AFRL-RI-RS-TR-2011-098

COUNTERING BOTNETS: ANOMALY-BASED DETECTION, COMPREHENSIVE ANALYSIS, AND EFFICIENT MITIGATION

GEORGIA TECH RESEARCH CORPORATION

MAY 2011

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■**UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

AFRL-RI-RS-TR-2011-098 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/	/s/
KEESOOK HAN	WARREN H. DEBANY JR., Technical Advisor
Work Unit Manager	Information Exploitation and Operations Division
	Information Directorate

REPORT DOCUMENTATION PAGE	*Form Approved* **OMB No. 0704-0188**

1. REPORT DATE *(DD-MM-YYYY)* May 2011	**2. REPORT TYPE** Final Technical Report	**3. DATES COVERED** *(From - To)* March 2008 – October 2010

4. TITLE AND SUBTITLE COUNTERING BOTNETS: ANOMALY-BASED DETECTION, COMPREHENSIVE ANALYSIS, AND EFFICIENT MITIGATION	**5a. CONTRACT NUMBER** N/A
	5b. GRANT NUMBER FA8750-08-2-0141
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Wenke Lee, David Dagon, Jon Giffin, Nick Feamster, Gunter Ollman, Jody Westby, Rick Wesson, Paul Vixie	**5d. PROJECT NUMBER** DHS1
	5e. TASK NUMBER BO
	5f. WORK UNIT NUMBER TN
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Georgia Tech Research Corporation, 505 10th St. NW, Atlanta, GA 30332-0001	**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/Information Directorate, Rome Research Site/RIGD, 525 Brooks Road, Rome NY 13441	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/RI
	11. SPONSORING/MONITORING AGENCY REPORT NUMBER AFRL-RI-RS-TR-2011-098

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
We cover five general areas: (1) botnet detection, (2) botnet analysis, (3) botnet mitigation, (4) add-on tasks to the original contract, including the Conficker Working Group Lessons Learned, Layer-8 Exploration of Botnet Organization, and DREN research, and (5) commercialization in this paper. We have successfully developed new botnet detection and analysis capabilities in this project. These algorithms have been evaluated using real-world data, and have been put into actual, deployed systems. The most significant technical developments include a new dynamic reputation systems for DNS domains, a scalable anomaly detection system for botnet detection in very large network, and a transparent malware analysis system. In addition, on several occasions we have used our botnet data and analysis to help law enforcement agencies arrest botmasters. We also have had great success transitioning technologies to commercial products that are now used by government agencies, ISPs, and major corporations.

15. SUBJECT TERMS
Cyber Security, Cyber Attack, Botnet, Botnet Detection, Botnet Traceback and Attribution, Malware, Malware Analysis, DNS, DNS-Based Monitoring, DNS-Based Redirection, BGP, BGP Route Injection.

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
					KEESOOK HAN
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	UU	58	**19b. TELEPHONE NUMBER** (*Include area code*) N/A

TABLE OF CONTENTS

SECTION **PAGE**

LIST OF FIGURES

LIST OF TABLES

1.0 SUMMARY

1.1 Overview

We cover five general areas: (1) botnet detection, (2) botnet analysis, (3) botnet mitigation, (4) add-on tasks to the original contract, including the Conficker Working Group Lessons Learned, Layer-8 Exploration of Botnet Organization, and DREN research, and (5) commercialization in this final technical paper.
The introduction to the project and an overview of each area is given in Section 2. In Section 3, we present the procedures and results for each area. We evaluate a DNS reputation system and discuss the most significant results in Section 4. Section 5 is the conclusion.

2.0 INTRODUCTION

2.1 Botnet Detection

Our goal was to develop botnet detection technology that is robust against new and evolving forms of botnets and evasion techniques. Toward this goal, we have researched and developed complementary approaches for DNS-based monitoring and network-based anomaly detection.

In DNS monitoring, our approach is to analyze DNS query traffic to identify domain names that are used for botnet command and control (C&C). We have systematically analyzed the DNS protocol behaviors at each layer of the DNS hierarchy, extracted statistical and temporal features, and applied machine learning algorithms to learn models to distinguish the look-up patterns of a botnet C&C domain from those of a legitimate DNS domain.

In network-based anomaly detection, our approach is to analyze in-bound and out-bound traffic of an enterprise network, and apply clustering analysis to identify the group(s) of hosts that have similar C&C like traffic (e.g., communicate to an outside server with similar network flows) and similar malicious activity traffic (e.g., spam). According to our definition, these group(s) of hosts are part(s) of a botnet(s).

2.2 Botnet Analysis

We have developed new malware analysis technologies with the focus of defeating obfuscation techniques employed by malware. Our technologies include an automated unpacker, and a transparent analysis environment where malware cannot know it is being analyzed.

We have also explored techniques and procedures to investigate evidence of at network service providers, communications in on-line forums, etc. that can suggest the whereabouts of botmasters. Our work has yielded promising results and suggested research directions for effective botnet traceback and attribution techniques.

2.3 Botnet Mitigation

We have developed a host-based forensic analysis and recovery tool. It uses virtual machine technologies to build a clean execution context on an infected system and automatically analyzes and recovers from infections. The analysis includes both an off-line forensic analysis of the affected disk as well as host-level execution of behavioral profiles generated by executing the infected system in a protected emulation environment.

We have developed several network-based response techniques and procedures, including DNSbased redirection, and blackhole route injection.

2.4 Add-on Tasks

We organized and led an effort to document the lessons learned in the Conficker botnets. The participants included academic researchers, network providers, and analysts from security companies. A report has been published.

Together with the Internet Law Group, we also explored the regulatory frameworks that permit civil investigation of the financial structure, organization and management of botnets. The goal is to develop techniques that can be used for de-monetizing botnets through the eventual disruption of payment processing in criminal networks.

With the help of DREN, we developed spam and botnet data analysis systems for identifying compromised government assets, clustering analysis of DNS reputation, and DNS Weather Map data.

2.5 Commercialization

Support Intelligence has developed and commercialized several tools and data feed, including DNSBL data analysis tools.

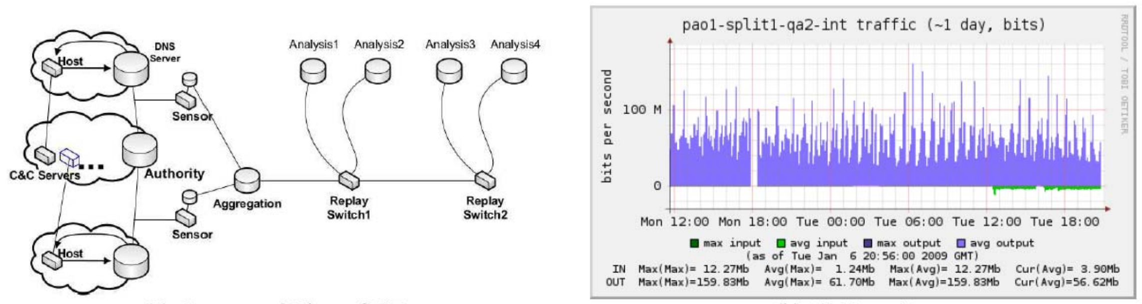
Damballa has also incorporated several algorithms developed in this project into their products and services, including dynamic and passive DNS monitoring, and DNS redirection. Damballa’s growing list of customers including major ISPs and Fortune 500 companies, banks, and universities.

3.0 METHODS, ASSUMPTIONS AND PROCEDURES

3.1 Botnet Detection

3.1.1 DNS-Based Monitoring

Passive DNS Data For part of this project, Georgia Tech and ISC were required to create a passive-DNS data collection system. This has resulted in the creation of the Security Information Exchange, SIE. Details about SIE are found at https://sie.isc.org. Figure 1 (a) shows a conceptual view of the SIE data collection system.



Host
DNS Server
Sensor
C&C Servers
Authority
Aggregation
Sensor
Host
Analysis1
Analysis2
Analysis3
Analysis4
Replay Switch1
Replay Switch2
pao1-split1-qa2-int traffic (~1 day, bits)
bits per second
100 M
0
Mon 12:00 Mon 18:00 Tue 00:00 Tue 06:00 Tue 12:00 Tue 18:00
max input
avg input
max output
avg output
(as of Tue Jan 6 20:56:00 2009 GMT)
IN Max(Max)= 12.27Mb Avg(Max)= 1.24Mb Max(Avg)= 12.27Mb Cur(Avg)= 3.90Mb
OUT Max(Max)=159.83Mb Avg(Max)= 61.70Mb Max(Avg)=159.83Mb Cur(Avg)=56.62Mb
RRDTOOL / TOBI OETIKER

(a) Conceptual View of SIE (b) SIE Data Rates

Figure 1: (a) SIE's architecture: sensors contribute data to an aggregation node, which rebroadcasts DNS data for analysis nodes (b) SIE data averages 45Mb/s, with bursts up to 100Mb/s.

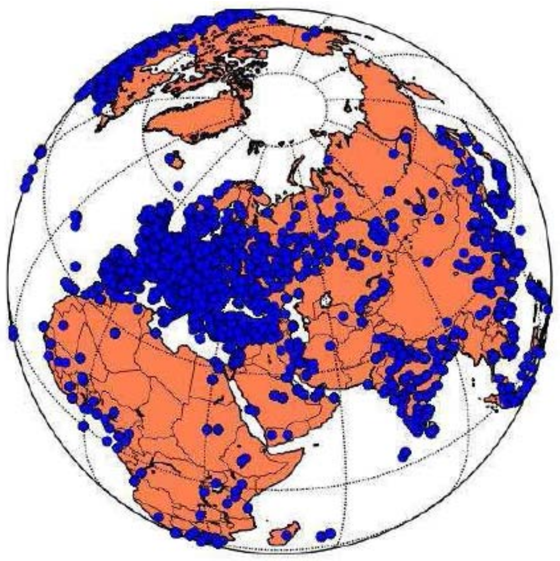


Figure 2: GeoIP Plotting of Bot Victims.

GeoIP Information Georgia Tech has produced a IPv4-to-jurisdiction database. The database allows one to map a given IP (if located in the US) to a US District Court jurisdiction. These jurisdictions in turn correspond to FBI districts. This mapping permits researchers to identify areas where victims are located, and which FBI office may have jurisdiction over any resulting investigation.

As a result of this exercise, we have also created a variety of tools that plot geo-locations of botnets. These tools can create animations or static images, as shown in Figure 2. This sort of visualization assisted in the development and debugging of the GeoIP deliverable, and may prove useful to other researchers.

dnscap We have developed dnscap, a dns capture and analysis tool. A copy of the tool is available at https://www.dns-oarc.net/tools/dnscap. The tool is similar to various PCAP handling tools, except that it understands the DNS protocol, and exposes filtering options to command line users. Thus, one can use dnscap to filter pcap data based on the content of RRsets, and not merely based on the IP and UDP header information.

Detection of Fast Flux Networks We developed a passive approach for detecting and tracking malicious flux service networks. Our detection system is based on passive analysis of recursive DNS (RDNS) traffic traces collected from multiple large networks. Contrary to previous work, our approach is not limited to the analysis of suspicious domain names extracted from spam emails or precompiled domain blacklists. Instead, our approach is able to detect malicious flux service networks in-the-wild, i.e., as they are accessed by users who fall victims of malicious content advertised through blog spam, instant messaging spam, social website spam, etc., beside email spam. We experimented with the RDNS traffic passively collected at two large ISP networks. Overall, our sensors monitored more than 2.5 billion DNS queries per day from millions of distinct source IPs for a period of 45 days. Our experimental results show that the proposed approach is able to accurately detect malicious flux service networks. Furthermore, the results show that our passive detection and tracking of malicious flux service networks can facilitate spam filtering applications.

We describe our approach in more detail below.

Our detection system is based on passive analysis of recursive DNS traces collected from multiple large networks. In practice, we deploy a sensor in front of the recursive DNS (RDNS) server of different networks, passively monitor the DNS queries and responses from the users to the RDNS, and selectively store information about potential fast-flux domains into a central DNS data collector. Since the amount of RDNS traffic in large networks is often overwhelming, we devised a number of prefiltering rules that aim at identifying DNS queries to potential fast-flux domain names, while discarding the remaining requests to legitimate domain names. Our prefiltering stage is very conservative, nevertheless, it is able to reduce the volume of the monitored DNS traffic to a tractable amount without discarding information about domain names actually related to malicious flux services. Once information about potential malicious flux domains has been collected for a certain epoch E (e.g., one day), we perform a more fine-grain analysis. First, we apply a clustering process to the domain names collected during E, and we group together domain names that are related to each other. For example we group together domain names that point to the same Internet service, are related to the same content distribution network (CDN), or are part of the same malicious flux network. Once the monitored domain names have been grouped, we classify these clusters of domains and the related monitored resolved IP addresses as either being part of a malicious flux service network or not. This is in contrast with most previous works, in which single domain names are considered independently from each other, and classified as either fast-flux or non-fast-flux. Our detection approach has a fundamental advantage, compared to previous work. Passively monitoring live users' DNS traffic offers a new vantage point, and allows us to capture queries to flux domain names that are advertised through a variety of means, including for example blog spam, social websites spam, search engine spam, and instant messaging spam, beside email spam and precompiled domain blacklists. Furthermore, differently from the active probing approach used in previous work, we passively monitor live users traffic without interacting ourselves with the flux networks. Active probing of fast-flux domain names may be detected by the attacker, who often controls the authoritative name servers responsible for responding to DNS queries about her fast-flux domain names. If the attacker detects that an active probing system is trying to track her malicious flux service network, she may stop responding to queries coming from the probing system to prevent unveiling further information. On the other hand, our detection system is able to detect flux services in a stealthy way. This work is to be published in the 2009 Annual Computer Security Applications Conference [20].

Passive DNS Monitoring for Dynamic Reputation of Domains The Domain Name System (DNS) is an essential protocol used by not only legitimate Internet applications but also cyber attacks. For example, botnets are known to rely on DNS to support agile command and control infrastructures. An effective way to disrupt these attacks is to put the domains known to be involved in malicious activities on a block- list (or, blacklist) or a filtering rule in a firewall or network intrusion detection system. To evade such security countermeasures, attackers have recently developed techniques to make very agile use of DNS, e.g., by using new domains daily, to render static blacklisting or firewalling ineffective. We developed a dynamic reputation system for DNS called Notos (see Figure 3). The premise of this system is that malicious, agile use of DNS has unique characteristics and can be distinguished from legitimate, professionally-provisioned DNS services. Notos uses passive DNS query data and analyzes the network and zone features of domains. It builds models of known legitimate domains and malicious domains, and uses these models to compute a reputation score for a new domain indicative of whether the domain is likely to be for malicious or legitimate uses. We have evaluated Notos in a large ISP's networks with DNS traffic from 1.4 million users. Our results show that Notos can identify malicious domains with very high accuracy (true positive rate) (96.8%) and very low false positive rate (0.38%), and can identify these domains weeks or even months before they appear in public blacklists. This work was published in the 19th USENIX Security Symposium, August 2010[7].

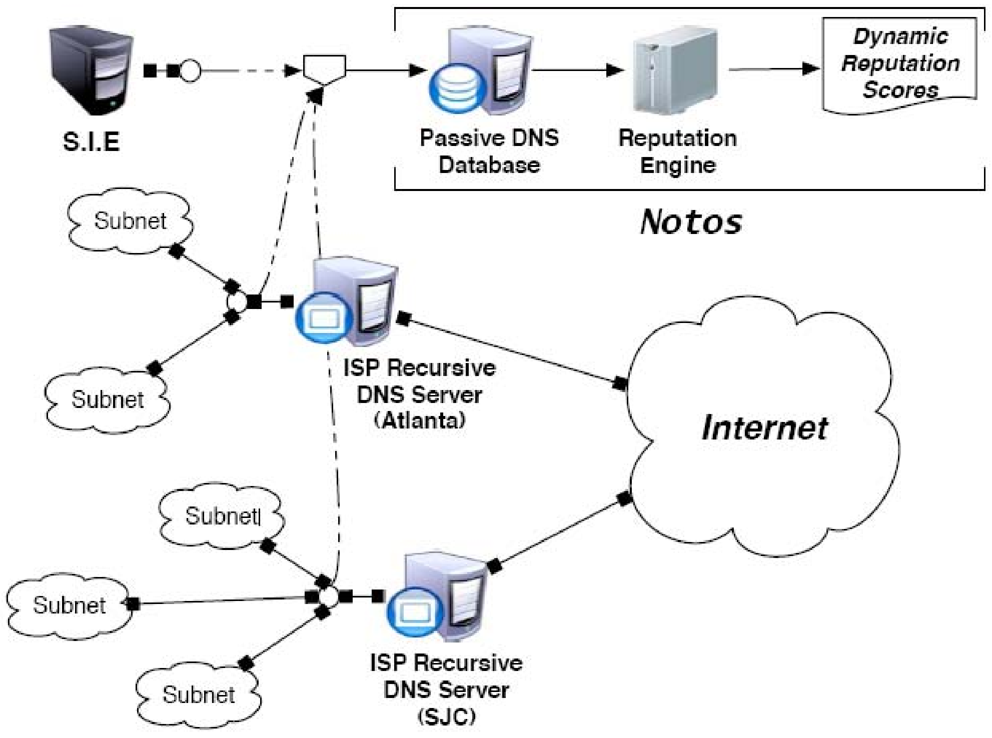

S.I.E
Passive DNS Database
Reputation Engine
Dynamic Reputation Scores
Notos
Subnet
Subnet
ISP Recursive DNS Server (Atlanta)
Internet
Subnet
Subnet
Subnet
ISP Recursive DNS Server (SJC)

Figure 3: System Overview

Below we provide some details of the Notos system.

The goal of the Notos reputation system is to dynamically assign reputation scores to domain names. Given a domain name d, we want to assign a low reputation score if d is involved in malicious activities (e.g., if it has been involved with botnet C&C servers, spam campaigns, malware propagation, etc.). On the other hand, we want to assign a high reputation score if d is associated with legitimate Internet services.

Notos' main source of information is a passive DNS (pDNS) database, which contains historical information about domain names and their resolved IPs. Our pDNS database is constantly updated using real-world DNS traffic from multiple geographically diverse locations as shown in Figure 3. We collectDNS traffic from two ISP recursive DNS servers (RDNS) located in Atlanta and San Jose. The ISP nodes witness 30,000 DNS queries/second during peak hours. We also collect DNS traffic through the Security Information Exchange (SIE) [10], which aggregates DNS traffic received by a large number of RDNS servers from authoritative name servers across North America and Europe. In total, the SIE project processes approximately 200 Mbit/s of DNS messages, several times the total volume of DNS traffic in a single US ISP.

Another source of information we use is a list of known malicious domains. For example, we run known malware samples in a controlled environment and we classify as suspicious all the domains contacted by malware samples that do not match a pre-compiled white list. In addition, we extract suspicious domain names from spam emails collected using a large spam-trap. Again, we discard the domains that match our whitelist and consider the rest as potentially malicious. Furthermore, we collect a large list of popular, legitimate domains from `alexa.com`. The set of known malicious and legitimate domains represents our *knowledge base*, and is used to train our reputation engine.

Intuitively, a domain name d can be considered suspicious when there is evidence that d or its IP addresses are (or were in previous months) associated with known malicious activities. The more evidence of "bad associations" we can find about d, the lower the reputation score we will assign to it. On the other hand, if there is evidence that d is (or was in the past) associated with legitimate, professionally run Internet services, we will assign it a higher reputation score.

Before describing the internals of our reputation system, we introduce some basic terminology. A domain name d consists of a set of substrings or labels separated by a period; the rightmost label is called the *top-level* domain, or TLD. The *second-level* domain (2LD) represents the two rightmost labels separated by a period; the *third-level* domain (3LD) analogously contains the three rightmost labels, and so on. As an example, given the domain name d="a.b.example.com", $TLD(d)$="com", $2LD(d)$="example.com", and $3LD(d)$="b.example.com".

Let s be a domain name (e.g., s="example.com"). We define $Zone(s)$ as the set of domains that include s and all domain names that end with a period followed by s (e.g., domains ending in ".example.com").

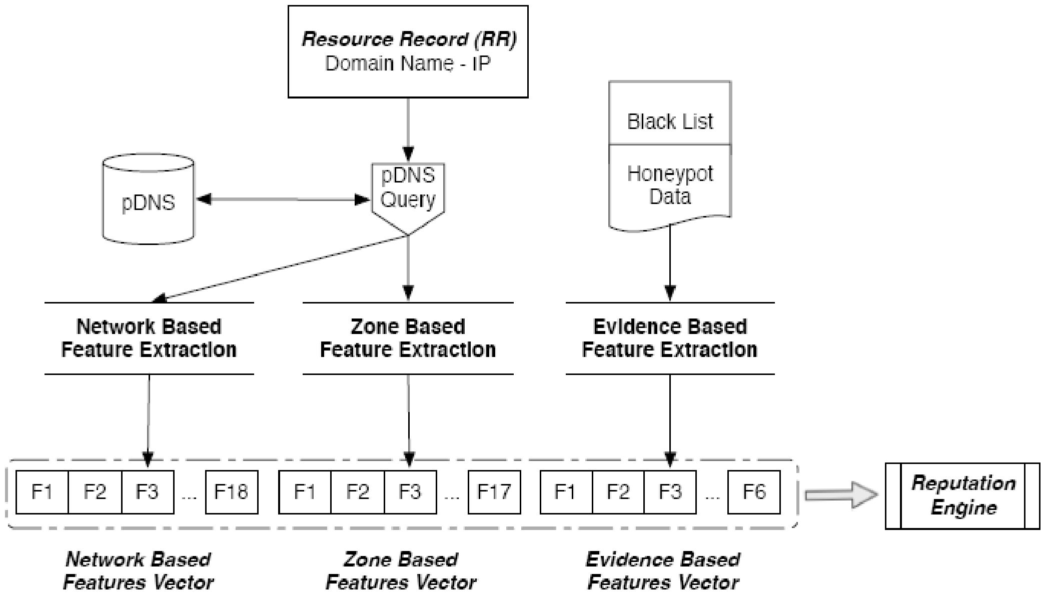
Let $D = \{d_1, d_2, ..., d_m\}$ be a set of domain names. We call $A(D)$ the set of IP addresses ever pointed to by any domain name $d \in D$.

Given an IP address a, we define $BGP(a)$ to be the set of all IPs within the BGP prefix of a, and $AS(a)$ as the set of IPs located in the autonomous system in which a resides. In addition, we can extend these functions to take as input a set of IPs: given IP set $A = a_1, a_2, ..., a_N$, $BGP(A) = S_{k=1..N} BGP(a_k)$; $AS(a)$ is similarly extended.

To assign a reputation score to a domain name d we proceed as follows. First, we consider the most current set $A_c(d) = \{a_i\}_{i=1..m}$ of IP addresses to which d points. Then, we query our pDNS database to retrieve the following information:

- ***Related Historic IPs (RHIPs),*** which consist of the union of $A(d)$, $A(Zone(3LD(d)))$, and $A(Zone(2LD(d)))$. In order to simplify the notation we will refer to $A(Zone(3LD(d)))$ and $A(Zone(2LD(d)))$ as $A_{3LD}(d)$ and $A_{2LD}(d)$, respectively.
- ***Related Historic Domains (RHDNs)***, which comprise the entire set of domain names that ever resolved to an IP address $a \in AS(A(d))$. In other words, RHDNs contain all the domains d_i for which $A(d_i) \cap AS(A(d)) 6= \emptyset$.

After extracting the above information from our pDNS database, we measure a number of statistical features. Specifically, for each domain d we extract three groups of features, as shown in Figure 4:



Resource Record (RR)
Domain Name - IP
Black List
Honeypot Data
pDNS
pDNS Query
Network Based Feature Extraction
Zone Based Feature Extraction
Evidence Based Feature Extraction
F1 F2 F3 ... F18
F1 F2 F3 ... F17
F1 F2 F3 ... F6
Reputation Engine
Network Based Features Vector
Zone Based Features Vector
Evidence Based Features Vector

Figure 4: Computing network/zone/evidence based features.

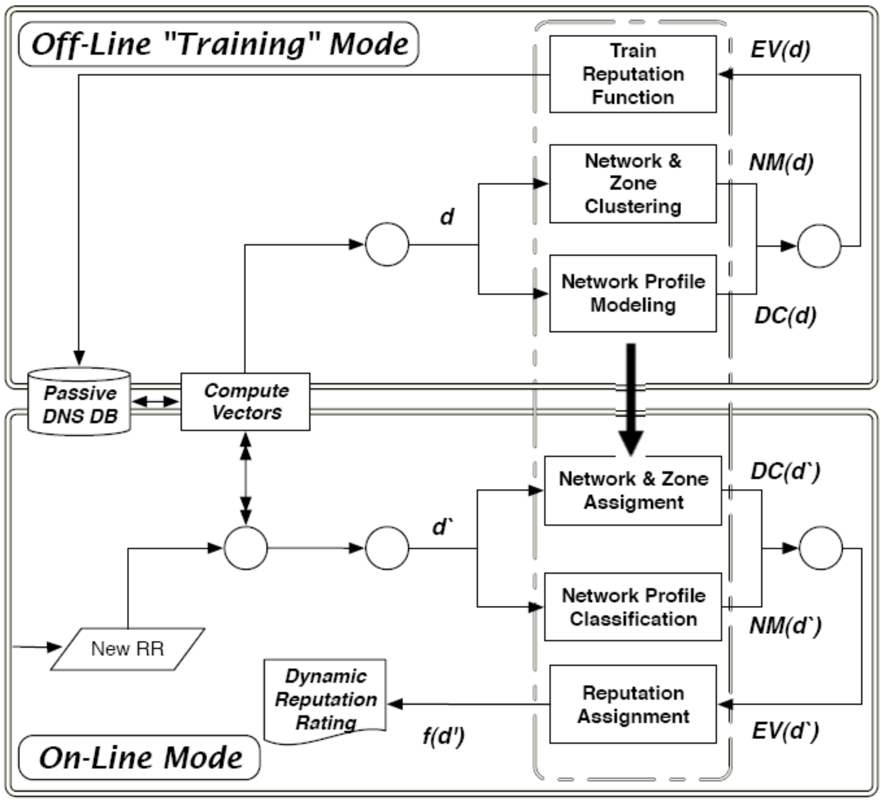
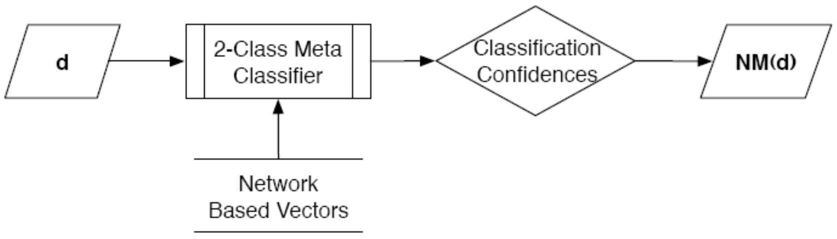
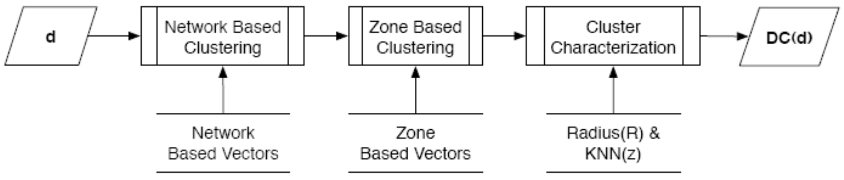


Figure 5: Off-line and on-line modes in Notos.



(a)



(b)

Figure 6: (a) Network profile modeling in Notos. (b) Network and zone based clustering in Notos.

- **Network-based features:** The first group of statistical features is extracted from the set of RHIPs. We measure quantities such as the total number of IPs historically associated with d, the diversity of their geographical location, the number of distinct autonomous systems (ASs) in which they reside, etc.
- **Zone-based features:** The second group of features we extract are those from the RHDNs set. We measure the average length of domain names in RHDNs, the number of distinct TLDs, the occurrence frequency of different characters, etc.
- **Evidence-based features:** The last set of features includes the measurement of quantities such as the number of distinct malware samples that contacted the domain d, the number of malware samples that connected to any of the IPs pointed by d, etc.

Once extracted, these statistical features are fed to the reputation engine. Notos' reputation engine operates in two modes: an off-line "training" mode and an on-line "classification" mode. During the off-line mode, Notos *trains* the reputation engine using the information gathered in our *knowledge base*, namely the set of known malicious and legitimate domain names and their related IP addresses. Afterwards, during the on-line mode, for each new domain d, Notos queries the trained reputation engine to compute a reputation score for d (see Figure 5). We now explain the details about the statistical features we measure, and how the reputation engine uses them during the off-line and on-line modes to compute a domain names' reputation score.

We now describe key statistical features and the intuition behind their selection.

Given a domain d we extract a number of statistical features from the set RHIPs of d. Our network-based features describe how the operators who *own* d and the IPs that domain d points to, allocate their network resources. Internet miscreants often abuse DNS to operate their malicious networks with a high level of *agility*. Namely, the domain names and IPs that are used for malicious purposes are often short-lived and are characterized by a high *churn* rate. This agility avoids simple blacklisting or removals by law enforcement. In order to measure the level of agility of a domain name d, we extract eighteen statistical features that describe d's *network profile*. Our network features fall into the following three groups:

- ***BGP features***: This subset consists of a total of nine features. We measure the number of distinct BGP prefixes related to BGP(A(d)), the number of countries in which these BGP prefixes reside, and the number of organizations that *own* these BGP prefixes; the number of distinct IP addresses in the sets A_{3LD}(d) and A_{2LD}(d); the number of distinct BGP prefixes related to BGP(A_{3LD}(d)) and BGP(A_{2LD}(d)), and the number of countries in which these two sets of prefixes reside.
- ***AS features***: This subset consists of three features, namely the number of distinct autonomous systems related to AS(A(d)), AS(A_{3LD}(d)), and AS(A_{2LD}(d)).
- ***Registration features***: This subset consists of six features. We measure the number of distinct registrars associated with the IPs in the A(d) set; the diversity in the registration dates related to the IPs in A(d); the number of distinct registrars associated with the IPs in the A_{3LD}(d) and A_{2LD}(d) sets; and the diversity in the registration dates for the IPs in A_{3LD}(d) and A_{2LD}(d).

While most legitimate, professionally run Internet services have a very stable network *profile*, which is reflected into low values of the network features described above, the profiles of malicious networks (e.g., fast-flux networks) usually change relatively frequently, thus causing their network features to be assigned higher values. We expect a domain name d from a legitimate zone to exhibit a small values in its AS features, mainly because the IPs in the RHIPs should belong to the same organization or a small number of different organizations. On the other hand, if a domain name d participates in malicious activities (i.e., botnet activities, flux networks), then it could reside in a large number of different networks. The list of IPs in the RHIPs that correspond to the malicious domain name will produce AS features with higher values. In the same sense, we measure that homogeneity of the registration information for benign domains. Legitimate domains are typically linked to address space owned by organizations that acquire and announce network blocks in some order. This means that the registration-feature values for a legitimate domain name d that owned by the same organizations will produce a list of IPs in the RHIPs that will have small registration feature values. If this set of IPs exhibits high registration feature values, it means that they very likely reside in different registrars and were registered on different dates. Such registration-feature properties are typically linked with fraudulent domains.

The network-based features measure a number of characteristics of IP addresses historically related to a given domain name d. On the other hand, the zone-based features measure the characteristics of domain names historically associated with d. The intuition behind the zonebased features is that while legitimate Internet services may be associated with many different domain names, these domain names usually have strong similarities.

For example, `google.com`, `googlesyndication.com`, `googlewave.com`, etc., are all related to Internet services provided by Google, and contain the string “google” in their name. On the other hand, malicious domain names related to the same spam campaign, for example, often look randomly generated and share few common characteristics. Therefore, our zone-based features aim to measure the level of *diversity* across the domain names in the RHDNs set. Given a domain name d, we extract seventeen statistical features that describe the properties of the set RHDNs of domain names related to d. We divide these seventeen features into two groups:

- ***String features*****:** This group consists of twelve features. We measure the number of distinct domain names in RHDNs, and the average and standard deviation of their length; the mean, median, and standard deviation of the occurrence frequency of each single character in the domain name strings in RHDNs; the mean, median and standard deviation of the distribution of 2-grams (i.e., pairs of characters); the mean, median and standard deviation of the distribution of 3-grams.
- ***TLD features*****:** This group consists of five features. For each domain d_i in the RHDNs set, we extract its top-level domain $TLD(d_i)$ and we count the number of distinct TLD strings that we obtain; we measure the ratio between the number of domains d_i whose $TLD(d_i)$=“.com” and the total number of TLD different from “.com”; also, we measure the mean, median, and standard deviation of the occurrence frequency of the TLD strings.

It is worth noting that whenever we measure the mean, median and standard deviation of a certain property, we do so in order to summarize the shape of its distribution. For example, by measuring the mean, median, and standard deviation of the occurrence frequency of each character in a set of domain name strings, we summarize how the distribution of the character frequency looks like.

We use the evidence-based features to determine to what extent a given domain d is associated with other known malicious domain names or IP addresses. As mentioned above, Notos collects a *knowledge base* of known suspicious, malicious, and legitimate domain names and IPs from public sources. For example, we collect malware-related domain names by executing large numbers of malware samples in a controlled environment. Also, we check IP addresses against a number of public IP blacklists. Given a domain name d, we measure six statistical features using the information in the knowledge base. We divide these features into two groups:

- ***Honeypot features***: We measure three features, namely the number of distinct malware samples that, when executed, try to contact d or any IP address in $A(d)$; the number of malware samples that contact any IP address in $BGP(A(d))$; and the number of samples that contact any IP address in $AS(A(d))$.
- ***Blacklist features***: We measure three features, namely the number of IP addresses in $A(d)$ that are listed in public IP blacklists; the number of IPs in $BGP(A(d))$ that

are listed in IP blacklists; and the number of IPs in AS(A(d)) that are listed in IP blacklists.

Notos uses the blacklist features from the evidence vector so it can identify the re-use of known malicious network resources like IPs, BGP prefixes or even ASs. Domain names are significantly cheaper than IPv4 addresses; so malicious users tend to reuse address space with new domain names. We should note that the evidence-based features represent only part of the information we used to compute the reputation scores. The fact that a domain name was queried by malware does not automatically mean that the domain will receive a low reputation score.

Notos' reputation engine is responsible for deciding whether a domain name d has characteristics that are similar to either legitimate or malicious domain names. In order to achieve this goal, we first need to *train* the engine to recognize whether d belongs (or is "close") to a known *class of domains*. This training can be repeated periodically, in an off-line fashion, using historical information collected in Notos' *knowledge base*. Once the engine has been trained, it can be used in on-line mode to assign a reputation score to each new domain name d.

We first explain how the reputation engine is trained, and then we explain how a trained engine is used to assign reputation scores. During off-line training (Figure 5), the reputation engine builds three different modules.

We briefly introduce each module and then elaborate on the details:

- ***Network Profiles Model*****:** A model of how well known networks behave. For example, we model the network characteristics of popular content delivery networks (e.g., Akamai, AmazonCloudFront), and large *popular* websites (e.g., google.com, yahoo.com). During the on-line mode, we compare each new domain name d to these models of well-known network profiles, and use this information to compute the final reputation score, as explained below.
- ***Domain Name Clusters*****:** We group domain names into clusters sharing similar characteristics. We create these clusters of domains to identify groups of domains that contain mostly malicious domains, and groups that contain mostly legitimate domains. In the on-line mode, given a new domain d, if d (more precisely, d's projection into a statistical feature space) falls within (or close to) a cluster of domains containing mostly malicious domains, for example, this gives us a hint that d should be assigned a low reputation score.
- ***Reputation Function*****:** For each domain name d_i, $i = 1..n$, in Notos' knowledge base, we *test* it against the trained network profiles model and domain name clusters. Let $NM(d_i)$ and $DC(d_i)$ be the output of the Network Profiles (NP) module and the Domain Clusters (DC) module, respectively. The reputation function takes in input $NM(d_i)$, $DC(d_i)$, and information about whether d_i and its resolved IPs $A(d_i)$ are known to be legitimate, suspicious, or malicious (i.e., if they appeared in a domain name or IP blacklist), and builds a model that can assign a reputation score between zero and one to d. A reputation score close to

zero signifies that d is a malicious domain name while a score close to one signifies that d is benign.

We now describe each module in detail.

During the off-line training mode, the reputation engine builds a model of well-known network behaviors. An overview of the network profile modeling module can be seen in Figure 6. In practice we select five sets of domain names that share similar characteristics, and *learn* their network profiles. For example, we identify a set of domain names related to very popular websites (e.g., google.com, yahoo.com, amazon.com) and for each of the related domain names we extract their network features. We then use the extracted feature vectors to train a statistical classifier that will be able to recognize whether a new domain name d has network characteristics similar to the popular websites we modeled.

In our current implementation of Notos we model the following classes of domain names:

- **Popular Domains**. This class consists of a large set of domain names under the following DNS zones: google.com, yahoo.com, amazon.com, ebay.com, msn.com, live.com, myspace. com, and facebook.com.
- **Common Domains**. This class of domains includes domain names under the top one hundred zones, according to `alexa.com`. We exclude from this group all the domain names already included in the *Popular Domains* class (which we model separately).
- **Akamai Domains**. Akamai is a large content delivery network (CDN), and the domain names related to this CDN have very peculiar network characteristics. To model the network profile of Akamai's domain names, we collect a set of domains under the following zones: akafms.net, akamai.net, akamaiedge.net, akamai.com, akadns.net, and akamai.com.
- **CDN Domains**. In this class we include domain names related to CDNs other than Akamai. For example, we collect domain names under the following zones: panthercdn.com, llnwd.net, cloudfront.net, nyud.net, nyucd.net and redcondor.net. We chose not to aggregate these CDN domains and Akamai's domains in one class, since we observed that Akamai's domains have a very unique network profile. Therefore, learning two separate models for the classes of *Akamai Domains* and *CDN Domains* allows use to achieve better classification accuracy during the on-line mode, compared to learning only one model for both classes.
- **Dynamic DNS Domains**. This class includes a large set of domain names registered under two of the largest dynamic DNS providers, namely No-IP (`no-ip.com`) and DynDNS (`dyndns.com`). For each class of domains, we train a statistical classifier to distinguish between one of the classes and all the others. Therefore, we train five different classifiers. For example, we train a classifier that can distinguish between the class of *Popular Domains* and all other classes of domains. That is, given a new domain name d, this classifier is able to recognize whether d's network profile looks like the profile of a well-known popular

domain or not. Following the same logic we, can recognize network profiles for the other classes of domains. In this phase, the reputation engine takes the domain names collected in our pDNS database during a *training period*, and builds clusters of domains that share similar network and zone based features. The overview of this module can be seen in Figure 6(b). We perform clustering in two steps. In the first step we only use the network-based features to create coarse-grained clusters. Then, in the second step, we split each coarse-grained cluster into finer clusters using only the zone-based features, as shown in Figure 7.

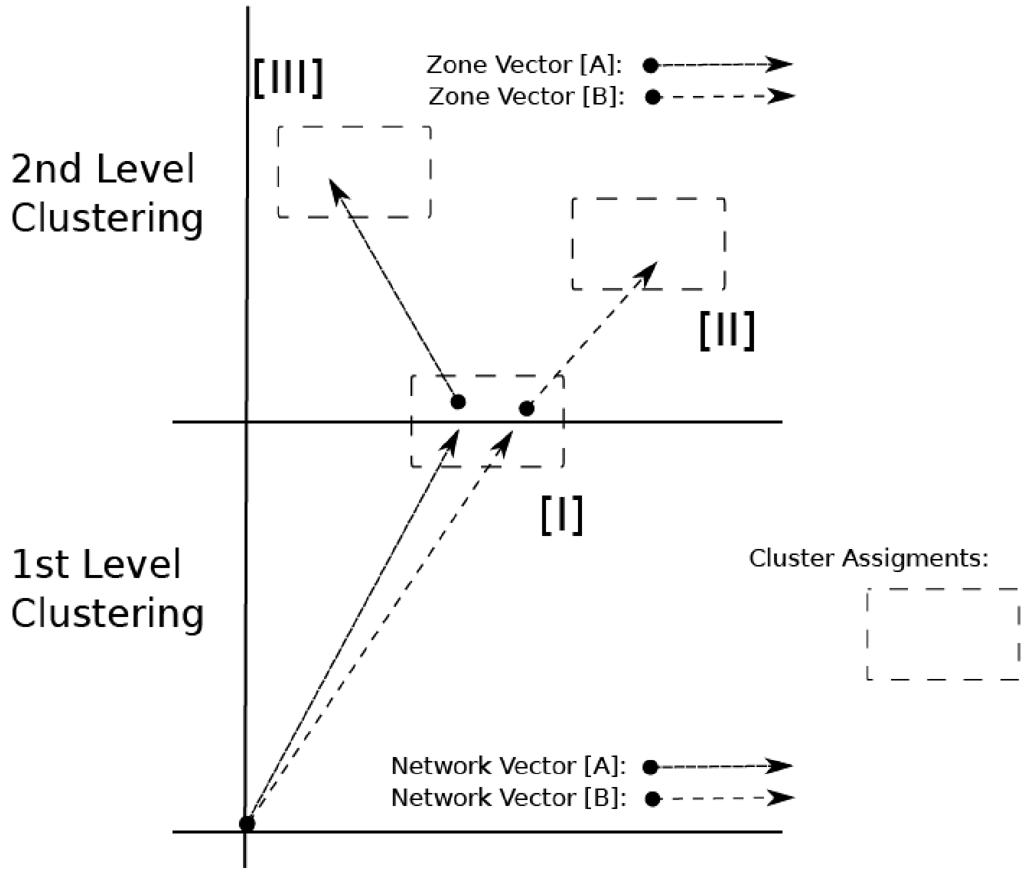


Figure 7: Network & zone based clustering process in Notos, in the case of a Akamai [A] and a malicious [B] domain name.

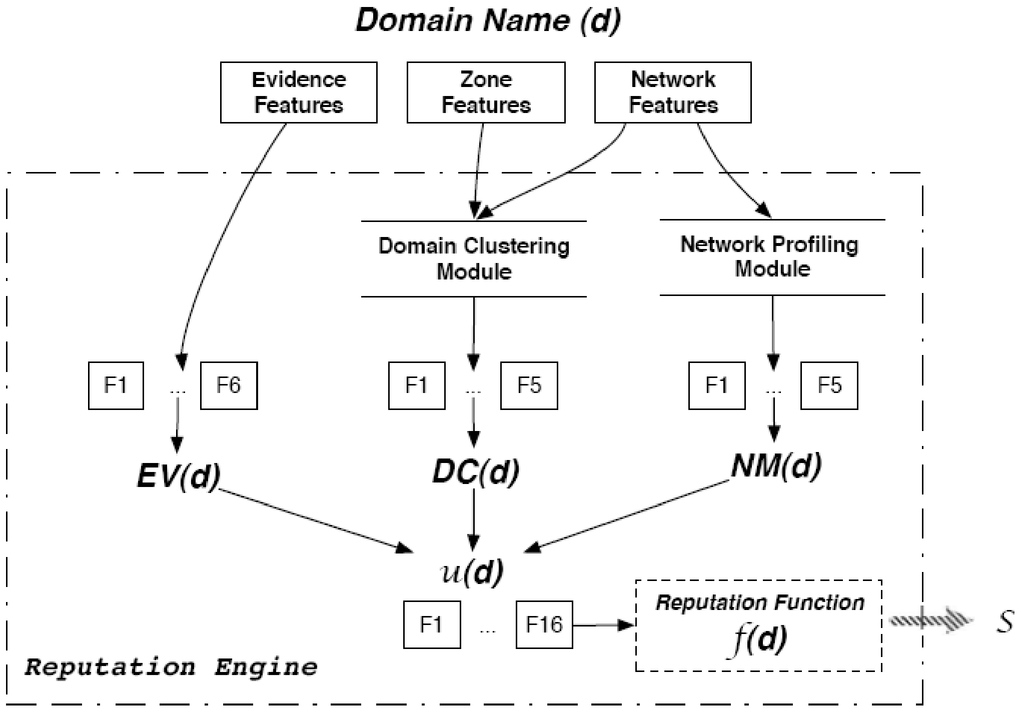


Figure 8: The output from the network profiling module, the domain clustering module and the evidence vector will assist the reputation function to assign the reputation score to the domain d.

Network-based Clustering The objective of network-based clustering is to group domains that share similar levels of *agility*. This creates separate clusters of domains with “stable” network characteristics and “non-stable” networks (like CDNs and malicious flux networks).

Zone-based Clustering After clustering the domain names according to their network-based features, we further split the network-based clusters of domain names into finer groups. In this step, we group domain names that are in the same network-based cluster and also share similar zone-based features. To better understand how the zone-based clustering works, consider the following examples of zone-based clusters:

Cluster 1:

..., 72.247.176.81 e55.g.akamaiedge.net, 72.247.176.94 e68.g.akamaiedge.net, 72.247.176.146 e120.g.akamaiedge.net, 72.247.176.65 e39.na.akamaiedge.net, 72.247.176.242 e216.g.akamaiedge.net, 72.247.176.33 e7.g.akamaiedge.net, 72.247.176.156 e130.g.akamaiedge.net, 72.247.176.208 e182.g.akamaiedge.net, 72.247.176.198 e172.g.akamaiedge.net, 72.247.176.217 e191.g.akamaiedge.net, 72.247.176.200 e174.g.akamaiedge.net, 72.247.176.99 e73.g.akamaiedge.net, 72.247.176.103 e77.g.akamaiedge.net, 72.247.176.59 e33.c.akamaiedge.net, 72.247.176.68 e42.gb.akamaiedge.net, 72.247.176.237 e211.g.akamaiedge.net, 72.247.176.71 e45.g.akamaiedge.net, 72.247.176.239 e213.na.akamaiedge.net, 72.247.176.120 e94.g.akamaiedge.net, ...

Cluster 2:

..., **90.156.145.198 spzr.in,** 90.156.145.198 vwui.in, 90.156.145.198 x9e.ru, 90.156.145.50 v2802.vps.masterhost.ru, 90.156.145.167 www.inshaker.ru, 90.156.145.198 x7l.ru, 90.156.145.198 c3q.at, 90.156.145.198 ltkq.in, 90.156.145.198 x7d.ru, 90.156.145.198 zdlz.in, 90.156.145.159 www.designcollector.ru, 90.156.145.198 x7o.ru, 90.156.145.198 q5c.ru, 90.156.145.159 designtwitters.com, 90.156.145.198 u5d.ru, 90.156.145.198 x9d.ru, 90.156.145.198 xb8.ru, 90.156.145.198 xg8.ru, 90.156.145.198 x8m.ru, 90.156.145.198 shopfilmworld.cn, 90.156.145.198 bigappletopworld.cn, 90.156.145.198 uppd.in, ...

Each element of the cluster is a *domain name - IP address* pair. These two groups of domains belonged to the same network cluster, but were separated into two different clusters by the zonebased clustering phase. *Cluster 1* contains domain names belonging to Akamai's CDN, while the domains in *Cluster 2* are all related to malicious websites that distribute malicious software. The two clusters of domains share similar network characteristics, but have significantly different zonebased features. For example, consider domain names d_1="e55.g.akamaiedge.net" from the first cluster, and d_2="spzr.in" from the second cluster. The reason why d_1 and d_2 were clustered in the same network-based cluster is because the set of RHIPs for d_1 and d_2 have similar characteristics. In particular, the *network agility* properties of d_2 make it look like if it was part of a large CDN. However, when we consider the set of RHDNs for d_1 and d_2, we can notice that the zone-based features of d_1 are much more "stable" than the zone-based features of d_2. In other words, while the RHDNs of d_1 share strong domain name similarities (e.g., they all share the substring "akamai") and have low variance of the *string features*, the strong *zone agility* properties of d_2 affect the zone-based features measured on d_2's RHDNs and make d_2 look very different from d_1.

One of the main advantages of Notos is the reliable assignment of low reputation scores to domain names participating in "agile" malicious campaigns. Less agile malicious campaigns, e.g., Fake AVs campaigns may use domain names structured to resemble CDN related domains. Such strategies would not be beneficial for the FakeAV campaign, since domains like `virus-scan1.com`, `virus-scan2.com`, etc., can be trivially blocked by using simple regular expressions [21]. In other words, the attackers need to introduce more "agility" at both the network and domain name level in order to avoid simple domain name blacklisting. Notos would only require a few labeled domain names belonging to the malicious campaign for training purposes, and the reputation engine would then generalize to assign a low reputation score to the remaining (previously unknown) domain names that belong to the same malicious campaign.

Once we build a model of well-known network profiles and the domain clusters, we can build the reputation function. The reputation function will assign a reputation score in the interval $[0, 1]$ to domain names, with 0 meaning low reputation (i.e., likely malicious) and 1 meaning high reputation (i.e., likely legitimate). We implement our reputation function as a statistical classifier. In order to train the reputation function, we consider all the domain names d_i, $i = 1, .., n$ in Notos' *knowledge base*, and we feed each domain d_i to the *network profiles* module and to the *domain clusters* module to compute two output vectors $NM(d_i)$ and $DC(d_i)$, respectively. We explain the details of how $NM(d_i)$ and $DC(d_i)$. For now it sufficient to consider $NM(d_i)$ and $DC(d_i)$ as two feature vectors. For each d_i we also compute an *evidence features* vector $EV(d_i)$. Let $v(d_i)$ be a feature vector that combines the $NM(d_i)$, $DC(d_i)$, and $EV(d_i)$ feature vectors. We train the reputation function using the labeled dataset $L = \{(v(d_i), y_i)\}_{i=1..n}$, where $y_i = 0$ if d_i is a known malicious domain name, otherwise $y_i = 1$.

After training is complete; the reputation engine can be used in on-line mode (Figure 5) to assign a reputation score to new domain names. For example, given an input domain name d, the reputation engine computes a score $S \in [0, 1]$. Values of S close to zero mean that d appears to be related to malicious activities and therefore has a low reputation. On the other hand, values of S close to one signify that d appears to be associated with benign Internet services, and therefore has a high reputation. The reputation score is computed as follows. First, d is fed into the *network profiles* module, which consists of five statistical classifiers. The output of the *network profiles* module is a vector $NM(d) = \{c_1, c_2, ..., c_5\}$, where c_1 is the output of the first classifier, and can be viewed as the probability that d belongs to the class of *Popular Domains*, c_2 is the probability that d belongs to the class of *Common Domains*, etc. At the same time, d is fed into the *domain clusters* module, which computes a vector $DC(d) = \{l_1, l_2, ..., l_5\}$. The elements l_i of this vector are computed as follows. Given d, we first extract its network-based features and identify the closest network-based cluster to d, among the network-based clusters computed by the *domain clusters* module during the off-line mode. Then, we extract the zone-based statistical features and identify the zone-based cluster closest to d. Let this closest domain cluster be C_d. At this point, we consider all the zone-based feature vectors $v_j \in C_d$, and we select the subset of vectors $V_d \subseteq C_d$ for which the two following conditions are verified: i) $dist(z_d, v_j) < R$, where z_d is the zone-based feature vector for d, and R is a predefined *radius*; ii) $v_j \in KNN(z_d)$, where $KNN(z_d)$ is the set of k nearest-neighbors of z_d.

The feature vectors in V_d are related to domain names extracted from Notos' *knowledge base*. Therefore, we can assign a label to each vector $v_i \in V_d$, according to the nature of the domain name d from which v_i was computed. The domains in Notos' *knowledge base* belong to different classes. In particular, we distinguish between eight different classes of domains, namely *Popular Domains*, *Common Domains*, *Akamai*, *CDN*, and *Dynamic DNS*, and *Spam Domains*, *Flux Domains*, and *Malware Domains*.

In order to compute the output vector$DC(d)$, we compute the following five statistical features: the *majority class* label L (e.g., L may be equal to *Malware Domain*), i.e., the label that appears the most among the vectors $v_i \in V_d$; the standard deviation of label frequencies, i.e., given the occurrence frequency of each label among the vectors $v_i \in V_d$ we compute their standard deviation; given the subset $V_{(L)d} \subseteq V_d$ of vectors in V_d that are associated with label L, we compute the *mean*, *median* and *standard deviation* of the distribution of distances between z_d and the vectors $v_j \in V_{(L)d}$.

Given a domain d, once we compute the vectors $NM(d)$ and $DC(d_i)$ as explained above, we also compute the evidence vector $EV(d)$. At this point, we concatenate these three feature vectors into a sixteen dimensional feature vector $v(d)$, and we feed $v(d)$ in input to our *trained* reputation function. The reputation function computes a score $S = 1 - f(d)$, where $f(d)$ can be interpreted as the probability that d is a malicious domain name. S varies in the $[0, 1]$ interval, and the lower the value of S, the lower d's reputation.

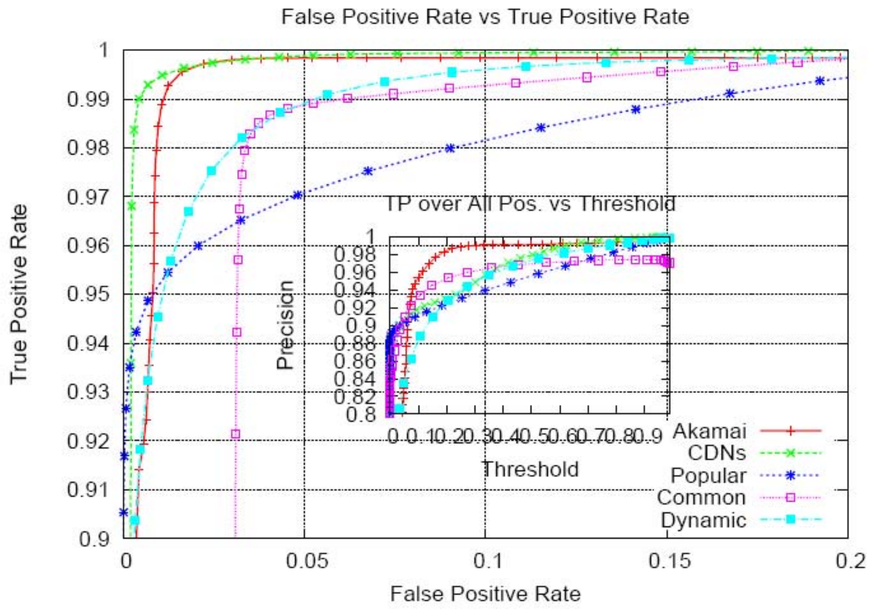


Figure 9: ROC curves for all network profile classes shows the Meta-Classifier's accuracy.

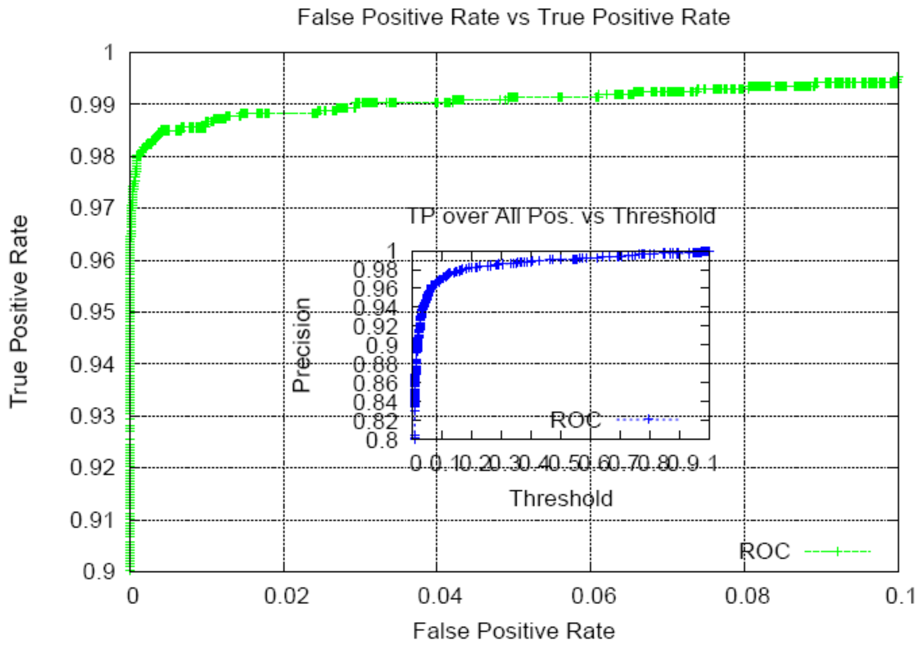


Figure 10: The ROC curve from the reputation function indicating the high accuracy of Notos.

The ROC curves in Figure 9 and Figure 10, show high accuracy of Notos. Detailed evaluation of Notos will be presented in Section 4.

3.1.2 Network-Based Anomaly Detection

BotSniffer and BotMiner We have developed the BotSniffer [13] and BotMiner [14] systems, which are intended for detecting bots in enterprise networks. BotSniffer is an anomaly detection system for detecting centrally controlled botnets. It uses several "spatial and temporal correlation" algorithms to identify a group of hosts that connect to the same outside server with similar traffic and at similar times. BotMiner is a much-improved version of BotSniffer in that it can detect botnets regardless of the structures (e.g., centralized or P2P) and protocols (e.g., IRC, HTTP, or SMTP, etc.) used in command and control. Our botnet detection work is of the highest research quality, with the BotSniffer paper in NDSS 2008 and the BotMiner paper in USENIX Security 2008). Our work also generates a lot of interests from the industry. The BotSniffer system has been widely reported in industry publications. Technical details:

We observed that network activities of bots within the same botnet are correlated with each other and even with their own previous behavior. Thus, we designed BotSniffer as a real-time anomaly detection system that captures the spatial-temporal and correlation properties of C&C activities by bots. Examples of such bot activities include similar connections to same outside server, and similar scan targets and spam messages. We have developed heuristics to recognize such C&C activities in IRC and HTTP traffic. We evaluated BotSniffer using several real network traces and found the results to be very promising. However, a BotSniffer can detect only centralized botnet. We designed BotMiner to be a general-purpose botnet detection system. It finds communication clusters of similar connections based on a set of connection features such as duration, flow sizes, frequency, etc. Bots of the same botnet will belong to the same cluster(s) regardless whether botnet is centralized or P2P because the communications between the bots and the command and control sever(s), or between the bot peers, are similar. BotMiner finds "activities" clusters based on similar (bot) attack-/fraudulent activities (e.g., scan and spam). The two kinds of clusters are then correlated to identify bots. We showed that BotMiner has very good accuracy (very high detection rate and very low false positive rate) in detecting botnets, including P2P botnets. However, the clustering algorithms are computationally intensive, and as a result, BotMiner is not yet suitable for real-time detection. We are working on traffic sampling techniques to scale up the performance of BotMiner.

Scalable Botnet Detection Using Adaptive Sampling and Spatial-Temporal Flow Correlation

We have developed a *botnet-driven* network flow analysis approach that aims to narrow down high volumes of traffic to a limited number of network flows that are most likely related to botnet C&C communications. The sources of such communications are considered highly suspicious hosts, and their traffic can be forwarded to existing packet-based botnet detectors for further, fine-grain analysis. This allows us to significantly reduce the amount of traffic on which DPI is applied, therefore boosting the scalability of existing DPI-based botnet detection systems.

Network flow analysis typically requires far less resource than DPI. However, collecting *precise* network flow information in high-speed networks is challenging, because we may not be able to afford to process every single packet that crosses the network. In order to solve this problem, packet sampling techniques are commonly employed to reduce the number of packets to be processed. For example, *uniform sampling* and its variant *periodic sampling* are among the most popular packet sampling techniques, and allow us to reconstruct *approximate* network flow information. However, the effect of these sampling techniques is that they are able to reconstruct relatively precise information about large flows (i.e., flows that carry a high number of packets), such as media streaming flows, but may poorly approximate or miss outright information about small and medium flows. As many botnet C&C communications are characterized by small- or medium-size flows (i.e., flows that carry a relatively low number of packets), uniform and periodic sampling are not suitable for performing flow-based botnet detection. Therefore, we introduced a novel, *adaptive* sampling technique that is able to reconstruct precise network flow information for flows that are most likely related to botnet C&C communications, while maintaining only approximate information about other flows.

Also, we have developed a new spatial-temporal correlation analysis to identify hosts in a network that share *persistently similar* communication patterns, which is one of the main characteristics of botnets. Our spatial-temporal flow correlation analysis is motivated by the following observation. Because of their (illegal) economy-driven nature, botnets are used by the botmaster for as long as possible to maximize profits (e.g., several months, or until the botnet is dismantled by law enforcement). Therefore, their C&C communications are active for a relatively long (though not necessarily continuous) period of time. Based on these observations, we focus our analysis on identifying hosts in a network that persistently share similar communication patterns for a relatively long (not necessarily continuous) period of time.

We implemented a proof-of-concept version of our system, and evaluated it using real-world legitimate and botnet-related network traces. Our experimental results show that the proposed approach is scalable and can effectively detect bots with few false positives, which can be further reduced by fine-grained botnet detection systems such as BotSniffer and BotMiner.

Below we provide some details of this scalable botnet detection framework.

As shown in Figure 11, our botnet detection framework has three components: **Flow-Capture**, **Flow-Correlation**, and **Fine-Grained Detector**.

The Flow-Capture module aims to monitor the traffic at the edge of high-speed networks to gather network flow information, based on the sampled packets. The Flow-Capture module is divided in two components: **Packet-Sampling** and **Flow-Assembler**. Packet-Sampling is a botnetaware sampling algorithm. Given an overall target sampling probability (SR_{Target}), it samples packets likely related to botnet C&C communications and delivers them to Flow-Assembler, along with their corresponding *instant* sampling probabilities. The Flow-Assembler reconstructs flow information, and assembles the sampled packets into **raw flow**s.

The Flow-Correlation module groups flows output by Flow-Assembler into C-flows. A C-flow is an abstraction introduced in BotMiner to represent the C&C communication patterns of potential bots. Each C-flow represents a view of the communication patterns from a monitored host to a remote service over a certain epoch (e.g, 12 hours). Flow-Correlation applies a scalable clustering algorithm over the C-flows to identify hosts that exhibit similar communication patterns towards machines outside the monitored network. This step is similar to the C-Plane analysis performed by BotMiner, but there are two fundamental differences. First, we use a significantly more efficient flow clustering process, compared to BotMiner, which can handle large traffic volumes typical of high-speed networks. Second, unlike BotMiner, our Flow-Correlation module performs *crossepoch* correlation to identify hosts that show persistently similar communication patterns, a telltale sign of botnets. Any pair of hosts that exhibit persistently similar communication patterns will then be labeled as suspicious hosts (potential bots) and delivered to the Fine-Grained Detector for further in-depth analysis. The Fine-Grained Detector can then focus on monitoring the packets related to only the suspicious IPs provided by our Flow-Correlation module, thus reducing the overall cost of the botnet detection process.

The design and implementation of the Flow-Capture and Flow-Correlation modules and the detection framework are the main contributions of this work. Existing DPI-based botnet detectors can be plugged within our framework with little or no modification to constitute the Fine-Grained Detector module. We developed a Fine-Grained Detector derived from BotMiner and BotSniffer, and we plugged it into our botnet detection framework.

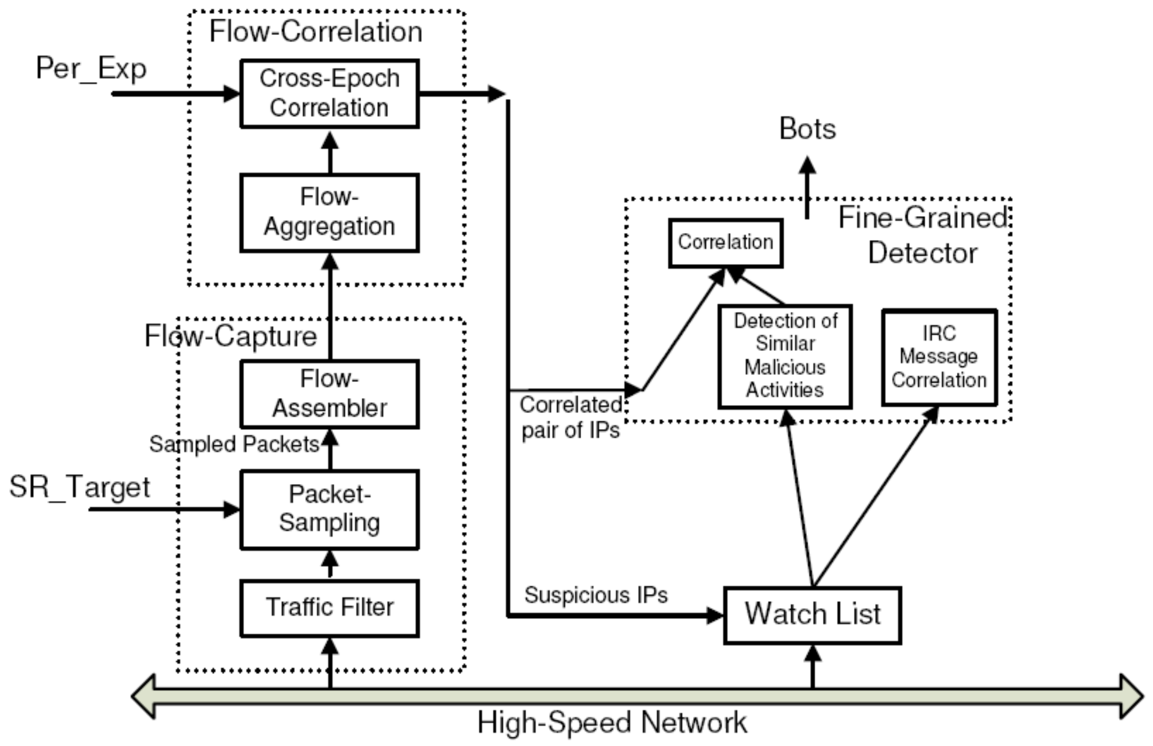


Figure 11: Architectural Overview

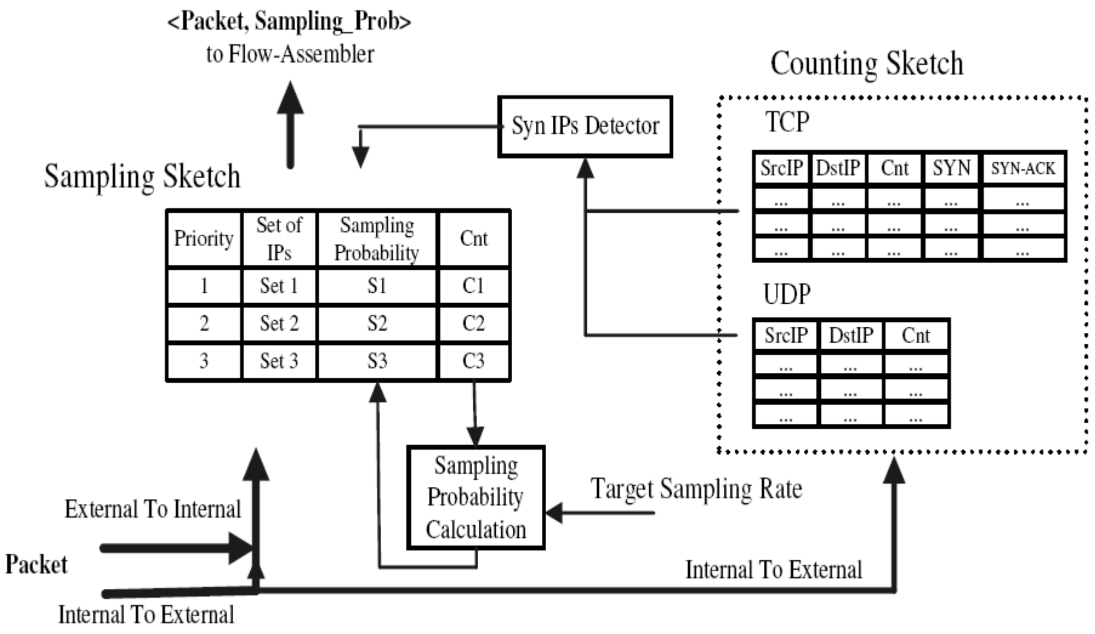


Figure 12: Packet Sampling Architecture

In particular, we used two components: i) an implementation of the malicious activities detector derived from BotMiner's A-Plane monitor, which can identify groups of similar malicious activities based on the attack features (e.g., the scanned port, the exploits or binary content), and ii) BotSniffer's IRC-based botnet detection module. Similar to the Cross-Plane correlation in BotMiner, the correlation component correlates communication patterns and activity patterns to detect bots. Any pair of IPs that share persistently similar communication patterns (generated by Flow-Correlation) and similar malicious activities (generated by the malicious activities detector) are labeled as bots by the correlation component. And any host identified by the BotSniffer's IRC-based botnet detection module will be labeled as bot.

We implemented a prototype system and evaluated it using traces of real-world network traffic and different botnets. The results show that Flow-Capture can achieve a significantly higher sampling rate for botnet-related packets compared to the pre-defined sampling rate. We compared B-Sampling to FlexSample, and the experimental results indicate that B-Sampling outperforms FlexSample regarding sampling rate for botnet packets and detection rate of Flow-Correlation. The cross-epoch correlation can effectively and efficiently identify bots given a small percentage of suspicious hosts. The fine-grained detector can achieve high detection rate and low false positive rate by only inspecting packets related to a small percentage of suspicious hosts.

Table 1: Background traces

Trace	# of Pkts	Dur	Info
Mar25	205,079,914	12h	header
Mar26	280,853,924	24h	header
Mar27	318,796,703	24h	header
Mar28	444,260,179	24h	header
Mar31	102,487,409	1.5h	full

Table 2: Botnet traces

Trace	Dur	Bots
Bot-IRC-A	4days	3
Bot-IRC-B	4days	4
Bot-HTTP-A	4days	3
Bot-HTTP-B	4days	4
Bot-HTTP-C	4days	4
Bot-P2P-Storm	4days	2
Bot-P2P-Waledac	4days	3

We mounted our monitors on a span port mirroring a backbone router at the college network (200Mbps-300Mbps at daytime) to collect data. The traffic covers various applications and we believe such kind of traffic provides good traces to evaluate our system. The dataset contains TCP and UDP headers for continuous 3.5 days and full packets for 1.5 hours in Table 1 We eliminated a B/16 subnet for dynamic IPs allocated for wireless connections, which are frequently changed and can not accurately represent the same hosts for multiple epochs. We observed a total of 1460 different IP addresses in 3.5 days. We also collected 1.5 hour traces with full payload.

We collected the traces of 7 different botnets including IRC-, HTTP- and P2P-based botnets, as described in Table 2. Bot-IRC-A and Bot-HTTP-A were collected by running bot instances ("TR/Agent.1199508.A" and "Swizzor.gen.c") inmultiple hosts in the honeypot. Bot-IRC-B and Bot-HTTP-B/Cwere generated using *Rubot* [19], a botnet emulation framework. In Bot-HTTP-B, bots periodically contacted the C&C server every 10 minutes. And in Bot-HTTP-C, the bots contacted the C&C server in a more stealthy way by adding a random time interval between 0 to 10 minutes on each time of visiting. Both of them conducted scanning attack on receiving the "scan" command. Bots in Bot-IRC-A send packets much more frequently to C&C server in the IRC session, resulting in much larger C&C flows compared to Bot-IRC-B. We collected traces of two P2P-based botnets, *Storm* [14] and *Waledac* [17], by running binaries in the controlled environment.

After aligning the timestamp of each packet in botnet traces according to the time of the first packet in background traces, we mixed 3.5 consecutive days of botnet traces into the college traces by overlaying them to randomly picked client IPs in college network. We took one epoch E as 12hr so there are 7 epochs in total. The filter covers major local DNS, email servers in the college, the IP ranges of the popular service networks (e.g., MICROSOFT, GOOGLE, YAHOO, SUN, etc.), popular content distribution networks (e.g., AKAMAI) , whose IP ranges are unlikely to be used for Botnet C&Cs, and IPs of top 10000 *alexa* domains (corresponding to 12230 IPs).

We evaluated B-Sampling algorithm using the mixed traces with different target sampling rates (0.01, 0.025, 0.05, 0.075 and 0.1). We compared B-Sampling to FlexSample [6], a state-of-theart sampling algorithm that can be configured with different "conditions" for different purposes. FlexSample used a specific condition (Figure 10 in FlexSample [6]) to capture botnet packets by allocating the majority of budgets to packets related to "servers with high indegree of small flows". However, since the number of infected machines could be small in real-world, the "high fan-in" feature may not hold and thus will probably miss the botnet packets. This condition causes very low sampling rates on botnet packets in our traces. Therefore, we modify the condition and only use the condition related to flow size for FlexSample.

Table 3: Sampling rate

SR_T	SR_{Actual}		$SR_{IRC-A/B}$		$SR_{HTTP-A/B/C}$		SR_{Storm}		$SR_{Waledac}$	
	B-	Flex	B-	Flex	B-	Flex	B-	Flex	B-	Flex
0.01	0.012	0.01	0.65/0.68	0.001/0.07	0.55/0.69/0.68	0.06/0.07/0.06	0.02	0.05	0.02	0.07
0.025	0.027	0.025	0.93/0.92	0.002/0.16	0.72/0.93/0.93	0.16/0.17/0.16	0.16	0.11	0.18	0.16
0.05	0.052	0.05	0.96/0.96	0.004/0.32	0.74/0.96/0.96	0.32/0.35/0.33	0.48	0.23	0.48	0.33
0.075	0.076	0.075	0.97/0.97	0.006/0.48	0.75/0.97/0.97	0.50/0.50/0.48	0.72	0.33	0.7	0.48
0.1	0.1	0.1	0.98/0.98	0.008/0.6	0.76/0.98/0.98	0.6/0.64/0.61	0.83	0.41	0.81	0.61

Table 3 presents the overall sampling rates and sampling rates for botnet-related packets on the mixed dataset, using both B-Sampling and FlexSample. The first column (SR_T) reports the pre-defined target sampling rates we experimented with. The second column (SR_{Actual},B) and the third column (SR_{Actual},Flex) report the actual overall sampling rates achieved by B-Sampling and FlexSample. The results show that both B-Sampling and FlexSample keep the actual sampling rate close to the target sampling rate. The remaining columns report the sampling rates related to different types of botnet-related packets, where we "zoom" in the sampled packets and evaluate the actual sampling rates for packets of each botnet. For example, the 4th column ($SR_{IRC-A/B}$,B) reports the actual sampling rate for packets in Bot-IRC-A and Bot-IRC-B using B-Sampling, whereas the 5th column ($SR_{IRC-A/B}$,Flex) presents the sampling rate using FlexSample. We can find that B-Sampling captures a higher percentage of botnet packets, compared to FlexSample. For example, considering the second row (target sampling rate is 0.025), B-Sampling achieves a sampling rate of 0.93 ($SR_{IRC-A/B}$, B column) while FlexSample achieves that of 0.002 ($SR_{IRC-A/B}$, Flex column) for packets in `Botnet-IRC-A`, where the C&C flows are large flows. The remaining columns report a comparison of B-Sampling and FlexSampling on the sampling rates for other botnets. As we can see, B-Sampling achieves higher sampling rate for botnet-related packets, compared to FlexSample. It is possible to increase the flow size in the FlexSample condition or reduce the budget for small flows to make FlexSample capture more packets in `Botnet-IRC-A`. However, it will cause FlexSample to decrease the sampling rates for packets related to botnets whose C&Cs are small flows such as `Bot-HTTP-` and `Bot-P2P-`. The reason is that the feature of flow size and server indegree are not intrinsic for botnets and different botnets can diverse greatly regarding these features. B-Sampling gave higher sampling rate for packets in `Bot-IRC-` and `Bot-HTTP-` than those in `Bot-P2P-`, because that the number of packets related to syn-server is much smaller than that related to syn-clients, and thus syn-servers have higher priority.

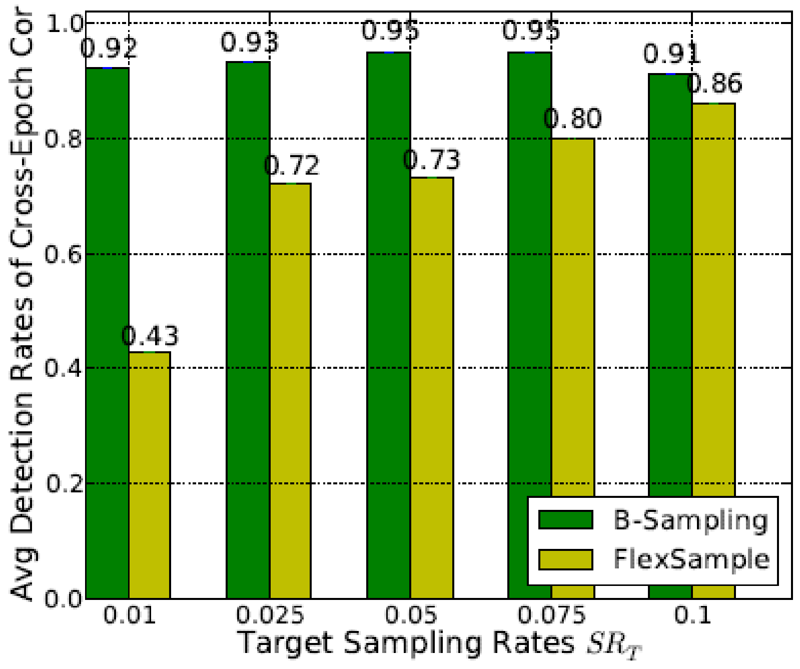


Figure 13: Average detection rate for cross-epoch correlation, over different PerExp

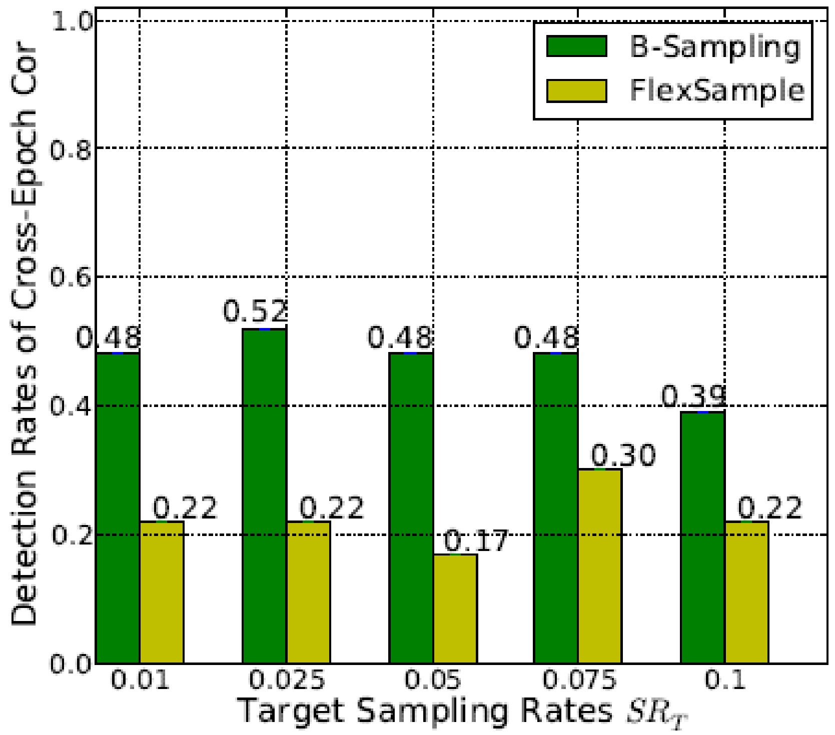


Figure 14: Detection rate for cross-epoch correlation, PerExp = 0.01

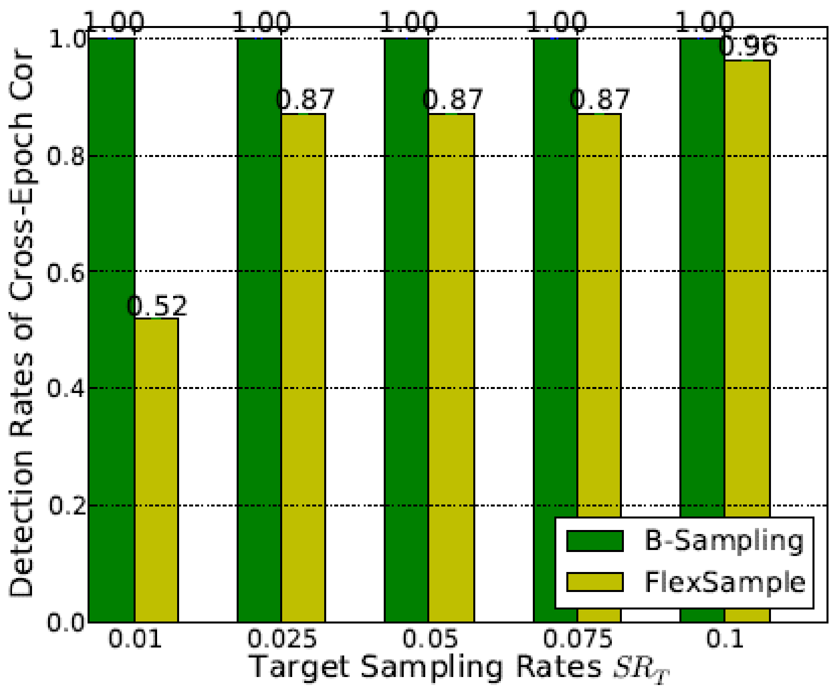


Figure 15: Detection rate for cross-epoch correlation, PerExp = 0.05

Table 4: Detection Rates of Cross-Epoch Correlation using B-Sampling

SR_T	For each Per_{Exp}, TP(bots/23), FP(noises/1460)									
	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
0.01	48%, 0.1%	83%, 0.5%	96%, 1%	96%, 2%	100%, 3%	100%, 4%	100%, 5%	100%, 6%	100%, 6%	100%, 8%
0.025	52%, 0%	87%, 0.5%	100%, 1%	100%, 2%	100%, 3%	100%, 4%	100%, 5%	100%, 6%	100%, 7%	100%, 8%
0.05	48%, 0.1%	100%, 0.3%	100%, 1%	100%, 2%	100%, 3%	100%, 4%	100%, 5%	100%, 5%	100%, 7%	100%, 7%
0.075	48%, 0.2%	100%, 0.3%	100%, 1%	100%, 2%	100%, 3%	100%, 4%	100%, 5%	100%, 6%	100%, 7%	100%, 8%
0.1	39%, 0.3%	78%, 0.8%	100%, 1%	100%, 2%	100%, 3%	100%, 3%	100%, 5%	100%, 5%	100%, 7%	100%, 8%
1	30%, 0.5%	65%, 0.8%	96%, 1%	100%, 2%	100%, 3%	100%, 4%	100%, 5%	100%, 5%	100%, 7%	100%, 8%

We evaluated the cross-epoch correlation with B-Sampling using the mixed traces for two properties, detection accuracy and scalability. We set

$$M = \llcorner \frac{N}{2} \lrcorner \; (N = 7, M = 3)$$

which means that two hosts sharing similar communication patterns for any 3 out of 7 epochs will be labeled as suspicious.

Given SR_T and Per_{Exp}, each cell shows the detection rate of bots(/23) and percentage of noises(/1460) identified by Flow-Correlation using B-Sampling (see Table 4). The results show that Flow-Correlation can achieve high detection rate with low Per_{Exp}. For example, with $Per_{Exp} \geq 5\%$, for all the SR_T evaluated, Flow-Correlation can successfully identify all the bots. While for the very low Per_{Exp} (e.g., 2% and 3%), more than half of the bots were still captured. Figure 13 illustrates the average detection rates over different Per_{Exp} for each target sampling rate, and Figure 14 and Figure 15 present the detection rates using B-Sampling and FlexSample with PerExp of 0.01 and 0.05. The comparison results show that by using B-Sampling, Flow-Correlation can achieve higher detection rate.

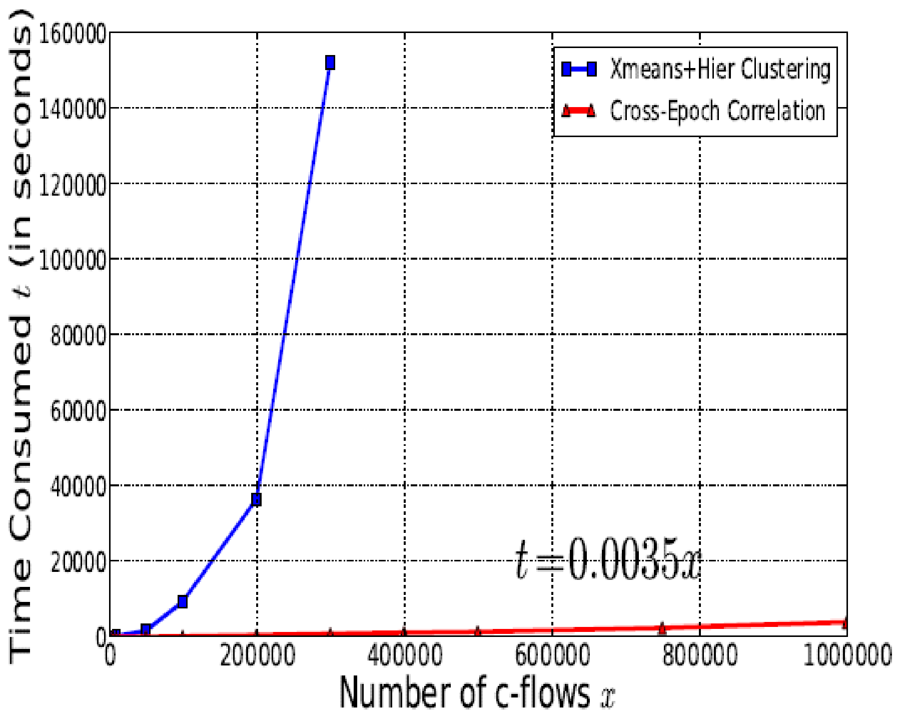


Figure 16: Scalability of Cross-Epoch Correlation

Figure 16 presents the time consumption (in a 4Gmemory and 2-core CPU computer) for crossepoch correlation and the C-Plane clustering of BotMiner as the number of C-flows increases. We configured `Birch` to run MaxRound = 50 to simulate the process of identifying up to PerExp suspicious hosts. The exponential time increment for C-Plane clustering of BotMiner indicates its limited scalability. The cross-epoch correlation shows linear pattern and its linear regression model is t = 0.0035x.

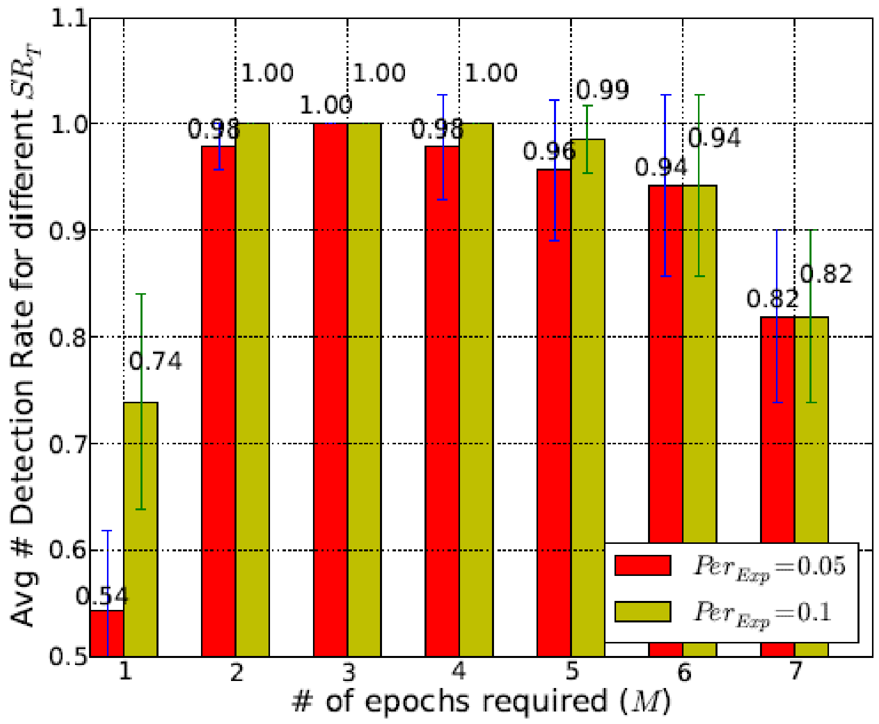


Figure 17: Avg detection rate (over SRT s) of Cross-Epoch Correlation using B-Sampling

Figure 17 presents the mean and standard deviation for detection rates by Flow-Correlation with B-Sampling for different M, given Per_{Exp} (5% or 10%) for all SR_T . First, the results demonstrate the effectiveness of cross-epoch correlation. When no cross-epoch correlation is used (M = 1), many legitimate IPs show stronger similarity than bots in a single epoch. Therefore, given a certain Per_{Exp}, more than 50% bots are missed. While cross-epoch correlation can effectively eliminate these legitimate IPs that show strong similarity in one epoch but do not have persistently similar patterns. For example, cross-epoch correlation with M = 2 can successfully detect most bots. Second, the results indicate that cross-epoch correlation is not sensitive to the value of M. For example, for M = 3/4/5, the cross-epoch correlation achieves similar detection rate. Such observation also indicates that N/2 is a good value for M.

Table 5: Detection Rates of Fine-Grained Detectors

SR_T	For each Per_{Exp}, TP(bots/23), FP(noises/1460)									
	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
0.01	48%, 0	83%, 0	96%, 0	96%, 0	100%, 0	100%, 0	100%, 0	100%, 0	100%, 0	100%, 0
0.025	52%, 0	87%, 0	100%, 0	100%, 0	100%, 0	100%, 0	100%, 0	100%, 0	100%, 0	100%, 0
0.05	48%, 0	100%, 0	100%, 0	100%, 0	100%, 0	100%, 0	100%, 0	100%, 0	100%, 0	100%, 0
0.075	48%, 0	100%, 0	100%, 0	100%, 0	100%, 0	100%, 0	100%, 0	100%, 0	100%, 0	100%, 0
0.1	39%, 0	78%, 0	100%, 0	100%, 0	100%, 0	100%, 0	100%, 0	100%, 0	100%, 0	100%, 0
1	30%, 0	65%, 0	96%, 0	100%, 0	100%, 0	100%, 0	100%, 0	100%, 0	100%, 0	100%, 0

Fine-grained botnet detector inspects all the packets related to suspicious IPs detected by Flow-Correlation. Using 1.5hr trace mixed with botnet traces, we evaluated the detection rate and performance of the fine-grained detector.

By analyzing the similarity among IRC messages, "IRC Message Correlation" component in our detector detected bots in `Bot-IRC-A/B`. Other bots were detected by the "Correlation" component. For example, Bots in `Bot-HTTP-B/C` trigger alerts when they scan the local network. `Bot-HTTP-A` bots trigger alerts when they make update requests. `Storm` and `Waledac` trigger alerts when they discover peers. These bots were detected by correlating such activities/alerts with corresponding pairs of IPs from Flow-Correlation. Table 5 presents the detection rates and false positive rates for the fine-grained detector for different SR_T s and Per_{Exp}s. For most combinations of SR_T and Per_{Exp}, our framework can reduce traffic volume by more than 90% for fine-grained detector but still keep high detection rates and low false positives. For example, for $SR_T = 0.01$ and $Per_{Exp} = 0.05$, the fine-grained detector can detect all bots with false positive of 0, and it only needs to focus on 1.7% percentage of packets.

Table 6: Performance of Fine-Grained Detector

	With Flow-Corr $(Per_E = 5\%, M = 3)$						direct
SR_T	0.01	0.025	0.05	0.075	0.1	1	
Per of Pkts	1.7%	2.9%	2.1%	3%	4.3%	2%	100%
$Time$	33s	39s	35s	40s	49s	33s	858s

Table 6 presents the performance comparison, including the percentage of packets inspected and the processing time of the fine-grained detector in two situations: i) the detector is directly applied, ii) the detector is applied with Flow-Correlation and B-Sampling ($Per_{Exp} = 0.05$ and $M = 3$). By using Flow-Correlation, fine-grained detector to reduce 95% time to process off-line traces, indicating a great workload reduction in real time.

Detection of Obscure Botnet Command and Control Channels We studied the problem of identifying obscure chat-like botnet command and control (C&C) communications, which are indistinguishable from human-human communication using traditional signature-based techniques. Existing passive-behavior-based anomaly detection techniques are limited because they either require monitoring multiple bot infected machines that belong to the same botnet or require extended monitoring times. We explored the potential use of active botnet probing techniques in a network middlebox as a means to augment and complement existing passive botnet C&C detection strategies, especially for small botnets with obfuscated C&C content and infrequent C&C interactions. We present an algorithmic framework that uses hypothesis testing to separate botnet C&C dialogs from human-human conversations with desired accuracy and implement a prototype system called BotProbe. Experimental results on multiple real-world IRC bots demonstrated that our proposed active methods can successfully identify obscure and obfuscated botnet communications. A realworld user study on about one hundred participants also showed that the technique has a low false positive rate on human-human conversations. This work was published in the 2009 Annual Computer Security Applications Conference [15].

httprecon We have developed a novel, network-level behavioral malware clustering system. We focused on the analysis of structural similarities among malicious `http` traffic traces generated by `http`-based malware. The proposed clustering system is able to unveil similarities (or relationships) among malware samples that may not be captured by system-level behavioral clustering, thus offering a new point of view and valuable information to malware analysts. Also, we defined the similarity metrics among `http` traffic traces, and tailored our clustering algorithms so that the output of our system provides a good input for algorithms that automatically generate network signatures. Such network signatures can in turn be deployed to an Intrusion Detection System (IDS) located at the edge of a network in order to detect malicious outbound `http` requests, which are a symptom of infection. We implemented a proof-of-concept version of our network-level malware clustering system, and performed experiments withmore than 19,000malware samples. Our experimental results confirmed the effectiveness of the proposed clustering approach, and showed how the process of automatically generating network signatures for detecting malware-compromised machines can benefit from it. This work was published in the 7th USENIX Symposium on Networked Systems Design and Implementation (NSDI), in April 2010 [21].

3.2 Botnet Analysis

3.2.1 Malware Analysis

Ether We have developed a new and transparent malware analysis platform. The focal point in the malware analysis battle is how to detect versus how to hide a malware analyzer from malware in runtime. The state-of-the-art analyzers reside in or emulate part of the guest operating system and its underlying hardware, making them easy to detect and evade. We developed a transparent and external approach to malware analysis, which is motivated by the intuition that for a malware analyzer to be transparent, it must not induce any side-effects that are unconditionally detectable by malware. Our analyzer, Ether, is based on a novel application of hardware virtualization extensions such as Intel VT, and resides completely outside of the target OS environment. Thus, there are no in-guest software components vulnerable to detection, and there are no shortcomings that arise from incomplete or inaccurate system emulation. Our experiments were based on our study of obfuscation techniques used to create 25,000 recent malware samples. The results showed that Ether remains transparent and defeats the obfuscation tools that evade existing approaches. We also demonstrated that with coarse-grained tracing (e.g., system-call level tracing), Ether incurs negligible overhead, and with fine-grained tracing (e.g., instruction-level tracing), Ether can be significant depending on the analysis tasks. We are working on scaling the performance of Ether. A paper on Ether was published in the 15th ACM Conference on Computer and Communications Security (CCS 2008) [11]

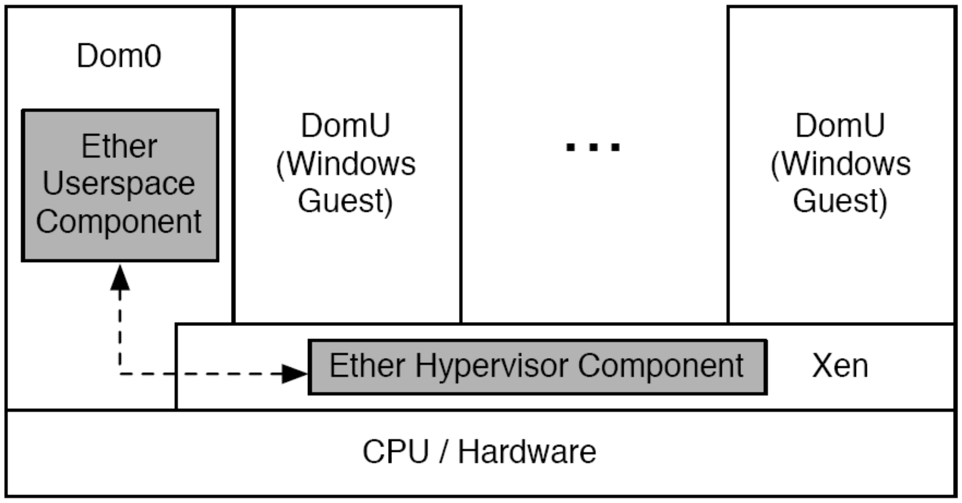


Figure 18: Ether's system architecture.

Below we highlight the system architecture of Ether as well as the evaluation results.

As shown in Figure 18, the architecture of Ether consists of a hypervisor component and a userspace component running in domain 0. Ether's hypervisor component is responsible for detecting events in the analysis target. Currently, such events include system call execution, instruction execution, memory writes, and context switches. Ether's user space component acts as a controller regulating which processes and events in the guest should be monitored. This component also contains logic to derive semantic information from analyzed events, such as translating a system call number into system call name or displaying system call argument content based on argument data type.

The analysis target consists of a Xen domU running Windows XP Service Pack 2. The only change we made to the default installation of Windows XP was disabling PAE and large memory pages. These modifications exist solely to make memory write detection easier in the initial Ether implementation and are not a limitation of our approach.

We developed two tools based on Ether: EtherUnpack and EtherTrace. EtherUnpack traces memory writes and single instructions (i.e., fine-grained tracing), while EtherTrace traces system calls (i.e., coarse-grained tracing). We used these tools to evaluate Ether and compare it against current approaches.

To compare EtherUnpack against current approaches, we tested it alongside two other automated unpacking tools. These were PolyUnpack [22], a generic in-guest unpacking tool and Renovo [18], an external unpacker based on the BitBlaze Binary Analysis Platform [3]. The results of our testing are shown in Table 7 and Table 8.

As shown in Table 7 and Table 8, EtherUnpack was able to reveal hidden code from tElock, the CERTLEXSI sample, Armadillo [2], Obsidium, and Themida [4] because it does not rely on correct CPU emulation, and instead utilizes native hardware for instruction execution. Ether was also able to trace the PCPrivacyTool and our synthetic QEMU [8] detection sample without issue. Since it is inherently impossible to ensure the equivalence of an emulated processor to a real processor, more emulation inconsistencies are likely present in QEMU. Therefore, as an alternative to QEMU, we advocate the use of hardware virtualization-based approaches such as Ether.

The results indicate once again that Armadillo, which is quite popular in current malware, provides strong anti-analysis protections and detected both Anubis [1] and Norman Sandbox [5]. Besides Armadillo, Anubis failed to trace tElock, which crashed after reporting failure of an internal CRC check. Reasons why Anubis failed to trace ASProtect or why Norman Sandbox failed to trace yoda's Protector are unclear. In contrast to both Anubis and Norman, EtherTrace successfully traced all samples.

Automatic Reverse Engineering of Malware Emulators Malware authors have recently begun using emulation technology to obfuscate their code. They convert native malware binaries into bytecode programs written in a randomly-generated instruction set and paired with a native binary emulator that interprets the bytecode. No existing malware analysis technique can reliably defeat this obfuscation technique. We developed the first approach in automatic reverse engineering of malware emulators. Our algorithms are based on dynamic analysis. We execute the emulated malware in a protected environment and record the entire x86 instruction trace generated by the emulator.

Table 7: Effectiveness of generic unpackers.

Packing Tool	EtherUnpack	Renovo	PolyUnpack
Armadillo	yes	no	no
Aspack	yes	yes	no
Asprotect	yes	yes	yes
FSG	yes	yes	yes
MEW	yes	yes	yes
Molebox	yes	yes	no
Morphine	yes	yes	yes
Obsidium	yes	no	no
PECompact	yes	yes	no
Themida	yes	yes	no
Themida VM	yes	no	no
UPX	yes	yes	yes
UPX S	yes	yes	yes
WinUPack	yes	yes	no
yoda's Prot.	yes	yes	no

Table 8: Effectiveness of sandboxing environments.

Packing Tool	EtherTrace	Anubis	Norman
None	yes	yes	yes
Armadillo	yes	no	no
UPX	yes	yes	yes
Upack	yes	yes	yes
Themida	yes	yes	yes
PECompact	yes	yes	yes
ASPack	yes	yes	yes
FSG	yes	yes	yes
ASProtect	yes	no	yes
WinUpack	yes	yes	yes
tElock	yes	no	yes
PKLITE32	yes	yes	yes
yoda's Prot.	yes	yes	no
NsPack	yes	yes	yes
MEW	yes	yes	yes
nPack	yes	yes	yes
RLPack	yes	yes	yes
RCryptor	yes	yes	yes

We then use dynamic data-flow and taint analysis over the trace to identify data buffers containing the bytecode program and extract the syntactic and semantic information about the bytecode instruction set. With these analysis outputs, we are able to generate data structures, such as control-flow graphs (CFG), that provide the foundation for subsequent malware analysis. We implemented a proof-of-concept system called Rotalume and evaluated it using both legitimate programs and malware emulated by VMProtect and Code Virtualizer. The results showed that Rotalume accurately revealed the syntax and semantics of the emulated instruction sets and reconstructed execution paths of the original programs from their bytecode representations. This work was published in the 2009 IEEE Symposium on Security and Privacy, and won the best student paper award [23].

3.2.2 Botnet Traceback and Attribution

We have developed several techniques to analyze (historical) DNS query data, network traces, messages in on-line forums (e.g. IRC), and correlate to identify and locate botmasters. We have used these techniques to help law enforcement agencies. For example, we were members of the Mariposa Working Group (MWG), which performed attribution and traceback on the so-called Mariposa botnet. In the end, several groups were arrested in several countries. While many of the suspects still face trial, one public piece of information about the MWG was noted by Director Mueller at the August 5, 2010 International Conference on Cyber Security:

Law enforcement agencies alone cannot defeat our cyber adversaries. In the Mariposa case, our private sector partners also provided valuable help. The Mariposa Working Group, an informal band of security researchers and volunteers, gave us intelligence to track down the subjects, and worked to dismantle the botnet after we made our arrests. Thus far, our botnet traceback and attribution techniques are still mostly manual. In the near future, we will research the creation of *automated* approaches.

3.3 Botnet Mitigation

We have developed a system for host-based forensics analysis and recovery. These tools run from a bootable CD that can be used to boot the infected machine into a secure environment for analysis and clean-up. We (in particular, Damballa) have developed several DNS redirection techniques to disrupt botnet C&C. These include DNS sinkholing where the resolved IP is pointed to a sinkhole, and DNS reset where the resolved IP is the local loopback address. Damballa has incorporated these techniques in their products and services.

We (in particular, Support Intelligence) have continued to develop our network prophylactic for ISPs. Using DNSRBL lists to identify address to provide specific routes into a network device that does further deep packet inspection (DPI) to assess the problem and advise customer service fraud department or technical support with advise on malicious traffic. Routes are injected into the ISP core to route suspicious traffic through analysis engine. Support Intelligence also worked on BGP monitoring with algorithms to track routes for networks with significant botnet telemetry emanations.

3.4 Add-on Tasks

We have successfully produced a report detailing the lessons learned in detecting and responding to the Conficker botnets. The report is published as:

• ConfickerWorking Group: Lessons Learned. `http://www.confickerworkinggroup.org/wiki/uploads/Conficker Working Group Lessons Learned 17 June 2010 final.pdf`. January 2011.

With the Internet Law Group, we also explored the regulatory frameworks that permit civil investigation of the financial structure, organization and management of botnets. The work is still on-going because it takes a long time for our study, which must include real investigation cases, to conclude. We hope to be able to report our results in the near future.

With the help of DREN, we have developed spam and botnet data analysis systems for identifying compromised government assets, clustering analysis of DNS reputation, and DNS Weather Map data. These systems are now used on a daily basis by DREN operators.

3.5 Commercialization

Damballa was founded in 2006. The PI Wenke Lee and co-PI David Dagon were among the cofounders. Damballa participated in this project, and has incorporated Ether into their malware analysis system, Notos into their DNS-based monitoring product, and DNS sinkholing and reset into their products. Damballa has also developed a network-based anomaly detection based on BotMiner. Damballa now has over 50 employees and several dozens of customers. Several customers, including Comcast and Vanderbilt University have gone public with their anti-botnet strategies based on Damballa products and services.

4.0 RESULTS AND DISCUSSION

Results of various tasks are in Section 3. We especially present the evaluation of a dynamic reputation system for DNS and discuss the most significant results of DNS monitoring, network-based anomaly detection and malware analysis in this section.

4.1 Evaluation of Notos

As a first step, we computed vectors based on the statistical features from 250,000 unique RRs. These vectors were computed based on historic passive DNS information from the last two weeks of DNS traffic observed on the same two ISP recursive resolvers in Atlanta and San Jose. The accuracy of the Meta-Classification system (see Figure 6 (a)) in the network profile module is critical for the overall performance of Notos. This is because, in the on-line mode, Notos will receive unlabeled vectors which must be classified and correlated with what is already present in our knowledge base. For example, if the classifier receives a new RR and assigns to it the label Akamai with very high confidence, that implies the RR which produced this vector will be part of a network similar to Akamai. However, this does not necessarily mean that it is part of the actual Akamai CDN. We will see next how we can draw conclusions based on the proximity between labeled and unlabeled RRs within the same zone-based clusters. Furthermore, we discuss the accuracy of the Meta-Classifier when modeling each different network profile class. OurMeta-Classifier consists of five different classifiers, one for each different class of domains we model. We chose to use a Meta-Classification system instead of a traditional single classification approach becauseMeta-Classification systems typically perform better than a single statistical classifier [9] [16]. Throughout our experiments this proved to be also true. The ROC curve in Figure 9, shows that the Meta-Classifier can accurately classify RRs for all different network profile classes.

The training dataset for the Meta-Classifier is composed of sets of 2,000 vectors from each of the five network profile classes. The evaluation dataset is composed of 10,000 vectors, 2,000 from each of the five network profile classes. The classification results for the domains in the Akamai, CDN, dynamic DNS and Popular classes showed that the supervised learning process in Notos is accurate, with the exception of a small number of false positives related to the Common class (3.8%). After manually analyzing these false positives, we concluded that some level of confusion between the vectors produced by Dynamic DNS domain names and the vectors produced by domain names in the Common class still remains. However, this minor misclassification between network profiles does not significantly affect the reputation function. This is because the zone profiles of the Common and Dynamic DNS domain names are significantly different. This difference in the zone profiles will drive the network-based and zone-based clustering steps to group the RRs from Dynamic DNS class and Common class in different zone-based clusters.

Despite the fact that the network profile modeling process provides accurate results, it doesn’t mean this step can independently designate a domain as benign or malicious. The clustering steps will assist Notos to group vectors not only based their network profiles but also based on their zone properties. In the following we show how the network and zone profile clustering modules can better associate similar vectors, due to properties of their domain name structure. In the domain name clustering process, Figure 6(b)) we used X-Means clustering in series, once for the network-based clustering and again for the zone-based clustering. In both steps we set the minimum and maximum number of clusters to one and the total number of vectors in our dataset, respectively. We run these two steps using different numbers of zone and network vectors.

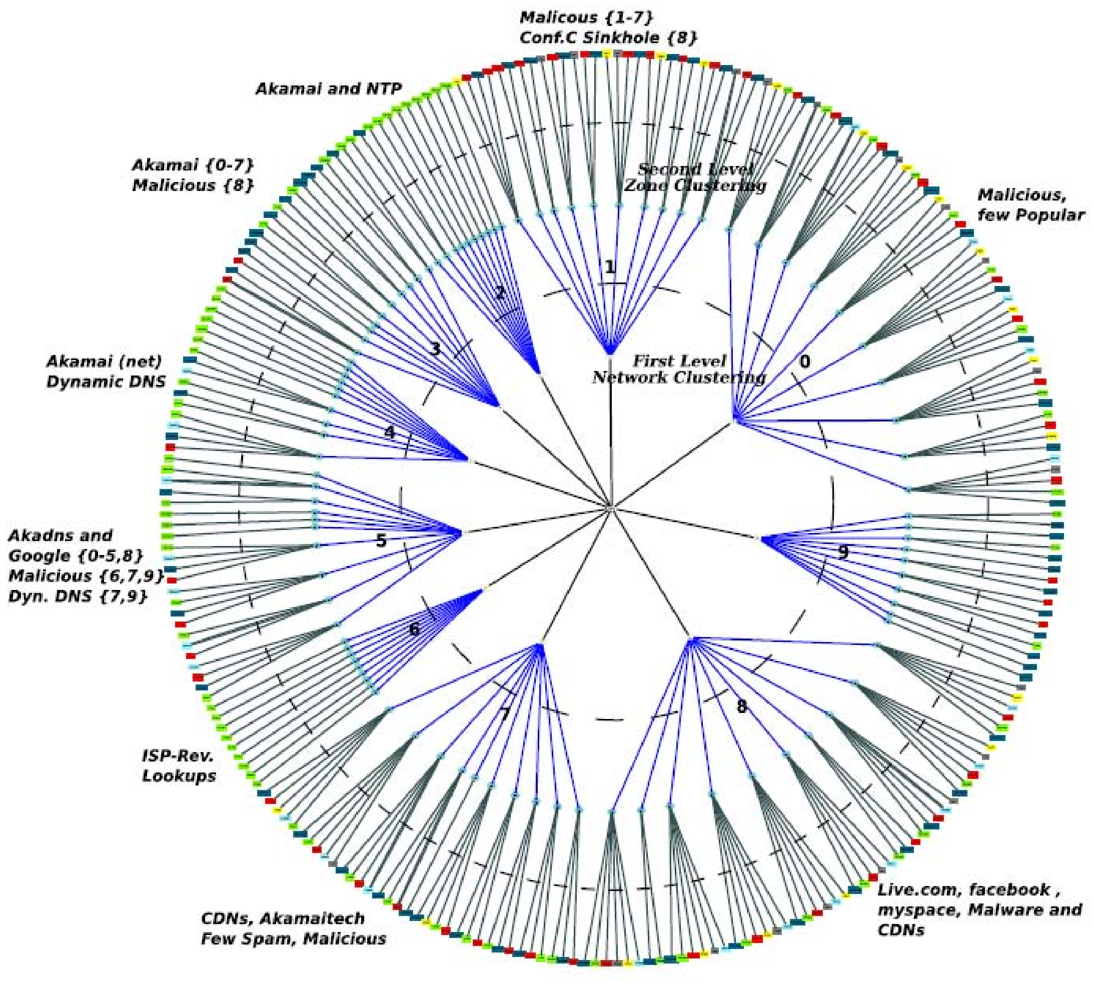


Figure 19: With the 2-step clustering step, Notos is able to cluster large trends of DNS behavior.

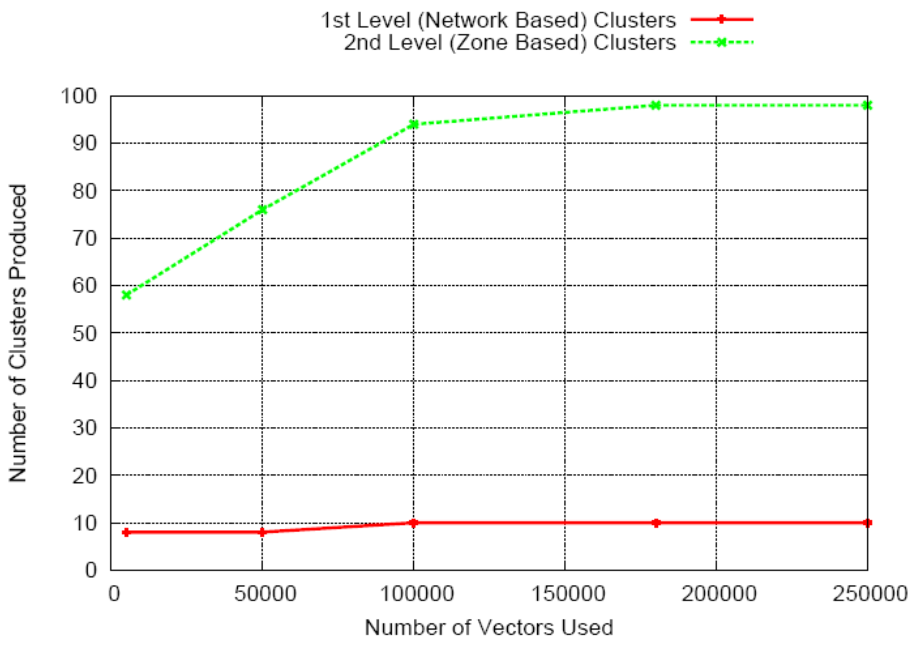


Figure 20: By using different number of network and zone vectors, we observe that after the first 100,000, there is no significant variation in the absolute number of produced clusters during the 1st and 2nd level clustering steps.

Figure 20 shows that after the first 100,000 vectors are used, the number of network and zone clusters remains fairly stable. This means that by computing at least 100,000 network and zone vectors—using a 15-day old passive DNS database—we can obtain a stable population of zone and network based clusters for the monitored network. We should note that reaching this network and cluster equilibrium does not imply that we do not expect to see any new type of domain names in the ISP's DNS recursive. This just denotes that based on the RRs present in our passive DNS database, and the daily traffic at the ISP's recursive, 100,000 vectors are enough to reflect the major network profile trends in the monitored networks. Figure 20 indicates that a sample set of 100,000 vectors may represent the major trends in a DNS sensor. It is hard to safely estimate the exact minimum number of unique RRs that is sufficient to identify all major DNS trends. An answer to this should be based upon the type, size and utilization of the monitored network. Without data from smaller corporate networks it is difficult for us to make a safe assessment about the minimum number of RR necessary for reliably training Notos. The evaluation dataset we used consisted of 250,000 unique domain names and IPs. The cluster overview is shown in Figure 19. In the following paragraphs we discuss some interesting observations that can be made from these network-based and zone-based cluster assignments. As an example, network clusters 0 and 1 are predominantly composed of zones participating in fraudulent activities like spam campaigns (yellow) and malware dropping or C&C zones (red). On the other hand, network clusters 2 to 5 contain Akamai, dynamic DNS, and popular zones like Google, all labeled as benign (green). We included the unlabeled vectors (blue) based on which we evaluated the accuracy of our reputation function. We have a sample of unlabeled vectors in almost all network and zone clusters. We will see how already labeled vectors will assist us to characterize the unlabeled vectors in close proximity.

Before we describe two sample cases of dynamic characterization within zone-based clusters, we need to discuss our radius R and k value selection. After looking into the distribution of Euclidean distances between unlabeled and labeled vectors within the same zone clusters, we concluded that in the majority of these cases the distances were between 0 and 1000. We tested different values of the radius R and the value of k for the K-nearest neighbors (KNN) algorithm. We observed that the experiments with radius values between 50 and 200 provided the most accurate reputation rating results. We also observed that if $k > 25$ the accuracy of the reputation function is not affected for all radius values between 50 and 200. Based on the results of these pilot experiments, we decided to set k equal to 50 and the radius distance equal to 100.

Figures 13 and 14 show the effect of this radius selection on two different types of clustering problems. In Figure 21, unknown RRs for `akamaitech.net` are clustered with a labeled vector `akamai.net`. CDNs such as Akamai tended to have new domain names with each RR, but to also reuse their IPs. By training with only a small set of labeled `akamai.net` RRs, our classifier put the new, unknown RRs for `akamaitech.net` into the existing Akamai class. IP-specific features therefore brought the new RRs close to the existing labeled class. Figure 21 compresses all of the dimensions into a two-dimensional plot (for easier visual representation), but it is clear the unknown RRs were all within a distance of 100 to the labeled set.

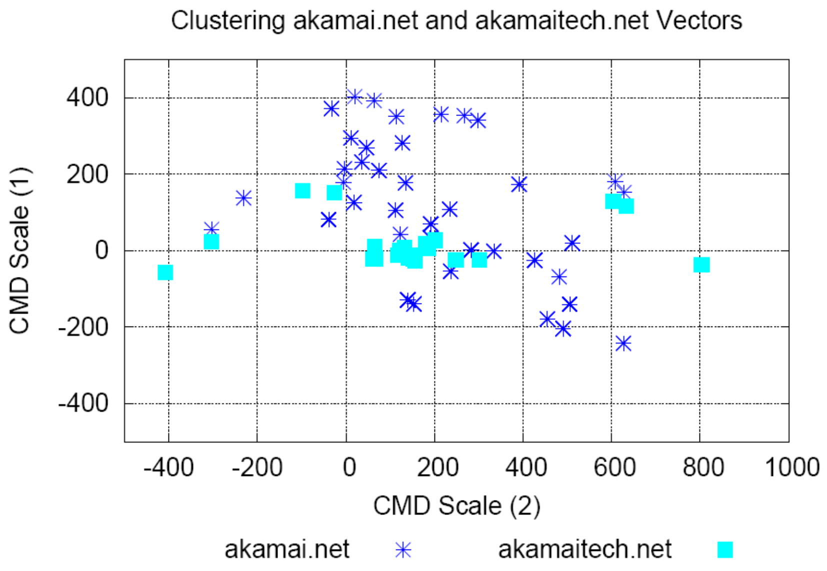


Figure 21: An example of characterizing the akamaitech.net unknown vectors as benign based on the already labeled vectors (akamai.net) present in the same cluster.

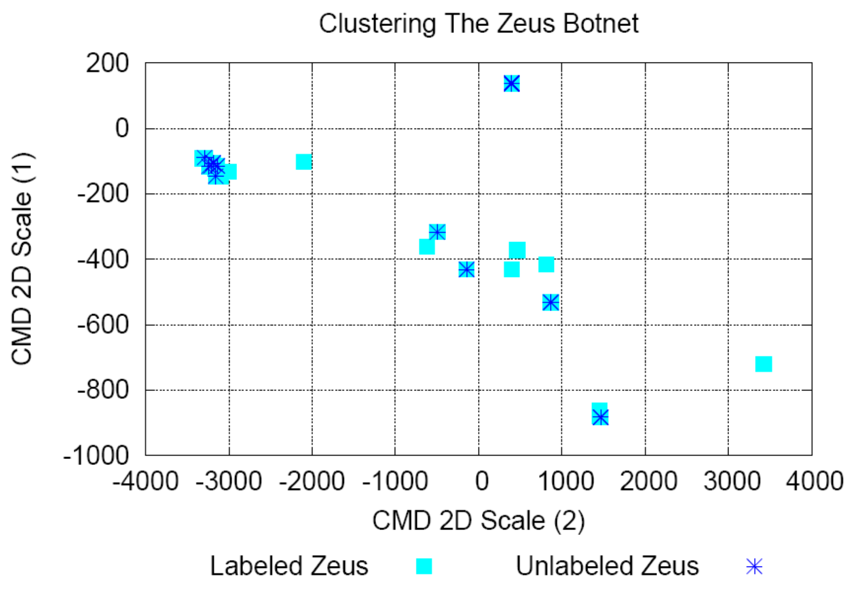


Figure 22: An example of how the Zeus botnet clusters during our experiments. All vectors are in the same network cluster and in two different zone clusters.

This result validates that just a few weeks' worth of labeled data was necessary for training. Thus, one does not have to exhaustively discover all whitelisted domains. Notos is resilient to changes in the zone classes we selected. Services like CDNs and major web sites can add new IPs or adjust domain formats, and these will be *automatically* associated with a known labeled class.

The ability of Notos to associate new RRs based on limited labeled inputs is demonstrated again in Figure 22. In this case, labeled Zeus domains (approximately 2,900 RRs from three different Zeus-related BLs) were used to classify new RRs. Figure 22 plots the distance between the labeled Zeus-related RRs and new (previously unknown) RRs that are also related Zeus botnets. Most of the new (unlabeled) Zeus RRs lay very close, and often even overlap, to known Zeus RRs (see Table 9 Table 10). This is a good result, because Zeus botnets are notoriously hard to track, given the botnet's extreme agility. Tracking systems such as `zeustracker.abuse.ch` and `malwaredomainlist.com` have limited visibility into the botnet, and often produce disjoint blacklists. Notos addresses this problem, by leveraging a limited amount of training data to correctly classify new RRs. During our evaluation set, Notos correctly detected 685 new (previously unknown) Zeus RRs.

We decide to use a Decision Tree using Logit-Boost strategy (LAD) as the reputation function. Our decision is motivated by the time complexity, the detection results and the precision (true positives over all positives) of the classifier. We compared the LAD classifier to several other statistical classifiers using a typical model selection procedure [12]. LAD was found to provide the most accurate results in the shortest training time for building the reputation function. As we can see from the ROC curve in Figure 10, the LAD classifier exhibits a low false positive rate (FP%) of 0.38% and true positive rate (TP%) of 96.8%. It is was noting that these results were obtained using 10-fold cross-validation, and the detection threshold was set to 0.5. The dataset using for the evaluation contained 10,719 RRs related to 9,530 *known bad* domains. The list of *known good* domains consisted of the top 500 most popular domains according to Alexa.

Table 9: Sample cases form Zeus domains detected by Notos and the corresponding days that appeared in the public BLs. All evidence information in this table were harvested from zeustracker.abuse.ch.

Domain Name	IP	Date
google-bot004.cn	213.182.197.229	08-15
analf.net	222.186.31.169	08-15
pro-buh.ru	89.108.67.83	08-15
ammdamm.cn	92.241.162.55	08-15
briannazfunz.com	95.205.116.55	08-15
mybank-of.com	59.125.229.73	08-15
oc00co.com	212.117.165.128	08-15
avangadershem.com	195.88.190.29	08-19
securebizccenter.cn	122.70.145.140	08-19
adobe-updating-service.cn	59.125.231.252	09-02
0md.ru	219.152.120.118	09-19
avrev.info	98.126.15.186	09-27
g00glee.cn	218.93.202.100	09-02

Table 10: Anecdotal cases of malicious domain names detected by Notos and the corresponding days that appeared in the public BLs .[1]: hosts-file.net, [2]: malwareurl.com, [3] siteadvisor.com, [4] virustotal.com, [5] ddanchev.blogspot.com, [6] malwaredomainlist.com.

Domain Name	IP	Type	Src	Date
lzwn.in	94.23.198.97	MAL	[1]	08-26
3b9.ru	213.251.176.169	MAL	[2]	08-30
antivirprotect.com	64.40.103.249	RAV	[3]	09-05
1speed.info	212.117.163.165	CWS	[2]	09-05
spy-destroyer.com	67.211.161.44	CWS	[4]	09-05
free-spybot.com	63.243.188.110	RAV	[2]	09-05
a3l.at	89.171.115.10	MAL	[2]	09-09
gidromash.cn	211.95.79.170	BOT	[2]	09-13
iantivirus-pro.com	188.40.52.180	KBF	[5]	09-19
ericwanhouse.cn	220.196.59.19	EXP	[6]	09-22
1165651291.com	212.117.165.126	RAV	[2]	10-06

We also benchmarked the reputation function on other two datasets containing a larger number of *known good* domain names. We experimented with bot the top 10,000 and top 100,000 Alexa domain names. The detection results for these experiments are as follows. When using the top 10,000 Alexa domains, we obtained a true positive rate of 93.6% and a false positive rate of 0.4% (again using 10-fold cross-validation and a detection threshold equal to 0.5). As we can see, these results are not very different from the ones we obtained using only the top 500 Alexa domains. However, when we extended our list of *known good* domains to include the top 100,000 Alexa domain names, we observed a significant decrease of the true positive rate and an increase in the false positives. Specifically, we obtained a TP% of 80.6% and a FP% of 0.6%. We believe this degradation in accuracy may be due to the fact that the top 100,000 Alexa domains include not only professionally run domains and network infrastructures, but also include less good domain names, such as file-sharing, porn-related websites, etc., most of which are not run in a professional way and have disputable reputation (e.g., A quick analysis of the top 100,000Alexa domains reported that about 5%of the domains appeared in the SURBL (www.surbl.org) blacklist, at certain point in time. A more rigorous evaluation of these results is left to future work.).

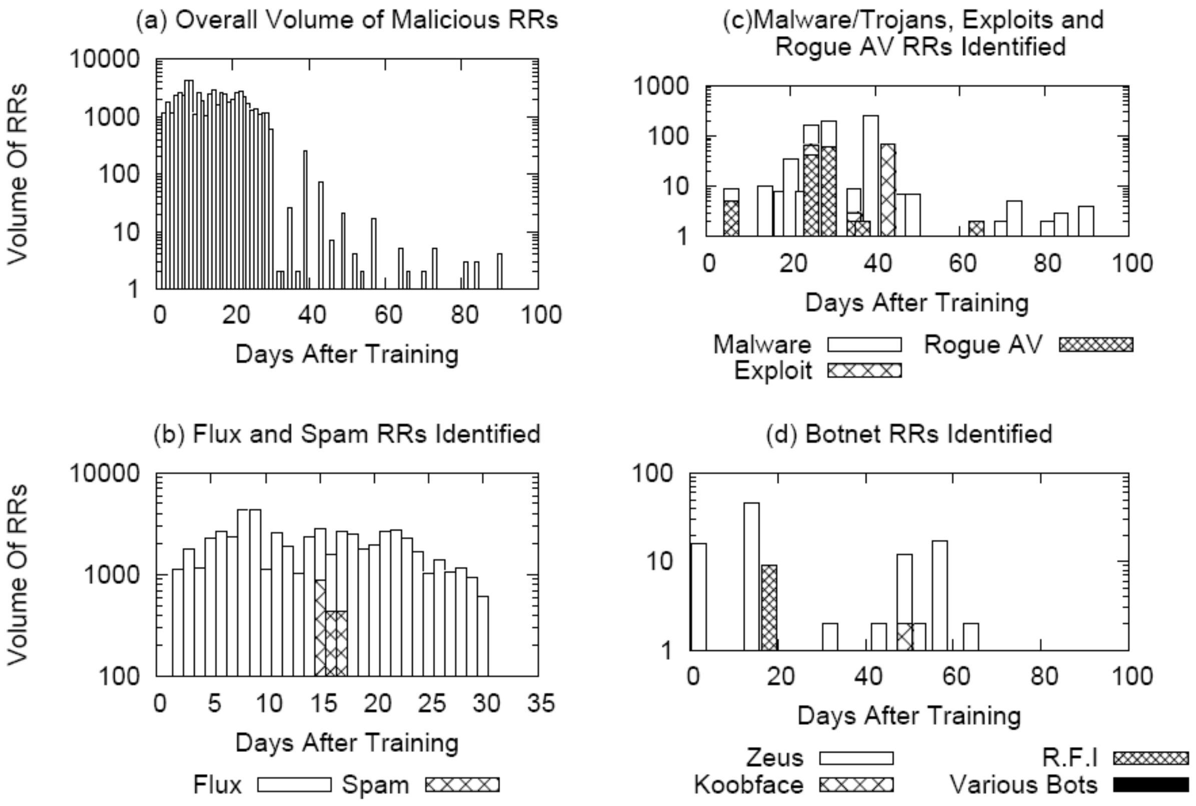


Figure 23: Dates in which various blacklists confirmed that the RRs were malicious after Notos assigned low reputation to them on the 1st of August.

We also wanted to evaluate how well Notos performs, compared to static blacklists. To this end, we performed a number of experiments as follows. Given an instance of Notos trained with data collected up to July 31, 2009, we fed Notos with 250,000 distinct RRs found in DNS traffic we collected on August 1, 2009. We then computed the reputation score for each of these RRs. First, we set the detection threshold to 0.5, and with this threshold we identified 54,790 RRs that had a low reputation (lower than the threshold).

These RRs where related to a total of 10,294 distinct domain names (notice that a domain name may map to more than one IP, and this explains the higher number of RRs). Of these 10,294 domains, 7,984 (77.6%) appeared in at least one of the public blacklists we used for comparison within 60 day after August 1, and were therefore confirmed to be malicious. Figure 23 (a) reports the number and date in which RRs classified as having low reputation by Notos appeared in the public blacklists. The remaining three plots (Figure 23 (b), (c) and (d)), report the same results organized according to the type of malicious domains. In particular, it is worth noting that Notos is able to detect never-before-seen domain names related to the Zeus botnet several days or even weeks before they appeared in any of the public blacklists.

For the remaining 22.4% of the 10,294 domains we considered, we were not able to draw a definitive conclusion. However, we believe many of those domains are involved in some kind of more or less malicious activities. We also noticed that 7,980 or the 7,984 *confirmed bad* domain names were assigned a reputation score lower or equal to 0.15, and that none of the other non-confirmed suspicious domains received a score lower than this threshold. In practice, this means that an operator who would like to use Notos as a stand-alone dynamic blacklisting system while limiting the false positives to a negligible (or even zero) amount may fine-tune the detection threshold and set it around 0.15.

4.2 Discussion

In this project, we have developed algorithms to detect botnets through the analysis of DNS traffic, local area network traffic, and malware samples. These algorithms were evaluated using real-world data, and have been implemented into working, deployed systems. We highlight the most significant results below.

For DNS monitoring, Notos, which is a system that provides a reputation score of a domain based on passive DNS traffic, represents a new and dynamic way to create and manage DNS reputation. Static DNS reputation systems have been around for several years, but increasingly, botmasters are able to use a number of techniques (e.g., generate many new domains daily) to render these static systems ineffective. With Notos, even a new domain is assigned a score based on where the domain server is hosted, the reputation of the provider, etc. Our experiments have shown that Notos is highly accurate: 98% true positive rate and below 1% false positive rate. And it can identify malicious domains weeks to months before they are actively used for malicious activities (and then appear in static reputation systems). This means that Notos can provide predictive capability that missing in static systems. As for future work, we will extend Notos so as to provide more refined classification and reputation score on type of malicious activities are associated with a domain.

In network-based anomaly detection, we developed a scalable botnet detection system using adaptive sampling and spatial-temporal correlation. This is the first system that aims at analyzing traffic in a very large network, such as an ISP, to detection botnets. The key to this system is the traffic sampling techniques that are designed to identify possible flows that are involved in botnets C&C. Our experimental results show that the proposed approach is scalable and can effectively detect bots with few false positives, which can be further reduced by fine-grained botnet detection systems such as BotSniffer and BotMiner.

In malware analysis, we developed Ether, a malware analysis system that uses hardware virtualization features to ensure that malware cannot detect the presence of the analyzer and evade analysis. The key insight is that with hardware virtualization, there is no emulation required and hence there is no observable discrepancy (between execution on an emulator vs. true hardware) that the malware can observe. Our experiment results show that while many malware programs are now evading popular analysis systems because they are based on emulation, they cannot evade Ether.

5.0 CONCLUSION

In this project, we have successfully developed a wide range of novel botnet detection, analysis, and mitigation techniques. These techniques can be deployed at the end-hosts, enterprise networks, ISPs, and network/DNS operators. The deliverables of this project include many publications in prestigious academic conferences, software and data delivered to the government, and commercial products and services. We also successfully completed several add-on tasks.

6.0 REFERENCES

[1] Anubis: Analyzing Unknown Binaries. http://anubis.seclab.tuwien.ac.at.

[2] Armadillo. http://www.siliconrealms.com.

[3] BitBlaze Binary Analysis Platform. http://bitblaze.cs.berkeley.edu.

[4] Themida. http://www.oreans.com/themida.php.

[5] Norman Sandbox Whitepaper. http://www.norman .com/documents/wp sandbox.pdf, 2003.

[6] A. Ramachandran, S. Seetharaman, and N. Feamster. Fast monitoring of traffic subpopulations. In *Proc. ACM IMC*, 2008.

[7] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster. Building a dynamic reputation system for dns. In *Proceedings of The 19th USENIX Security Symposium*, 2010.

[8] F. Bellard. QEMU, a Fast and Portable Dynamic Translator. In *ATEC*, pages 41–41, 2005.

[9] L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[10] I. S. Consortium. SIE@ISC : Security Information Exchange.

https://sie.isc.org/,2004.

[11] A. Dinaburg, P. Royal, M. Sharif, and W. Lee. Ether: Malware analysis via hardware virtualization extensions. http://ether.gtisc.gatech.edu/, 2009.

[12] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley-Interscience, 2nd edition, 2000.

[13] G. Gu, J. Zhang, and W. Lee. Botsniffer: Detecting botnet command and control channels in network traffic. In *Proc. NDSS*, 2008.

[14] G. Gu, R. Perdisci, J. Zhang, and W. Lee. Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. In *Proc. USENIX Security*, 2008.

[15] G. Gu, V. Yegneswaran, P. Porras, J. Stoll, and W. Lee. Active botnet probing to identify obscure command and control channels. In *Proceedings of The 25th Annual Computer Security Applications Conference (ACSAC 2009)*, 2009.

[16] T. Hothorn and B. Lausen. Double-bagging: Combining classifiers by bootstrap aggregation. *Pattern Recognition*, 36(6):1303–1309, 2003.

[17] Infosecurity. Storm deadnet reanimates as waledac botnet. http://infosecurityus/?p=6262, 2009.

[18] M. G. Kang, P. Poosankam, and H. Yin. Renovo: A Hidden Code Extractor for Packed Executables. In *WORM*, 2007.

[19] C. P. Lee. *FRAMEWORK FOR BOTNET EMULATION AND ANALYSIS*. PhD thesis, Georgia Institute of Technology, Atlanta, GA, Nov. 2008.

[20] R. Perdisci, I. Corona, D. Dagon, and W. Lee. Detecting malicious flux service networks through passive analysis of recursive dns traces. In *Proceedings of The 25th Annual Computer Security Applications Conference (ACSAC 2009)*, 2009.

[21] R. Perdisci, W. Lee, and N. Feamster. Behavioral clustering of http-based malware and signature generation using malicious network traces. In *Proceedings of The 7th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2010.'

[22] P. Royal, M. Halpin, D. Dagon, R. Edmonds, and W. Lee. PolyUnpack: Automating the Hidden-Code Extraction of Unpack-Executing Malware. In *ACSAC*, pages 289–300, 2006.

[23] M. Sharif, A. Lanzi, J. Giffin, and W. Lee. Automatic reverse engineering of malware emulators. In *Proceedings of The 2009 IEEE Symposium on Security and Privacy*, 2009.

6.1 List of Performers

The PI and co-PIs of this project are:

- Dr. Wenke Lee, PI; Professor of Computer Science, Georgia Tech.
- Dr. Jon Giffin, co-PI; Assistant Professor of Computer Science, Georgia Tech
- Dr. Nick Feamster, co-PI; Assistant Professor of Computer Science, Georgia Tech
- David Dagon, co-PI; Research Scientist, Georgia Tech
- Gunter Ollman, co-PI; VP of Research, Damballa Inc.
- Jody Westby, co-PI; CEO, Global Cyber Risk, LLC
- Rick Wesson, co-PI; CEO, Support Intelligence
- Paul Vixie, co-PI; Internet Systems Consortium

6.2 List of Publications/Reports

The list of publications in referred conferences and journals:

1. Junjie Zhang, Roberto Perdisci, Wenke Lee, Unum Sarfraz, and Xiapu Luo. Detecting Stealthy P2P Botnets Using Statistical Traffic Fingerprints. In *Proceedings of the 41st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, Hong Kong, China, June 2011 (to appear).
2. Junjie Zhang, Xiapu Luo, Roberto Perdisci, Guofei Gu, Wenke Lee, and Nick Feamster. Boosting the Scalability of Botnet Detection Using Adaptive Traffic Sampling. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, Hong Kong, China, March 2011 (to appear).
3. Manos Antonakakis, David Dagon, Xiapu Luo, Roberto Perdisci, and Wenke Lee. A Centralized Monitoring Infrastructure for Improving DNS Security. In *Proceedings of The 13th International Symposium on Recent Advances in Intrusion Detection (RAID)*, Ottawa, Ontario, Canada, September 2010.
4. Long Lu, Vinod Yegneswaran, Phil Porras, and Wenke Lee. BLADE: An Attack-Agnostic Approach for Preventing Drive-By Malware Infections. In *Proceedings of The 17th ACM Conference on Computer and Communications Security (CCS)*, Chicago, IL, October 2010.
5. Manos Antonakakis, Roberto Perdisci, David Dagon,Wenke Lee, and Nick Feamster. Building a Dynamic Reputation System for DNS. In *Proceedings of The 19th USENIX Security Symposium*,Washington, DC, August 2010.
6. Kapil Singh, Samrit Sangal, Nehil Jain, Patrick Traynor, and Wenke Lee. Evaluating Bluetooth as a Medium for Botnet Command and Control. In *Proceedings of The 7th Conference on Detection of Intrusions and Malware Vulnerability Assessment (DIMVA)*, Bonn, Germany, July 2010.
7. Kapil Singh, Alexander Moshchuk, Helen J. Wang, and Wenke Lee. On the Incoherencies in Web Browser Access Control Policies. In *Proceedings of The 2010 IEEE Symposium on Security and Privacy*, Oakland, CA, May 2010.

8. Roberto Perdisci, Wenke Lee, and Nick Feamster. Behavioral Clustering of HTTP-based Malware and Signature Generation using Malicious Network Traces. In *Proceedings of The 7th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA, April 2010.
9. Roberto Perdisci, Igino Corona, David Dagon, and Wenke Lee. Detecting Malicious Flux Service Networks through Passive Analysis of Recursive DNS Traces. In *Proceedings of The 25th Annual Computer Security Applications Conference (ACSAC 2009)*, Honolulu, HI, December 2009.
10. Guofei Gu, Vinod Yegneswaran, Phillip Porras, Jennifer Stoll, and Wenke Lee. Active Botnet Probing to Identify Obscure Command and Control Channels. In *Proceedings of The 25th Annual Computer Security Applications Conference (ACSAC 2009)*, Honolulu, HI, December 2009.
11. Monirul Sharif, Wenke Lee, Weidong Cui, and Andrea Lanzi. Secure In-VM Monitoring Using Hardware Virtualization. In *Proceedings of The 16th ACM Conference on Computer Communications Security (CCS 2009)*, Chicago, IL, November, 2009.
12. Martim Carbone, Weidong Cui, Long Lu, Wenke Lee, Marcus Peinado, and Xuxian Jiang. Mapping Kernel Objects to Enable Systematic Integrity Checking. In *Proceedings of The 16th ACM Conference on Computer and Communications Security (CCS 2009)*, Chicago, IL, November, 2009.
13. Kapil Singh, Sumeer Bhola, and Wenke Lee. xBook: Redesigning Privacy Control in Social Networking Platforms. In *Proceedings of The 18th USENIX Security Symposium*, Montreal, Canada, August, 2009.
14. Roberto Perdisci, Manos Antonakakis, Xiapu Luo, and Wenke Lee. WSEC DNS: Protecting Recursive DNS Resolvers from Poisoning Attacks. In *Proceedings of The 39th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2009)*, Lisbon, Portugal, June 2009.
15. Monirul Sharif, Andrea Lanzi, Jon Giffin, and Wenke Lee. Automatic Reverse Engineering of Malware Emulators. In *Proceedings of The 2009 IEEE Symposium on Security and Privacy*, Oakland, CA, May 2009. (Best Student Paper Award)
16. Andrea Lanzi, Monirul Sharif, and Wenke Lee. K-Tracer: A System for Extracting Kernel Malware Behavior. In *Proceedings of The 16th Annual Network and Distributed System Security Symposium (NDSS 2009)*, San Diego, CA, February 2009.
17. David Dagon, Manos Antonakakis, Kevin Day, Xiapu Luo, Christopher P. Lee, and Wenke Lee. Recursive DNS Architectures and Vulnerability Implications. In *Proceedings of The 16th Annual Network and Distributed System Security Symposium (NDSS 2009)*, San Diego, CA, February 2009.
18. Roberto Perdisci, Davide Ariu, Prahlad Fogla, Giorgio Giacinto, and Wenke Lee. McPAD: A Multiple Classifier System for Accurate Payload-Based Anomaly Detection. *Computer Networks*, Vol. 53, 2009.
19. Roberto Perdisci, Andrea Lanzi, andWenke Lee. McBoost: Boosting Scalability in Malware Collection and Analysis Using Statistical Classification of Executables. In *Proceedings of The 24th Annual Computer Security Applications Conference (ACSAC 2008)*, Anaheim, CA, December 2008.

20. Roberto Perdisci, Andrea Lanzi, and Wenke Lee. Classification of Packed Executables for Accurate Computer Virus Detection. *Pattern Recognition Letters* Vol. 29, No. 14 (October 2008).

21. Artem Dinaburg, Paul Royal, Monirul Sharif, and Wenke Lee. Ether: Malware Analysis via Hardware Virtualization Extensions. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS 2008)*, Alexandria, VA, October 2008.

22. David Dagon, Manos Antonakakis, Paul Vixie, Tatuya Jinmei, and Wenke Lee. Increased DNS Forgery Resistance Through 0x20-Bit Encoding. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS 2008)*, Alexandria, VA, October 2008.

23. Monirul Sharif, Vinod Yegneswaran, Hassen Saidi, Phillip Porras, and Wenke Lee. Eureka: A Framework for Enabling Static Malware Analysis. In *Proceedings of the 13th European Symposium on Research in Computer Security (ESORICS)*, Malaga, Spain, October 2008.

24. Guofei Gu, Roberto Perdisci, Junjie Zhang, and Wenke Lee. BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection. In *Proceedings of The 17th USENIX Security Symposium (Security'08)*, San Jose, CA, July 2008.

25. Kapil Singh, Abhinav Srivastava, Jon Giffin, andWenke Lee. Evaluating Email's Feasibility or Botnet Command and Control. In *Proceedings of the 38th Annual IEEE/IFIP International*

Conference on Dependable Systems and Networks (DSN 2008), Anchorage, Alaska, June 2008.

26. Bryan D. Payne, Martim Carbone, Monirul Sharif, and Wenke Lee. Lares: An Architecture for Secure Active Monitoring Using Virtualization. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, Oakland, CA, May 2008.

27. Guofei Gu, Alvaro A. Cardenas, and Wenke Lee. Principled Reasoning and Practical Applications of Alert Fusion in Intrusion Detection Systems. In *Proceedings of the ACM Symposium on InformAction, Computer and Communications Security (ASIACCS'08)*, Tokyo,Japan, March 2008.

28. ConfickerWorking Group: Lessons Learned. http://www.confickerworkinggroup.org/wiki/uploads/Conficker Working Group Lessons Learned 17 June 2010 final.pdf. January 2011.

LIST OF ACRONYMS

AS	Autonomous System
BGP	Border Gateway Protocol
BL	Black List
C&C	Command and Control
CDN	Content Distribution Network
DNS	Domain Name System
DPI	Deep Packet Inspection
DREN	Defense Research and Engineering Network
FP	False Positive
HTTP	HyperText Transfer Protocol
IDS	Intrusion Detection System
IP	Internet Protocol
IRC	Internet Relay Chat
ISC	Internet Systems Consortium
ISP	Internet Service Provider
LAD	Logit-Boost Strategy
P2P	Peer-to-Peer
PCAP	Packet Capture
pDNS	passive DNS
RDNS	Recursive DNS
RHIP	Related Historic IP
RHDN	Related Historic Domain
RR	Resource Record
SIE	Security Information Exchange
SMTP	Simple Mail Transport Protocol
SR	Sampling Rate
TCP	Transmission Control Protocol
TLD	Top-Level Domain
TP	True Positive
UDP	User Datagram Protocol