending, they are easily spotted by the *IDS*. In fact, when blending is applied, beside the payload other traffic features are modified by changing the shape of the traffic. *SnortAD* is able to detect every exfiltrations we tested, except those using *PBT*. With *PBT*, we calculated the byte distribution and we applied byte substitution according to it. The exfiltrated traffic byte distribution is shown in Figure 10 compared with the ones of the connection and the file. As we can see, the exfiltrated data (blue bars) has the same byte distribution of the connection (red bars) but with the same number of bytes occurrences of the file (green bars). The tests performed on *SnortAD* were also able to select the most effective traffic feature that permits the detection: the bandwidth. In fact, by tweaking the bandwidth we were able to fix *30%* as the threshold within which the bandwidth can fluctuate without producing detection.

All the *IDSs* performed as expected with the exception of *McPAD*. It produced no detection in all the tests performed as shown in Figure 9. It has been fed with the same traffic profile we provided to *SnortAD* but in this case it was not useful. The reason for the *McPAD* behaviour is that it has too narrowed capabilities. Even though there are configurable parameters, they do not allow to specify any extra rules able to detect our traffic. We placed this *IDS* with the same condition of all the others but it was the only one that did not produce any detection and so we can conclude that it is not fit for the task. In fact, our technique removes the invariant of the decryption code always present in few bytes of the packets of the PBA, those specific bytes that help *McPAD* for detection.

5. COUNTERMEASURE

We successfully proved that *PBT* is able to exfiltrate sensitive data, while avoiding detection of the most common and widely used *IDSs*. Unfortunately, this technique is not bullet proof: by improving the *IDS* detection engine we can be able to detect also our technique. For example, we can improve the anomaly-based *IDS* combining the information of the number of packets in a time frame with the concept of flow. For example, when the *IDS* collects the statistics about the normal traffic we can detect the average, the max, and the min of the data exchanged within the same flow (*TCP* or *UDP*), and the number of time frame used by an active flow. By using this information, our exfiltration can be detected because of three reasons: the flow goes over a long time, the flow is always active, and the amount of data is of considerable size for a single flow. Moreover, we have an exposed point in our exfiltration: the first packet. We transmit crucial information for the exfiltration in the first packet and we do it in clear. When using byte substitution, this packet contains the decryption table that, in most cases, it is *512* bytes big. Consequently, we can create a signature of such packet for the *IDS* so that it can be able to

detect the exfiltration from the beginning. Unfortunately, even though it can be a useful alert, this signature is too generic and it can produce too many false positive. In fact, we cannot make any assumption about the content of the packet since it changes completely every time, and it depends on both the selected connection and the file to be exfiltrated.

Such is the well-known discussion always present in security: the *Achilles and the Tortoise* paradox of security. We are witnesses of defenders and attackers running after each other. Even in this case this scenario applies perfectly. Improving the *IDS* with the previously described features does not stop the exfiltration to adjust the shot. The improved *IDS* can be easily evaded by our exfiltration by using multiple connections for a single exfiltration or by using multiple drop zones. In this way, the single flow is shorter and it is transmitting a smaller amount of data. The exfiltration can also be spread over a longer time just by stopping and restarting the connection in different time frames. Lastly, the exfiltration can also be improved by removing the single of point of failure of the first unobfuscated packet. Since it is relatively small (only *512* bytes), it can be transmitted using back channels such as *DNS tunnelling* without being suspicious and attracting any attention.

While all these improvements can be considered reasonable, they have not been tested yet and we consider this as part of future works.

As we can see, the *PBT* results a really effective technique due to its capacity of imitating whatever is considered legitimate traffic, while it makes the detection by *IDSs* almost impossible.

6. CONCLUSION

Data leakage represents a major threat for companies and states, so the attention is moving more and more on studying exfiltration technique. Gradually, the crucial aspect is to find solutions able to detect the exfiltrations. Signature-based *IDSs* can detect exfiltrations if they do not use any kind of obfuscation. As soon as even the most classical obfuscation method is used, *SIDSs* are not able to detect them anymore. Only the more advanced anomaly-based *IDSs* can be up to the task. Those are able to detect exfiltrations that use normal obfuscation methods.

In this work, we exfiltrated data by using the more advanced Polymorphic Blending Technique in order to avoid detection. We implemented this technique along with other obfuscation method in a tool capable to exfiltrate sensitive data from an infected machine to a server controlled by the attacker.

The tool is used to test the detection capabilities of different type of *IDSs* against exfiltration. Using the *PBT*, the tool is able to successfully exfiltrate any type of file evading our selection of the most used *IDSs*. Even the more advanced anomaly-based *IDSs* cannot detect such type of exfiltration.

Finally, we calculated 30% as exfiltration threshold within which the tool can perform a safe exfiltration exploiting the *IDSs* tolerance.

7. FUTURE WORKS

The tool we have implemented represents a proof of concept used to test the detection capabilities of the *IDSs*. It implements the really basic functionalities necessary to perform exfiltration and it can be extended with many more capabilities. For example, it can implement the detection countermeasures described in Chapter 5. Moreover, the tool applies only manipulation of the traffic features but it does not use any other evasion techniques [25] such as fragmentation or session manipulation. This could be a different front where to test the *IDSs*.

Finally, we limited our tests to a subset of *IDSs* but it would be interesting to test the tool against other *IDSs* such as *ExFILD* [15].

REFERENCES

[1] R. Van Antwerp, "Exfiltration techniques: An examination and emulation" PhD Thesis, University of Delaware, 2011.

[2] S. Axelsson, "Intrusion Detection Systems : A Survey and Taxonomy" Technical Report 99-15, Department of Computer Engineering, Chalmers University, March 2000.

[3] A. Giani, V. H. Berk, and G. V Cybenko, "Data exfiltration and covert channels," in Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense V. Edited by Carapezza, Edward M. Proceedings of the SPIE, 2006, vol. 6201, p. 620103.

[4] J. Andress and S. Winterfeld, Cyber Warfare: Techniques, Tactics and Tools for Security Practitioners, 1st ed. Syngress Publishing, 2011.

[5] G Data SecurityLabs, "Uroburos Highly complex espionage software with Russian roots" Technical Report G Data SecurityLabs, 2014.

[6] F-secure Labs Security Responce, "COSMICDUKE: Cosmu with a twist of MiniDuke" Technical Report F-secure Labs, 2014.

[7] P. Fogla, M. Sharif, R. Perdisci, O. Kolesnikov, and W. Lee, "Polymorphic blending attacks" in Proceedings of the 15th USENIX Security Symposium, 2006, pp. 241–256.

[8] R. Perdisci, D. Ariu, P. Fogla, G. Giacinto, and W. Lee, "McPAD: A multiple classifier system for accurate payload-based anomaly detection" Comput. Networks, vol. 53, no. 6, pp. 864–881, 2009.

[9] I. I. Amit, V. P. Consulting, and S. Art, "Advanced Data Exfiltration – the way Q would have done it" GovcertNT, Rotterdam, Netherlands, 2011.

[10] S. Wendzel, W. Mazurczyk, L. Caviglione, and M. Meier, "Hidden and Uncontrolled-On the Emergence of Network Steganographic Threats" arXiv Prepr. arXiv1407.2029, 2014.

[11] S. Wendzel, "Novel Approaches for Network Covert Storage Channels" PhD Thesis, Fernuniversität Hagen, 2013.

[12] F. V. Yarochkin, S. Y. Dai, C. H. Lin, Y. Huang, and S. Y. Kuo, "Towards adaptive covert communication system" Proc. 14th IEEE Pacific Rim Int. Symp. Dependable Comput. PRDC 2008, pp. 153–159, 2008.

[13] S. Khattak, M. Javed, P. D. Anderson, and V. Paxson, "Towards Illuminating a Censorship Monitor's Model to Facilitate Evasion," Presented as part of the 3rd USENIX Workshop on Free and Open Communications on the Internet, Berkeley, CA, pp. 1–7.

[14] A. Houmansadr, C. Brubaker, and V. Shmatikov, "The parrot is dead: Observing unobservable network communications" in Security and Privacy (SP), 2013 IEEE Symposium on, 2013, pp. 65–79.

[15] T. Fawcett, "ExFILD: A tool for the detection of data exfiltration using entropy and encryption characteristics of network traffic" PhD Thesis, University of Delaware, 2010.

[16] D. Bolzoni, B. Crispo, and S. Etalle, "ATLANTIDES: Automatic Configuration for Alert Verification in Network Intrusion Detection Systems" In: LISA '07: Proc. 21th Large Installation System Administration Conference, USENIX Association (2007) 141–152.

[17] B. Guttman and E. A. Roback, An introduction to computer security: the NIST handbook. DIANE Publishing, 1995.

[18] A. Rashid, R. Ramdhany, M. Edwards, S. M. Kibirige, A. Babar, D. Hutchison, R. Chitchyan, "Detecting and Preventing Data Exfiltration" Technical Report, Academic Centre of Excellence in Cyber Security Research, 2014.

[19] M. Van Horenbeeck, “Deception on the network: thinking differently about covert channels” School of Computer and Information Science, Edith Cowan University, Perth, Western Australia, 2006.
[20] W. Mazurczyk, “VoIP steganography and its Detection—A survey,” ACM Computer Survey, vol. 46, no. 2, p. 20, 2013.
[21] M. Roesch and others, “Snort: Lightweight Intrusion Detection for Networks” in LISA, 1999, vol. 99, pp. 229–238.
[22] M. Szmit, R. Wezyk, M. Skowro’nski, and A. Szmit, “Traffic Anomaly Detection with Snort” Information Systems Architecture and Technology. Information Systems and Computer Communication Networks, Wydawnictwo Politechniki Wrocławskiej, Wrocław, pp. 181–187, 2007.
[23] D. Day and B. Burns, “A performance analysis of snort and suricata network intrusion detection and prevention engines” in ICDS 2011, The Fifth International Conference on Digital Society, 2011, pp. 187–192.
[24] V. Paxson, “Bro: a System for Detecting Network Intruders in Real-Time” Computer Networks, vol. 31, no. 23–24, pp. 2435–2463, 1999.
[25] K. Timm, “Ids evasion techniques and tactics” SecurityFocus (Infocus), vol. 7, 2002.