Detection of Botnet Command and Control Traffic by the Multistage Trust Evaluation of Destination Identifiers [★]

Pieter Burghouwt[1,*], Marcel E.M. Spruit[2], Henk J. Sips[1]

[1]Parallel and Distributed Systems Group, Delft University of Technology, Mekelweg 4, Delft 2628CD, The Netherlands
[2]Research Group Cyber Security and Safety, The Hague University of Applied Sciences, Johanna Westerdijkplein 75, The Hague 2521EN, The Netherlands

Abstract

Network-based detection of botnet Command and Control communication is a difficult task if the traffic has a relatively low volume and if popular protocols, such as HTTP, are used to resemble normal traffic. We present a new network-based detection approach that is capable of detecting this type of Command and Control traffic in an enterprise network by estimating the trustworthiness of the traffic destinations. If the destination identifier of a traffic flow origins directly from: human input, prior traffic from a trusted destination, or a defined set of legitimate applications, the destination is trusted and its associated traffic is classified as normal. Advantages of this approach are: the ability of zero day malicious traffic detection, low exposure to malware by passive host-external traffic monitoring, and the applicability for real-time filtering. Experimental evaluation demonstrates successful detection of diverse types of Command and Control Traffic.

Keywords: Botnets, Network Intrusion Detection, Anomaly Detection

Received on 20 August 2014; accepted on 22 December 2014; published on 05 October 2015

doi:10.4108/eai.5-10-2015.150476

1. Introduction

It is crucial for an organization to identify infected computers in its premises. Infected computers can attack other computers, steal sensitive information and disturb critical production processes. Infected computers are often bot instances that participate in a botnet. In addition to normal traffic, they produce malicious traffic, consisting of occasional connections or *phone home calls* to a C&C (Command and Control) entity on the Internet and optionally they generate attack traffic, such as DDoS and spam.

In this paper we present a new approach to detect botnet activity in an enterprise network. With the term enterprise network we refer to a computer network that is exclusively used by a private or public organization under one common administration.

Network-based detection of botnet traffic is, in addition to host-based solutions, an attractive defense component against botnets because of its low risk of compromise if implemented host-externally and passively. A basic approach is misuse detection, based on definitions of known malicious traffic, such as signatures [1] or blacklists of malicious hosts [2][3]. However, the dependency on prior knowledge of specific botnets, makes it ineffective against new types of C&C communication. Anomaly detection addresses this problem of zero-day C&C-traffic by observing deviations from normal traffic. Detection of C&C traffic by DNS anomalies is a popular approach [4][5], but only effective against a limited group of C&C communication types, because it depends on the presence of observable DNS anomalies. An important group of botnet-specific anomaly detection approaches evaluates correlation between traffic flows [6]. Although

[★] A short version of this paper will be published in the Springer LNICST proceedings of SECURECOMM 2014

[*]Corresponding author. Email: P.Burghouwt@hhs.nl

capable of detecting zero-day traffic, statistical anomaly detection is not successful in all cases. Firstly correlation-based detection cannot be used to block immediately all C&C traffic, because, during the correlation process, a part of the C&C communication will slip through. Secondly traffic correlation can not always be measured in an enterprise network. In case of only one infected computer in the network, correlation in C&C traffic of different bots cannot be measured, unless detection information is collected from multiple enterprise networks. Additionally some bots, in particular bots deployed for espionage, do not produce attack traffic, such as spam, network scans, or DDoS traffic. This reduces the correlation options to only C&C traffic.

In this paper we present a entirely different type of C&C anomaly detection with the capability of immediate detection of a single C&C flow, which can complement existing methods. 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 a more generic definition from Olmedilla et al. [7]. In our context, which is an enterprise network with inside potentially bot-recruited computers, we define trust as *the measurable belief of the 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 with the installed OS and software instances, are not trusted, since they can be 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 presents a practical implementation for experimental evaluation. Section 4 evaluates UDI detection by experiments with real traffic. Section 5 elaborates evasion possibilities. UDI detection is compared with other work in Section 6. Finally Section 7 concludes and proposes future work.

2. UDI Detection Approach

For UDI detection we assume the typical scenario of client computers in a segment of an enterprise network, protected by a stateful firewall, that blocks all traffic that is initiated from outside. This enforces inside bots as the initiator of all C&C communication(*phone home*). All traffic from and towards the inside computers is passively captured and evaluated by the UDI detector as shown in Figure 1. To limit the number of detection decisions, the UDI detector organizes all traffic in flows by protocol, source and destination IP address, and UDP/TCP port numbers. The stateful firewall assures that each ingress flow is associated with exactly one existing egress flow with swapped IP addresses and ports.

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 their associated egress flows.

For each new egress flow, trust is determined by the the origin of its *destination identifier* in the three consecutive decision stages of Figure 1.

The first stage tests if the destination identifier is present 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 prior ingress flows from a trusted destination. Examples of such references 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. If there is no match, the destination identifier is forwarded to the third stage.

The third stage evaluates the remaining destination identifiers on the likelihood of being directly inputted by a human. We assume that humans normally enter destinations that can be distinguished from machine-originated input by differences in complexity and surprisal. For example, humans will normally not type very long names or IP-addresses. These and many other features can be used in heuristics to differentiate between human and machine origin. If the destination identifier is estimated as human input, the destination is trusted and its flow is classified as normal and not further evaluated. The remaining destinations identifiers represent untrusted destinations that belong to flows that are likely automatically generated by illegal processes.

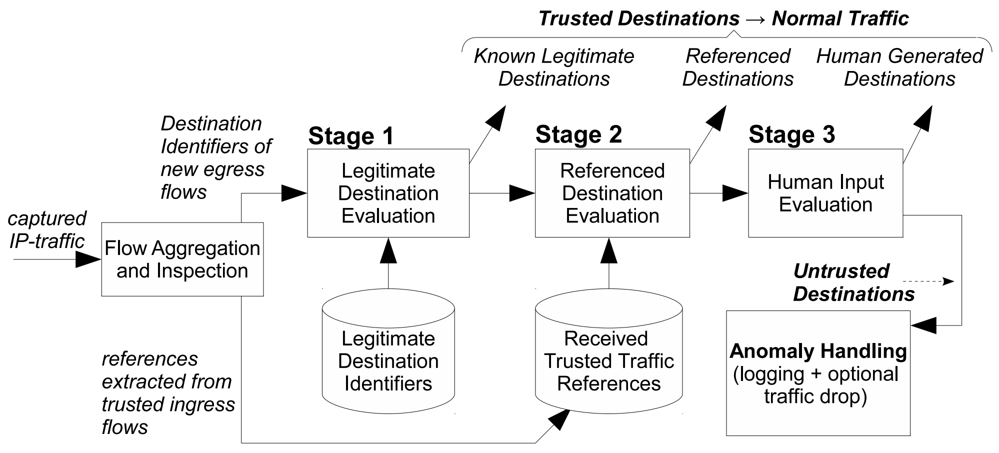
Trusted Destinations → Normal Traffic
Known Legitimate Destinations
Referenced Destinations
Human Generated Destinations
Destination Identifiers of new egress flows
Stage 1
Legitimate Destination Evaluation
Stage 2
Referenced Destination Evaluation
Stage 3
Human Input Evaluation
captured IP-traffic
Flow Aggregation and Inspection
Untrusted Destinations
Anomaly Handling (logging + optional traffic drop)
references extracted from trusted ingress flows
Legitimate Destination Identifiers
Received Trusted Traffic References

Figure 1. Schematic overview of UDI detection.

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 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 all received traffic payloads and the management of a set of known trusted legitimate destinations, are especially feasible in enterprise networks. Deployment in the networks of public ISP's with connected consumer devices and home networks is more difficult, due to the high diversity of consumer end systems, the lack of control and transparency in consumer networks, and privacy regulations.

2.1. Logical Destination Identifiers

Before further elaborating UDI detection, we present a more precise definition of the destination identifier of a flow, and will refer to this as the *logical destination identifier* or *ldi*. We assume a local computer that initiates an egress flow X to a remote destination that is identified by ldi_X, as defined by Equation 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 determined by the destination IP address of the flow as shown in Equation 2. A computer normally acquires a destination IP address by the translation of a hostname, performed by a translation service, in most cases DNS. If the translation is observed before flow X, the host-id will be the hostname. In all other cases it is directly the IP-address.

$$host\text{-}id_X = \begin{cases} hostname(IP_{dest,X}) \\ \quad if\ hostname(IP_{dest,X}) \neq 0 \\ \\ IP_{dest,X} \\ \quad if hostname(IP_{dest,X}) = 0 \end{cases} \tag{2}$$

In this formula $IP_{dest,X}$ is the destination address in the IP-header of egress packets of flow X. The function *hostname()*delivers the IP address, if a valid translation exists from hostname to IP address. Otherwise *hostname()* is 0.

- The *resource-id* in Equation 1 identifies a specific resource of the remote host. It is extracted from the payload of the egress flow. If not present in the payload, the *resource-id* is defined as zero. 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 zero.

The basic assumption of UDI detection is the low probability that a trusted destination belongs to the C&C infrastructure of a local bot-infected computer, or provides *ldi's* of the C&C infrastructure. However this assumption does not hold for trusted destinations that deliver translation services, such as corporate DNS-servers. Malware from infected computers can contact the DNS-server for resolving a hostname of a C&C server. In such a case the trust stateof the DNS-server should not transfer to translated destinations. Therefore the *ldi* of an egress DNS flow is defined by the query sent to the resolver for translation (Equation 3).

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

This means that a DNS flow, towards a DNS-server with a query that refers to an untrusted destination, is classified as anomalous. If an anomalous DNS flow is not immediately blocked, the received IP-addresses in the DNS answer are not placed in the list of trusted received traffic references, despite the fact that they are delivered by a trusted corporate DNS server. DNS is by far the most important protocol that generates resolver flows, but there are other protocols that can be regarded as resolvers or translators, such as the Bittorrent Tracker Protocol, that resolves hashes of file names to IP-addresses of peers.

2.2. Forward Reference Extraction

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. The adjective *forward* is used, to emphasize that the reference refers to the *ldi* of a future flow. It should not be confused with the *Referer* field in an egress HTTP request that refers *back* to the server that delivered the URL of the new request in a prior flow. For UDI detection all potential forward references in received payloads are stored in a list. The size of the list is limited by defining a maximum allowed validity time of unused forward references. In addition the complete list of references, received by one local computer can be cleared after a reboot of that computer.

Forward references are important in UDI detection, because they transfer the trust state from the destination of a prior flow to the destination of a new flow. This is illustrated in Figure 2.

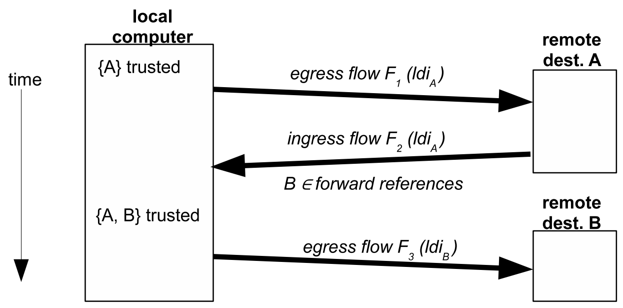
local computer
time
{A} trusted
{A, B} trusted
egress flow F_1 (ldi_A)
ingress flow F_2 (ldi_A)
$B \in$ forward references
egress flow F_3 (ldi_B)
remote dest. A
remote dest. B

Figure 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.

2.3. The UDI Detection Algorithm

The three stages of Figure 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.

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
```

- *isEgress(X)* is true if X is an egress flow
- *IdentifyDestination(X)* extracts the *ldi* from flow X according to equation 1 or equation 3 for respectively non-resolver or resolver flows.
- *isLegitimate()*, *isReferenced(ldi)*, *isUserSubmitted()* are the tests of the three consecutive stages of Figure 1.
- *getStatusofAssociatedFlow(X)* is NORMAL or ANOMALOUS, depending on the state of the associated egress flow of ingress flow(X).
- *extractForwardReferences(X)* will extract and store the forward references from trusted payloads.

2.4. Detection Accuracy

To elaborate the accuracy of UDI detection, we firstly introduce two classifications for *ldi's*.

1. A *malicious ldi* is the *ldi* of a destination that is used by a bot for a connection to its C&C. All other *ldi's* are in this context *non-malicious*.
2. A *trusted ldi* is the *ldi* of a destination that is classified by the UDI detector as trusted. An egress flow with a *trusted ldi* is normal. An egress flows with an *untrusted ldi* is anomalous. An ingress flow inherits the normal or anomalous status from its associated egress flow.

In the ideal situation the UDI Detection algorithm will classify exactly all malicious *ldi's* as untrusted, and all non-malicious *ldi's* as trusted. However practical imperfections of the detector will introduce classification errors, resulting in False Negatives and

False Positives. We treat here the two most important error sources: partial *ldi*-matching and selection of human-input features.

- *Partial* ldi-*matching*: The first and second stage of UDI detection use a list with respectively legitimate destinations and forward references. False Positives and False Negatives are directly related with the accuracy of the lists. An incomplete list increases the probability of false positives. To keep the list of legitimate destinations short and maintainable, it is convenient to use *partial matching*: only a part of the *ldi* has to match for classification as trusted. The match could be limited to the host-id of the *ldi* or even to just the domain-name of the host-id. A problem of partial matching, is the increased probability that malicious *ldi's* are erroneously classified as trusted, because of a partial match.

  The second stage uses a list of forward references, obtained from the payloads of ingress flows. Extracting complete forward references from payloads can be very complex. For example URL's in HTTP are often relative or dynamically composed from different elements in the payload by a client script. If the detector does not fully emulate the involved browser, this leads to missed forward references, that can cause false positives. Partial matching can also in this case reduce the false positives, however with an increased risk of false negatives, as in the first stage.

- *Selection of human-input features*: The heuristics to test the likeliness that the *ldi* is from human input, depends on a proper selection of features. Literature suggests many features to classify anomalous names [8] [9] [10] [11] [12], but non of the proposed feature sets is perfect, with false decisions as a result. A significant advantage of UDI detection is the removal of many non-human *ldi's*, such as references in prior flows, by the two preceding stages.

The complex design choices, make it difficult to derive a simple quantitative predictive model of the FPR (False Positive Rate) and the DR (Detection Rate). Instead we evaluated empirically the behavior of UDI detection and the resulting FPR and DR. Other causes of errors, related with detection evasion, are discussed in Section 5.

3. Detector Implementation

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++ with the usage of the pcap and zlib libraries on a X86-64 PC with 8GB of RAM and a Linux Operating system. Two network interfaces make the system applicable as a bridge in a LAN and allow for real-time inspection with optional removal of bridged traffic by the detector. The bridge can also be configured as a stateful firewall. In addition to real-time detection, the captured traffic can be stored in pcap format for offline evaluation by the UDI detector. The traffic is captured by Gulp [13], a capture tool with a low probability of packet loss. Tables of forward references, *ldi's*, and flows are implemented with hash tables, to speed up the search for existing flows and *ldi's*. Tables, intermediate results, and detection decisions are extensively logged for later manual evaluation.

3.1. Partial *ldi*-matching

To limit the complexity of payload parsing in this proof of concept, only the payloads of DNS and HTTP are inspected for forward references and partial *ldi*-matching is applied, as explained in Section 2.4. The extracted forward references are limited to the host-id part of Equation 1. The *ldi*-matching in the 2nd stage for DNS host names is limited to the TLD and at least 4 characters of the second level domain. If the second level domain is a well-known public suffix, such .co in .co.uk, 4 characters of the third level domain name are also included.

3.2. Name-Based Criteria

For the function *isUserSubmitted()* of Algorithm 1, we derived three name-based features from [10], [12], and [11] to test if an *ldi* origins directly from a human:

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

The result of *isUserSubmitted()* is only true, if all three conditions are true. The optimal value of C, N and the set of popular human-input TLD's depends on the behavior of the local average user. In particular the set of TLD's depends on the nationality of the user and the location of computer. For example in The Netherlands the TLD *.nl* is popular, along with some international TLD's, such as *.com* and *.org*.

4. Experimental Evaluation

The experimental evaluation of UDI detection has two objectives:

1. Determine the accuracy of practical UDI detection with different types of C&C traffic. The False Positive Rate (FPR) is determined by feeding the

UDI detector in a controlled environment with a defined set of real non-malicious traffic. The Detection Rate (DR) is determined by feeding the detector with various types C&C traffic, embedded in non-malicious traffic.

2. Demonstrate that each of the three stages contributes significantly in the reduction of the total FPR. In the case of a non-malicious flow, one of the stages has to classify the *ldi* as trusted. If a stage does not classify an *ldi* as trusted, it is passed to the next stage. The fraction of remaining *ldi's*, passed from a stage x to the next stage, is expressed by H_x in Equation 4;

$$H_x = \frac{\#ldi_{stage\ x,not\ trusted}}{\#ldi_{stage\ x,non\ malicious}} \qquad (4)$$

After the third stage the flows of the remaining untrusted *ldi's* are classified as anomalous. Since we assumed normal traffic, these are the False Positives. Equation 5 expresses the FPR (False Positive Rate).

$$FPR = H_t = H_1.H_2.H_3 \qquad (5)$$

Although the three stages are derived from a coherent model that is based on the origin of *ldi's*, it is difficult to exclude hidden dependencies in sieving properties between the stages. This can result in a stage that does not a have a net contribution in the reduction of untrusted *ldi's*. By changing the sequence of the stages, the ratios per stage H_1, H_2 , and H_3 can change, by dependencies in sieving properties of the proceeding stages, but the overall FPR will not change. If one of the three stages has no net contribution, placement as the third stage will equal the ratio H_3 to 1.

4.1. Controlled Environment

We evaluated False Positive behavior, True Positive behavior, and the dependency between stages, of the UDI detector with traces of both normal traffic and malicious C&C-traffic. All traffic was produced by, and captured from computers in a controlled environment:

- The normal traffic was generated by the use of popular applications and the visits to popular websites from computers with a freshly installed operating system and software.
- The C&C traffic was generated from computers, infected with real well-known botnet malware.

Due to corporate regulations and law, it is difficult in a large enterprise network to capture, store, and analyze traffic with complete payloads for experimental evaluation and review of the detection approach. Evaluation of UDI detection in just a small sample of an enterprise network is feasible, but results in a high risk that the traffic is very homogeneous with a limited number of different destinations. UDI-detection will then produce an optimistic False Positive behavior, caused by the fact that the production of new destinations is relatively low. In addition the probability of infection by various types of bots with sophisticated phone home traffic is small, resulting in unreliable information about the True Positive behavior. By testing in a controlled environment, we could evaluate UDI detection more accurately because:

- By the selection of a wide variety of legitimate applications, including many popular websites that result in abundant traffic to referred destinations, the False Positive behavior is evaluated under difficult conditions. This prevents a to optimistic estimation of the FPR.
- By the selection of different types of botnet traffic, combined with clean legitimate traffic, the malicious part of the traffic is precisely known, resulting in an accurate evaluation of the True Positive detection per type of C&C traffic.

4.2. Evaluation of False Positives and Stage Contribution

For evaluation of the False Positive performance, traffic was generated by 40 selected cases of preinstalled applications and web applications, all commonly used by students of our university. Although some of the applications, used by students, are not expected to be present in a corporate environment, we chose for this selection, to test the detector under difficult conditions by a wide variety of applications. Examples of the cases are: Email with a stand-alone client and a webclient, participation in several social media, usage of Google Maps and Street View, planning of a journey by Dutch public transport, communication by WhatsApp, games and downloading. Depending on the case, the traffic was produced by a Windows 7, Linux, or Android device. The applied list of legitimate *ldi's* was kept as small as possible and consisted only of: the IP-addresses in the same local subnet, the domain names of OCSP servers of well-known certificate authorities, and the domain names of servers that were configured in the installed legitimate applications for automatic updates, home-pages, etc. DNS and browser caches of the evaluated systems were cleared before the experiment, to start synchronized with the UDI detector, as needed for a proper functioning of the referenced destination evaluation.

All collected traces were evaluated by the detector. The parameters of the function *isUsersubmitted()* were chosen: C=20, N=3 and *{.com, .org, .net, .nl, .uk, .de, .gov}* ∈ TLD. Two particular cases resulted in an excessive number of false positives ($FPR > 0.5$). They were isolated from the other traces and further manually examined. 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 peer to peer connections were classified as anomalous. The second case was an Android game that connected continuously to different destinations. Since both cases are not representative for corporate usage, they were excluded from further FPR calculation. We will discuss these types of false positives and possible solutions in Section 5.

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. The FPR was calculated by the fraction of the False Positives in the total number of flows and resulted over all 38 cases in an FPR of 0.0026 (64 False Positives in 24362 flows). Additional analysis revealed that web traffic to Content Delivery Networks and advertisement-related traffic were responsible for the majority of the false positives. The *ldi's* were referred in prior encrypted HTTPS payloads that could not be inspected by our implementation of the UDI detector. Nevertheless the number of False Positives caused by this shortcoming remained relatively low, because HTTPS-objects often share the same domain name as the referring web page. Also many entry pages are not encrypted but contain references of *ldi's* to https-objects. We will also further elaborate this in Section 5.

The individual effect of each of the three *ldi* evaluating stages is evaluated, by placing each particular stage in turn as the last stage and measuring the *ldi* Ratio, H, as defined in Equation 4. The results are shown in Table 1. All stages reduce the net FPR, since all H_3-values

Table 1. Measured *ldi* ratios of the last stage (H_3) and the combined preceding stages ($H_1.H_2$) for different sequences. L=Legitimate *ldi* state, R=Referenced *ldi* stage, U=Human Input stage.

Last Stage (H_3)	H_3	$H_1.H_2$
U	0,28	0,0097
R	0,0038	0,70
L	0,083	0,032

are significantly smaller than 1. This supports our model of the *ldi* origin in UDI detection. Table 1 also demonstrates that the referred *ldi* stage is the largest contributor to the reduction of False Positives, since its *ldi* ratio is the smallest. The small contribution of the name complexity filter, is caused by the fact that only a small number of flows have an *ldi* that is directly typed by the user.

4.3. Evaluation of True Positives

For analysis of True Positives, traces with a mixture of normal traffic and malicious command and control traffic were composed. The malicious traffic consisted of C&C traffic of well-known bot malware. Five botnet instances were selected to cover different types of C&C traffic:

1. HTTP-based C&C by Kelihos [14] with DNS and HTTP-traffic
2. Peer-to-peer-based C&C by Storm [15] with Kademlia-based UDP-traffic
3. Social medium-based C&C by Twebot [16] with HTTP and HTTPS web traffic to a Twitter timeline.
4. TOR-based C&C by TBOT [17] with TOR TCP traffic on port 9001
5. DNS-based C&C by Morto [18] with DNS traffic to *ms.jifr.co.cc*.

The normal traffic was generated by visits to the 30 most popular global websites, derived from Alexa [19] and Google [20]. For each website visit, a typical functionality of the website was used, such as a login, a search, playing a clip, or a product selection. All traffic was captured and collected in one trace from a PC with a fresh Windows 7 installation with a Firefox browser, including popular plugins. The trace contains 8358 flows. Due to the popularity of the websites, the trace is a representative sample of web traffic. In addition popular websites truly test the effectiveness of the implemented referring *ldi* stage, because the received HTML objects contain a massive number of forward references that cause auxiliary flows, such as media, advertisements, and mesh-ups. Missed references can result in false positives.

The applied list of legitimate *ldi's* was limited in this experiment to just the IP-addresses in the same local subnet and the domain names of OCSP servers of well-known certificate authorities.

For each C&C trace, the flows of one representative call-home effort, were manually isolated from captured traffic. With a self-developed tool we injected 10 copies of a C&C traffic sample in the normal traffic. Our tool modified the packet positions and timestamps of 10 injected C&C copies to spread the phone communication equally over the entire observation interval of the 8358 legitimate flows. In addition the tool modified IP-addresses and port numbers of the

Table 2. Measured FPR and DR of UDI detection with 1 clean and 5 infected traces.

trace	phone home calls	malicious flows	TP	FP	DR	FPR
Top30 clean	0	0	0	16	-	0.0019
Top30+Kelihos	10	40	40	16	1	0.0019
Top30+Storm	10	20	20	16	1	0.0019
Top30+Twebot	10	60	0	16	0	0.0019
Top30+TBOT	10	20	20	16	1	0.0019
Top30+Morto	10	20	20	16	1	0.0019

C&C traffic to create one consistent trace with both normal and C&C traffic, originating from the same computer, without conflicts in the used ephemeral TCP and UDP ports. Table 2 shows the number of measured True Positives and the resulting DR of the traces with injected C&C flows. For reference the trace without C&C traffic is also evaluated.

All injected flows of Storm and TBOT are detected because the *ldi's* are not from known trusted applications, unreferenced, and bare IP-addresses. The phone home calls of Kelihos start with a DNS lookup of a hostname in the *.ru* domain. These DNS flows and the traffic to the resolved IP-address are therefore classified as anomalous, resulting in a DR of 1. The injected DNS-only C&C traffic by Morto is detected by the query of *ms.jifr.co.cc*. Similar to the lookup of the *.ru* domain, the *ldi* is not from a known legitimate application, neither from prior trusted traffic, nor from human input. The C&C traffic from Twebot is not detected, because the *ldi* is *Twitter.com*, which is a simple name that could have been entered by a human. Additionally *Twitter.com* is referred by other legitimate traffic. The inability to detect C&C traffic that uses popular hostnames, is caused by the partial *ldi*-matching that excludes the resource-id from the evaluation. A solution for this problem is proposed in the next Section.

5. Evasion of UDI Detection and Solutions

There are several ways for a bot to evade UDI detection. One approach is directly demonstrated in our experiments: by using a popular server as C&C, there is a high probability that the detector will classify the *ldi* as trusted. This is caused by partial *ldi*-matching, as demonstrated in our experiments with Twebot. The complete *ldi* of the C&C in our experiment was *Twitter.com/tlab32768*, including the timeline of a malicious account, but due to partial *ldi*-matching, the malicious resource-id was omitted and only the host-id *Twitter.com* was evaluated. This resulted in classification as trusted, because other objects of *Twitter.com* were already referenced by prior flows and additionally *Twitter.com* can origin from user input by its low complexity. The solution is a complete *ldi* match instead of a partial. This requires two techniques:

1. *encrypted payload inspection* A significant part of modern traffic uses TLS, such as HTTPS. The encrypted payload prevents the extraction of resource-id's from egress flows, and forward references from ingress flows. Complete *ldi*-matching would then result in a large number of false positives. In our experiments this was reduced by partial *ldi*-matching. However a better solution is the insertion of an SSL/TLS-interception proxy. By the installation of a public-key certificate on clients in an organization, a trust relationship can be established between the observed computers and the proxy that enables decryption of TLS-traffic, without certificate warnings of browsers [21]. Resource id's and forward references can now completely be extracted from the decrypted traffic. This allows for complete *ldi*-matching and prevents false negatives by the use of popular hosts. The appliance of an SSL-interception proxy changes the detector from a passive into an active device, since traffic is intercepted, decrypted, and encrypted. Although this theoretically results in more exposure of the detector to bots, it is our belief that it will not increase significantly the risk of compromise.

2. *browser emulation* Modern websites use complex client scripts that can construct URL's by dynamically combining different elements from ingress payloads and even user input. This complicates the complete extraction of *ldi's* and forward references with an increased risk of false positives. As explained, partial *ldi*-matching is a simple solution for this, but introduces evasion possibilities by inaccurate matching. A solution is the extraction of complete forward references in the UDI-detector by emulation of the clients browser.

It is evident that both techniques include complex and processing-intensive payload analysis that requires further research.

UDI detection can also be evaded by completely different techniques:

1. *The botnet controls a trusted destination*. This is for the botnet a complex type of evasion because it has to recruit at least two instances: a local computer as the inside bot and an external host with a trusted destination. Additionally in case of a takedown it would be difficult to replace the C&C destination.

2. *The botnet acquires a hostname that can origin from human input*. This also raises problems for the botnet, since *human-friendly* hostnames are often occupied, and again in case of a takedown, the replacement is difficult. In our experiment we only used three simple static rules to classify *ldi's* from human input. The use of more features and machine learning can result in a more accurate classification that can adapt to specific situations.

3. *The botnet does not phone home*. Nagaraja et al. proposed a botnet that piggybacks C&C messages in pictures that were exchanged between members of social media [22]. In this model there are no phone home connections, which makes it undetectable for UDI detection. However it requires the recruitment of at least two instances that exchange content by a popular social medium. Generally this is a difficult condition to achieve, with again problems in case of a takedown.

6. Related Work

In addition to the work, discussed in Section 1, we start here by discussing some important examples of network-based passive botnet detection.

Gu et al. propose Botminer [23]. BotMiner clusters similar communication flows and hosts that perform similar suspicious activities. Both clusters are cross correlated, to identify potential bots. Although it can detect many types of botnet traffic, it depends on the presence of multiple botnet flows and observable malicious activities.

Rieck et al. propose Botzilla for detection of botnet phone home traffic [24]. Basically it is misuse-based detection, because it uses signatures of repeatedly executed bots. This results in a low FPR and DR if the signature is known and parameterizable. However, unlike our anomaly-based UDI detection, it can only detect C&C traffic with known signatures.

Cocospot detects botnet C&C traffic by classifying traffic flows by protocol, length sequences of packets in a flow, and the encoding of URL's in a HTTP query [25]. Unlike UDI detection the system needs to learn from known C&C traffic, before it can distinguish this traffic from normal traffic. Evasion is possible by noise injection in the packets.

Giroire et al. propose detection of botnet C&C traffic by identifying new repeated combinations of traffic destinations within varying time windows [26]. This type of anomaly detection assumes C&C traffic that connects multiple times to the same destination. Unlike UDI detection it does not relate traffic destinations with prior references or user input. The C&C communication must repeatedly have taken place, before it can be detected by the method of Giroire.

The combination of legitimate destination evaluation, referenced destination evaluation, and human input evaluation, distinguishes UDI detection from the other detection approaches.

6.1. Work Related with Flow Analysis in Consecutive Stages

Detection of C&C traffic by flow-based analysis over several consecutive stages is a common approach. Strayer et al. propose a behavior-based detection system of IRC-based C&C [27]. Like our method they apply several consecutive stages to isolate the malicious traffic, with the first stage as a coarse filter to reduce processing in the other stages. However, unlike our *UDI* detection, the approach focuses on IRC and uses statistical flow-based and topological properties that depend on the presence of multiple infected bots.

Walsh et al. describe a first pass filter that uses simple statistical flow attributes to select flows for further analysis [28]. They only focus on this first filtering stage instead of a complete detector.

6.2. Work Related with Logical Destination Referencing

The second stage of our UDI detector tests if the *ldi* of a new flow is referenced in the received payloads of prior flows. Zhang et al. propose CR-miner [29], a system that evaluates traffic dependencies between connections and user events, to determine malicious automatic traffic. CR-miner associates a new connection with earlier references and user input. In contrast to our method CR-miner is implemented in the observed computer itself, since it needs user and process properties for classification. This significantly increases the exposure level to potential malware. In addition CR-minor uses a different method for associating flows: the Referer field in the HTTP header of a new connection is used to determine if the flow was previously referenced by another flow. This method is only applicable for HTTP traffic that supports this field and it can be easily manipulated by malware, since it is produced by an application in a potentially infected computer. Our method 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 [30]. Instead of the destination, the direct cause of a flow determines if communication is initiated by malware. Unlike *UDI* detection this demands for the accurate measurement of the 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 [31]. 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.

6.3. Work Related with human-input evaluation

Several name-based properties of hostnames and URL's have been proposed, to detect malicious destinations. Since there is not a unique name-based property that can decisively classify anomalous names, the proposed techniques use multiple lexographical and non-lexographical features, often combined with machine learning. Alphanumerical frequencies are used in work of Yadav et al. [8] and Mc Grath et al. [9]. Length of hostnames and substrings is used in work of Mc Grath [9]. et al. and Bilge et al. [10]. The number of dots and other delimiters in URL's are used by Ma et al. [11] and Blum et al. [12]. Our human-input evaluation is different in two ways from other approaches. Firstly the human-input classification in UDI detection is preceded by a stage that removes all traffic with referred destinations. This reduces the FPR. Secondly most name-evaluating detectors produce a list of malicious names for blacklisting. UDI-detection can classify in real-time and allows for an immediate drop of an anomalous flow.

7. 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. The destinations of egress traffic flows are classified as trusted or untrusted by three consecutive stages that evaluate the origin of the involved *ldi* (logical destination identifier). The detector can complement existing network-based anomaly detection approaches, because it works in a different way and it can classify C&C traffic in real-time.

Partial *ldi* matching allows for a relatively simple and feasible UDI-detector implementation. The results of experiments with C&C traffic of real bots and normal traffic support the detection approach with a low FPR and an accurate detection of various types of C&C traffic.

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 and an appropriate machine-learning algorithm for a more accurate and adaptive classification of human input.

References

[1] Roesch, M. *et al.* (1999) Snort: Lightweight intrusion detection for networks. In *LISA*, **99**: 229–238.

[2] Brustoloni, J., Farnan, N., Villamarín-Salomón, R. and Kyle, D. (2009) Efficient detection of bots in subscribers' computers. In *Communications, 2009. ICC'09. IEEE International Conference on* (IEEE): 1–6.

[3] Dietrich, C.J. and Rossow, C. (2009) Empirical research of ip blacklists. In *ISSE 2008 Securing Electronic Business Processes* (Springer), 163–171.

[4] Choi, H., Lee, H., Lee, H. and Kim, H. (2007) Botnet detection by monitoring group activities in dns traffic. In *Computer and Information Technology, 2007. CIT 2007. 7th IEEE International Conference on* (IEEE): 715–720.

[5] Villamarín-Salomón, R. and Brustoloni, J.C. (2008) Identifying botnets using anomaly detection techniques applied to dns traffic. In *Consumer Communications and Networking Conference, 2008. CCNC 2008. 5th IEEE* (IEEE): 476–481.

[6] Gu, G. (2008) *Correlation-based botnet detection in enterprise networks* (ProQuest).

[7] Olmedilla, D., Rana, O.F., Matthews, B. and Nejdl, W. (2005) Security and trust issues in semantic grids. *Semantic Grid* **5271**.

[8] Yadav, S., Reddy, A.K.K., Reddy, A. and Ranjan, S. (2010) Detecting algorithmically generated malicious domain names. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement* (ACM): 48–61.

[9] McGrath, D.K. and Gupta, M. (2008) Behind phishing: An examination of phisher modi operandi. *LEET* **8**: 1–8.

[10] Bilge, L., Kirda, E., Kruegel, C. and Balduzzi, M. (2011) Exposure: Finding malicious domains using passive dns analysis. In *NDSS*.

[11] Ma, J., Saul, L.K., Savage, S. and Voelker, G.M. (2009) 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* (ACM): 1245–1254.

[12] Blum, A., Wardman, B., Solorio, T. and Warner, G. (2010) Lexical feature based phishing url detection using online learning. In *Proceedings of the 3rd ACM workshop on Artificial intelligence and security* (ACM): 54–60.

[13] Satten, C. (2008) *Lossless Gigabit Remote Packet Capture With Linux*. Tech. rep., University of Washington. URL http://staff.washington.edu/corey/gulp/.

[14] DeependResearch, Trojan nap aka kelihos/hlux, http://www.deependresearch.org/2013/02/trojan-nap-aka-kelihoshlux-feb-2013.html. Visited Feb 2013.

[15] Holz, T., Steiner, M., Dahl, F., Biersack, E. and Freiling, F.C. (2008) Measurements and mitigation of

peer-to-peer-based botnets: A case study on storm worm. *LEET* **8**: 1–9.

[16] Nazario, J. (2009), Twitter-based botnet command channel, http://asert.arbornetworks.com/2009/08/twitter-based-botnet-command-channel/. Visited October 2013.

[17] Contagio, Skynet tor botnet / trojan.tbot samples, http://contagiodump.blogspot.nl/2012/12/dec-2012-skynet-tor-botnet-trojantbot.html. Visited Feb 2014.

[18] Microsoft.com, Worm:win32/morto.a, http://www.microsoft.com/security/portal/threat/encyclopedia/entry.aspx?Name=Worm:Win32/Morto.A. Visited April 2014.

[19] Alexa.com, Alexa, the web information company, http://www.alexa.com/topsites. Visited Mar 2013.

[20] Google.com, Top 1000 sites - doubleclick ad planner, http://www.google.com/adplanner/static/top1000/. Visited Mar 2013.

[21] Jarmoc, J. and Unit, D.S.C.T. (2012) Ssl/tls interception proxies and transitive trust. *Black Hat Europe* .

[22] Nagaraja, S., Houmansadr, A., Piyawongwisal, P., Singh, V., Agarwal, P. and Borisov, N. (2011) Stegobot: a covert social network botnet. In *Information Hiding* (Springer): 299–313.

[23] Gu, G., Perdisci, R., Zhang, J., Lee, W. *et al.* (2008) Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection. In *USENIX Security Symposium*: 139–154.

[24] Rieck, K., Schwenk, G., Limmer, T., Holz, T. and Laskov, P. (2010) Botzilla: detecting the phoning home of malicious software. In *Proceedings of the 2010 ACM Symposium on Applied Computing* (ACM): 1978–1984.

[25] Dietrich, C.J., Rossow, C. and Pohlmann, N. (2013) Cocospot: Clustering and recognizing botnet command and control channels using traffic analysis. *Computer Networks* **57**(2): 475–486.

[26] Giroire, F., Chandrashekar, J., Taft, N., Schooler, E. and Papagiannaki, D. (2009) Exploiting temporal persistence to detect covert botnet channels. In *Recent Advances in Intrusion Detection* (Springer): 326–345.

[27] Strayer, W.T., Lapsely, D., Walsh, R. and Livadas, C. (2008) Botnet detection based on network behavior. In *Botnet Detection* (Springer), 1–24.

[28] Walsh, R., Lapsley, D. and Strayer, W.T. (2009) Effective flow filtering for botnet search space reduction. In *Conference For Homeland Security, 2009. CATCH'09. Cybersecurity Applications & Technology* (IEEE): 141–149.

[29] Zhang, H., Banick, W., Yao, D. and Ramakrishnan, N. (2012) User intention-based traffic dependence analysis for anomaly detection. In *Security and Privacy Workshops (SPW), 2012 IEEE Symposium on* (IEEE): 104–112.

[30] Burghouwt, P., Spruit, M. and Sips, H. (2013) Detection of covert botnet command and control channels by causal analysis of traffic flows. In *Cyberspace Safety and Security* (Springer), 117–131.

[31] Whyte, D., Kranakis, E. and van Oorschot, P.C. (2005) Dns-based detection of scanning worms in an enterprise network. In *NDSS*.