Detection of Botnet Command and Control Traffic by the Identification of Untrusted Destinations

Pieter Burghouwt(✉), Marcel Spruit, and Henk Sips

Parallel and Distributed Systems Group, Delft University of Technology,
Mekelweg 4, 2628CD Delft, The Netherlands
{P.Burghouwt,H.J.Sips}@tudelft.nl, M.E.M.Spruit@hhs.nl

Abstract. We present a novel anomaly-based detection approach capable of detecting botnet Command and Control traffic in an enterprise network by estimating the trustworthiness of the traffic destinations. A traffic flow is classified as anomalous if its destination identifier does not origin from: human input, prior traffic from a trusted destination, or a defined set of legitimate applications. This allows for real-time detection of diverse types of Command and Control traffic. The detection approach and its accuracy are evaluated by experiments in a controlled environment.

Keywords: Botnets · Network intrusion detection · Anomaly detection

1 Introduction

In this paper we present a new approach to detect botnet C&C(Command and Control) traffic in an enterprise network. With the term enterprise network we refer to a computer network that is exclusively used by an organization under one common administration. Passive network-based detection of botnet traffic is an attractive defense layer against botnets because of its low risk of compromise. A basic approach is misuse detection, based on knowledge of malicious traffic, such as signatures [14]. However, the dependency on knowledge of specific botnets, makes it ineffective against new types of C&C communication. Anomaly detection addresses this problem by observing deviations from normal traffic. Detection by DNS anomalies is a popular approach, but obviously limited to bots that use DNS in their C&C communication. Correlation-based approaches can detect a broader range of C&C traffic, however they require multiple malicious traffic instances for detection [7].

In contrast, our approach is capable of real-time detection of a broad range of C&C traffic by just a single traffic instance. It is based on trust of traffic destinations. Trust is a complex concept and can be defined in many different ways. We use a context-specific definition of trust, derived from the more generic definition of Olmedilla et al. [13]. In our context, which is an enterprise network

J. Tian et al. (Eds.): SecureComm 2014, Part I, LNICST 152, pp. 174–182, 2015.
DOI: 10.1007/978-3-319-23829-6_13

with potentially bot-recruited computers, we define trust as *the measurable belief of an organization that a specific entity does not collude in a botnet.* We assume that the organization trusts its employees and a defined set of legitimate software applications if deployed on an uninfected computer. On the other hand, the enterprise computers including the installed OS and software instances, are not trusted, since they can become compromised and recruited in a botnet. Traffic destinations are initially not trusted, because they can be part of a C&C infrastructure that is contacted by an inside bot. However, a destination becomes trusted by transitivity, if its *identifier* origins from another trusted entity. The *identifier* of a destination can be an IP-address, name, URI, or any other data that is used to direct the traffic to a remote computer or resource.

Evaluation of the origin of *destination identifiers* enables the detection of C&C traffic. Traffic is classified as normal, if the destination identifier origins directly from: human input, a legitimate application, or the received content from a trusted destination. All other destination identifiers are not trusted and the associated traffic is classified as anomalous.

We will refer to this anomaly detection approach as *Untrusted Destination by Identifier Detection* or *UDI Detection*. Section 2 describes the details of UDI detection. Section 3 evaluates UDI detection by experiments with real traffic. Section 4 elaborates evasion possibilities. UDI detection is compared with other work in Sect. 5. Finally Sect. 6 concludes and proposes future work.

2 UDI Detection Approach

We assume the typical scenario of client computers in a segment of an enterprise network, protected by a stateful firewall. This enforces inside bots as the initiator of C&C communication(*phone home*). All traffic is passively captured by the UDI detector and organized in traffic flows. The detector evaluates the egress flows on trust of their destinations. An egress flow is only classified as normal if its destination is trusted. Ingress flows inherit the trust and anomaly state of the associated egress flow.

For each new egress flow, trust is determined by its *destination identifier* in three consecutive stages as shown in Fig. 1.

The first stage tests the presence of the destination identifier in a predefined set of legitimate destinations, used by trusted applications. This typically includes destinations of servers for software updates, browser home pages, and local management traffic. Flows to these destinations are classified as normal and not further evaluated.

The second stage tests if the destination identifier matches a reference that was received in the payload of a prior ingress flow from a trusted destination. Reference examples are URL's in HTTP content and IP-addresses in DNS replies. If the destination identifier matches a reference, the destination is trusted, and the associated flow is classified as normal and not further evaluated.

The third stage evaluates the remaining destination identifiers on the likelihood of being directly entered by a human. We assume that humans normally

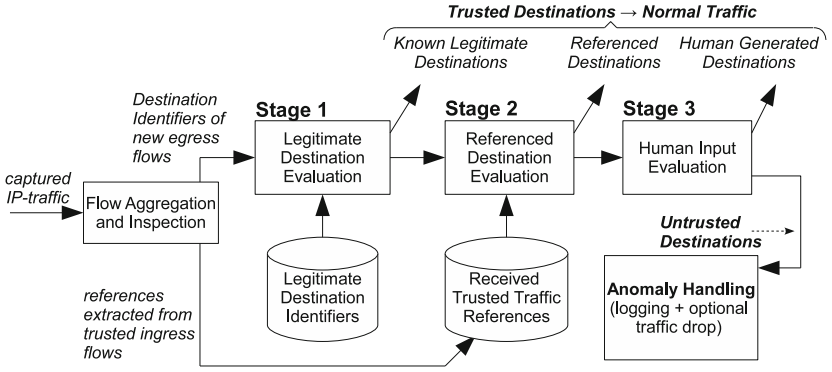

Fig. 1. Schematic overview of UDI detection.

enter destination identifiers that can be distinguished by their relatively low complexity and high predictability.

The remaining destination identifiers represent untrusted destinations that belong to flows that are likely automatically generated by illegal processes. The combination of the three stages results in a system that can immediately detect botnet phone home traffic, even if it has a low volume and uses popular traffic types, to stay below the radar of existing Intrusion Detection Systems. The passive traffic monitoring and real-time classification of UDI detection, allow for implementation in an edge-router, or a network Intrusion Prevention System, to prevent any contact between an inside bot and outside C&C entities. The necessary deep packet inspection of traffic payloads and the management of a set of known trusted legitimate destinations, are especially feasible in enterprise networks.

2.1 Logical Destination Identifiers and Forward References

Before further elaborating UDI detection, we introduce the *ldi* (logical destination identifier) of an egress flow X, defined by Eq. 1.

$$ldi_X = (host\text{-}id_X, resource\text{-}id_X) \tag{1}$$

The *host-id* identifies the contacted remote host of flow X. It is directly represented by the remote IP address or a hostname, as defined by Eq. 2.

$$host\text{-}id_X = \begin{cases} hostname(IP_{dest,X}) & if\ hostname(IP_{dest,X}) \neq null \\ IP_{dest,X} & if\ hostname(IP_{dest,X}) = null \end{cases} \tag{2}$$

$IP_{dest,X}$ is the destination address of the egress flow. If this address is the result of a DNS lookup, the hostname is obtained from a cache by *hostname()*.

The *resource-id* in Eq. 1 identifies a specific resource of the remote host and is extracted from the payload of the egress flow. An example of a resource-id is

the *path/querystring*, used in a HTTP GET request. In this particular example the complete *ldi* is very similar to a URI. A completely different example is an ICMP flow of a ping. In this case the *resource-id* in the *ldi* is empty.

In case of a DNS lookup, the *ldi* of the associated egress DNS flow is defined by the hostname that must be resolved instead of the IP-address of the involved DNS server (Eq. 3). This allows for immediate detection during a DNS question stage of C&C communication.

$$ldi_X = query_X \qquad if\ X = DNS\ flow \tag{3}$$

We define a *forward reference* as a data element in the payload of an ingress flow that can be used as the *ldi* of a future flow. It can range from a URL in a HTTP hyperlink to an IP-address in a DNS A-record. For UDI detection all forward references in the payloads of ingress are stored in a list of trusted references. Obviously, if the ingress flow is associated with an egress flow that was classified as anomalous by an untrusted ldi, the forward references are not stored. The size of the list remains limited by a maximum allowed validity time of forward references, defined by cache properties of the observed computers (Fig. 2).

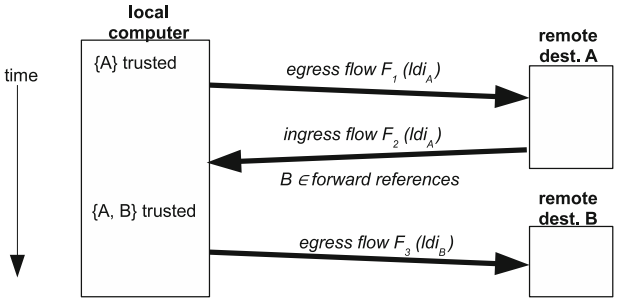

Fig. 2. The remote destination B of egress flow F_3, identified by ldi_B, is trusted because it was referenced in a prior ingress flow F_2 of trusted destination A.

2.2 The UDI Detection Algorithm

The three stages of Fig. 1 identify *ldi's* of trusted destinations. After the three stages, the remaining *ldi's* represent destinations that are not trusted and their associated flows are classified as anomalous. Algorithm 1 shows the complete detection procedure.

- *isEgress(X)* is true if X is an egress flow
- *IdentifyDestination(X)* extracts the *ldi* from egress flow X according to Eq. 1 or 3.
- *isLegitimate()*, *isReferenced(ldi)*, *isUserSubmitted()* are the tests of the three consecutive stages of Fig. 1.
- *getStatusofAssociatedFlow(X)* is NORMAL or ANOMALOUS, depending on the state of the associated egress flow of ingress flow(X).

Algorithm 1. UDI detection algorithm

```
for each new flow X do
  if isEgress(X) then
    ldi = identifyDestination(X);
    if isLegitimate(ldi) or isReferenced(ldi) or isUserSubmitted(ldi) then
      X.Status = NORMAL;
    else
      X.Status = ANOMALOUS;
      signalAnomaly(X);
    end if
  else
    X.Status = getStatusOfAssociatedFlow(X);
    if X.Status = NORMAL then
      extractForwardReferences(X);
    end if
  end if
end for
```

– *extractForwardReferences(X)* will extract and store forward references of flow X.

The string matching process in all three stages can be significantly simplified by truncation of the *ldi*, typically by excluding the resource-id and reducing the hostname to the second level domain name. We will refer to this as *partial ldi matching*. It will reduce the size of the list of trusted destinations and simplify the extraction of both *ldi* and forward refences. A disadvantage of partial *ldi* matching is the increase of False Negatives, caused by accidental matches of malicious *ldi's* with trusted *ldi's*. This will be discussed in Sect. 4.

3 Experimental Evaluation

We constructed a basic UDI detector as a proof of concept and evaluated its accuracy in experiments with real traffic. We implemented the UDI detection algorithm in C++ on a X86-64 PC with a Linux OS. It was inserted as a bridge in a LAN. In addition to real-time detection, traffic was captured in pcap format for offline evaluation by the UDI detector. To limit the complexity of payload parsing, only DNS and HTTP payloads were inspected for forward references. Partial *ldi* matching was implemented, by excluding the resource-id of Eq. 1 and truncating DNS hostnames to the second level domain name.

We derived simple name-based features from [2,3], an [10], resulting in *isUserSubmitted()*=TRUE if three conditions are met:

1. number of characters $\preceq$ C
2. number of non-letter characters $\preceq$ N
3. top level domain $\in$ {set of popular human-input TLD's (region dependend)}

We evaluated the accuracy of UDI detection with traces of both normal traffic and malicious C&C-traffic in a controlled environment that allowed for testing with a wide variety of legitimate and botnet traffic.

3.1 Evaluation of False Positives

In the first experiment we evaluated False Positives by traffic of 40 selected cases of preinstalled applications and web applications, all commonly used by students of our university, such as: the use of email, popular social media, Google Maps, the planning of a journey by Dutch public transport, WhatsApp, games and downloading. Depending on the case, the traffic was produced by a Windows 7, Linux, or Android device. Although corporate traffic is expected to be less diverse, we chose for this selection, to test the detector under difficult conditions.

The parameters of the function *isUsersubmitted()* were chosen: C=20, N=3 and a TLD set of {*.com, .org, .net, .nl, .uk, .de, .gov*}. Two particular cases resulted in an excessive number of false positives ($FPR > 0.5$). The first case was a download with Bittorrent. Since our implementation of UDI detection cannot extract the peer IP addresses of encrypted tracker information, all P2P connections were classified as anomalous. The second case was an Android game that continuously connected to different destinations. Since both cases are not representative for corporate usage, they were excluded from further FPR calculation.

The traces of the remaining 38 cases contain 24362 flows with 54 % HTTP, 8 % of HTTPS, 36 % DNS, and 2 % of other traffic. Since all cases were produced with freshly installed software, we assume no C&C traffic. Consequently every flow, classified by the detector as anomalous, is regarded as a False Positive. This resulted in a FPR of 0.0026 (64 False Positives in 24362 flows). Manual inspection revealed that the majority of False Positives was caused by failures of the detector to extract forward references from SSL payloads and complex web scripts. We will also further elaborate this in Sect. 4.

3.2 Evaluation of True Positives

For the analysis of True Positives, five traces with a mixture of normal and C&C traffic were composed. The normal traffic consisted of 8358 traffic flows, generated by the usage of the 30 most popular global websites, derived from rankings, such as Alexa [1]. The C&C traffic consisted of isolated C&C conversations, captured from bots with different types of C&C communication. Ten copies of the same conversation were injected in the normal traffic at equidistant times. We used a self-developed tool that could modify timestamps and ephemeral ports of the injected C&C traffic, to obtain a consistent composition of normal traffic and ten similar C&C conversations. We composed in this way the five traces, each with a different type of C&C traffic. Table 1 shows the number of measured True Positives and the resulting DR (Detection Rate or True Positive Rate).

All injected C&C flows were detected, with the exception of Twebot, because *Twitter.com* is a simple name that could have been entered by a human. In addition *Twitter.com* was also referred by other legitimate traffic. A solution for this problem is proposed in the next Section.

Table 1. Measured FPR and DR of UDI detection with 5 different infected traces.

Trace	C&C type	C&C dialogues	C&C flows	TP	FP	DR	FPR
Top30 + Kelihos [6]	DNS + HTTP	10	40	40	16	1	0.0019
Top30 + Storm [8]	P2P	10	20	20	16	1	0.0019
Top30 + Twebot [12]	Twitter	10	60	0	16	0	0.0019
Top30 + TBOT [5]	TOR	10	20	20	16	1	0.0019
Top30 + Morto [11]	DNS	10	20	20	16	1	0.0019

4 Evasion of UDI Detection and Solutions

If a C&C flow is erroneously classified as trusted by at least one of the three stages, UDI detection is evaded. For the first and second stage of Fig. 1, this is only possible if the adversary can communicate from a trusted destination. It requires control over a trusted destination in addition to the local bot. This makes the evasion effort relatively high in an enterprise environment with a limited number of trusted destinations. Evasion of the third stage is possible by the use of a simple *ldi* that could origin from human input. This also raises problems for the botnet, since *human-friendly* hostnames are often occupied and in case of a takedown, replacement is difficult. Addition of more features and machine learning can result in a more accurate human input classification that can adapt to specific situations.

Unfortunatly the partial *ldi* matching in our proof of concept facilitates evasion, because the *ldi* is not completely evaluated. This was demonstrated in our experiments with Twitter C&C traffic. Although the complete *ldi* of the contacted account was *Twitter.com/tlab32768*, including the timeline of the malicious account, partial *ldi* matching only evaluated the hostname *Twitter.com*, which resulted in a classification as trusted.

The solution is a complete *ldi* match instead of a partial, however, this requires an accurate matching process that can identify all resource-id's and forward references in payloads. SSL/TLS encryption and complex script constructions in web pages complicate the matching process. An SSL/TLS interception proxy with associated public-key certificate on all computers in an organization [9] allows for inspection of the encrypted traffic. Browser emulation in the UDI detector improves the identification of *ldi's* and forward references in complex payloads. The two mentioned techniques enable UDI detection with full *ldi* matching, however, but the accompanying complex and processing-intensive payload analysis requires further research.

5 Related Work

Detection of C&C traffic by flow-based analysis over several consecutive stages that isolate the malicious traffic, is a common approach. Strayer et al. propose

multistage detection for C&C traffic over IRC [15]. Unlike our *UDI* detection, the approach is limited IRC C&C traffic and uses statistical flow-based and topological properties that depend on the presence of multiple infected bots.

The second stage of our UDI detector tests if the *ldi* of a new flow is referenced in prior ingress flows. Zhang et al. propose CR-miner [17], a system that detects malicious automatic traffic, by evaluating traffic dependencies between connections and user events. In contrast to our method CR-miner is implemented in the observed computer itself, to observe user and process properties. This significantly increases the exposure level to potential malware. CR-minor associates flows by the Referer field in the HTTP header. This makes the approach only applicable to HTTP traffic that supports this field. It can be easily manipulated by malware, since it is produced in a potentially infected computer. UDI detection is not sensitive for this type of tampering, because forward references are captured from payloads of ingress flows that origin from other computers and because *ldi's* cannot be manipulated, without changing the egress flow destination.

Burghouwt et al. use causal relationships between flows to detect botnet C&C traffic [4]. Instead of the destination, detection is based on the direct cause of a flow. Unlike *UDI* detection this demands for the accurate measurement of delay between certain events and induced new flows. Another difference is the required monitoring of user events by a software agent or a hardware device.

Whyte et al. present a detector of scanning worms by determining IP-addresses that are not earlier seen in DNS-replies or received HTTP-data [16]. This can be seen as a special case of flow referral, that isolates flows with unreferenced destination IP-addresses, as is often seen with worms.

6 Conclusions and Future Work

UDI detection detects different types of stealth C&C *phone home* communication in an enterprise network by the trustworthiness of contacted destinations. It evaluates the *ldi's* in three different stages. Advantages of UDI detection are: real-time detection of even a single C&C flow, detection of zero-day traffic and a low exposure to malware.

Partial *ldi* matching simplifies the UDI detector implementation. The results of experiments with samples of C&C traffic and normal traffic support the detection approach.

In future work we plan improvement of UDI detection by complete *ldi* matching, to detect also C&C traffic over popular social media. This requires SSL traffic interception, payload parsing by browser emulation, and the selection of more features with an appropriate machine-learning algorithm for a more accurate and adaptive classification of human input.

References

1. Alexa.com: Alexa, the web information company. http://www.alexa.com/topsites. Accessed March 2013
2. Bilge, L., Kirda, E., Kruegel, C., Balduzzi, M.: Exposure: finding malicious domains using passive dns analysis. In: NDSS (2011)
3. Blum, A., Wardman, B., Solorio, T., Warner, G.: Lexical feature based phishing url detection using online learning. In: Proceedings of the 3rd ACM workshop on Artificial intelligence and security, pp. 54–60. ACM (2010)
4. Burghouwt, P., Spruit, M., Sips, H.: Detection of covert botnet command and control channels by causal analysis of traffic flows. In: Wang, G., Ray, I., Feng, D., Rajarajan, M. (eds.) CSS 2013. LNCS, vol. 8300, pp. 117–131. Springer, Heidelberg (2013)
5. Contagio: Skynet tor botnet/trojan.tbot samples. http://contagiodump.blogspot.nl/2012/12/dec-2012-skynet-tor-botnet-trojantbot.html. Accessed February 2014
6. DeependResearch: Trojan nap aka kelihos/hlux. http://www.deependresearch.org/2013/02/trojan-nap-aka-kelihoshlux-feb-2013.html. Accessed February 2013
7. Gu, G.: Correlation-based botnet detection in enterprise networks. ProQuest (2008)
8. Holz, T., Steiner, M., Dahl, F., Biersack, E., Freiling, F.C.: Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm. LEET **8**, 1–9 (2008)
9. Jarmoc, J., Unit, D.S.C.T.: Ssl/tls interception proxies and transitive trust. Black Hat Europe (2012)
10. Ma, J., Saul, L.K., Savage, S., Voelker, G.M.: Beyond blacklists: learning to detect malicious web sites from suspicious urls. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 1245–1254. ACM (2009)
11. Microsoft.com: Worm:win32/morto.a. http://www.microsoft.com/security/portal/threat/encyclopedia/entry.aspx?Name=Worm:Win32/Morto.A, Accessed April 2014
12. Nazario, J.: Twitter-based botnet command channel, August 2009. http://asert.arbornetworks.com/2009/08/twitter-based-botnet-command-channel/. Accessed October 2013
13. Olmedilla, D., Rana, O.F., Matthews, B., Nejdl, W.: Security and trust issues in semantic grids. Semantic Grid 5271 (2005)
14. Roesch, M., et al.: Snort: lightweight intrusion detection for networks. In: LISA, vol. 99, pp. 229–238 (1999)
15. Strayer, W.T., Lapsely, D., Walsh, R., Livadas, C.: Botnet detection based on network behavior. In: Lee, W., Wang, C., Dagon, D. (eds.) Botnet Detection. Advances in Information Security, vol. 36, pp. 1–24. Springer, New York (2008)
16. Whyte, D., Kranakis, E., van Oorschot, P.C.: Dns-based detection of scanning worms in an enterprise network. In: NDSS (2005)
17. Zhang, H., Banick, W., Yao, D., Ramakrishnan, N.: User intention-based traffic dependence analysis for anomaly detection. In: 2012 IEEE Symposium on Security and Privacy Workshops (SPW), pp. 104–112. IEEE (2012)