# Detecting Botnets Using a Collaborative Situational-aware IDPS

M. Lisa Mathews[1], Anupam Joshi[1] and Tim Finin[1]
[1]*Department of Computer Science & Electrical Engineering, University of Maryland, Baltimore County, Baltimore, MD, USA*
{*math1, joshi, finin*}*@umbc.edu*

Keywords: intrusion detection, situational-aware, botnet detection

Abstract: Botnet attacks turn susceptible victim computers into bots that perform various malicious activities while under the control of a botmaster. Some examples of the damage they cause include denial of service, click fraud, spamware, and phishing. These attacks can vary in the type of architecture and communication protocol used, which might be modified during the botnet lifespan. Intrusion detection and prevention systems are one way to safeguard the cyber-physical systems we use, but they have difficulty detecting new or modified attacks, including botnets. Only known attacks whose signatures have been identified and stored in some form can be discovered by most of these systems. Also, traditional IDPSs are point-based solutions incapable of utilizing information from multiple data sources and have difficulty discovering new or more complex attacks. To address these issues, we are developing a semantic approach to intrusion detection that uses a variety of sensors collaboratively. Leveraging information from these heterogeneous sources leads to a more robust, situational-aware IDPS that is better equipped to detect complicated attacks such as botnets.

## 1 INTRODUCTION

Botnets have the potential to cause substantial damage to the various hosts and networks they target or affect. They are a pervasive type of attack in which bots perform various malicious activities while under the control of a botmaster. A few of their well known consequences are denial of service, click fraud, spamware, and phishing (Thuraisingham et al., 2008; Feily et al., 2009). To make matters even more interesting, they can be difficult to detect. Bailey et al. summarized this well when they said botnets "can propagate like worms, hide from detection like many viruses, attack like many stand-alone tools, and have an integrated command and control system" (Bailey et al., 2009). The destructive effects of botnets have lead the Obama Administration to propose giving the Department of Justice the ability to issue injunctions for attacks where at least 100 computers have been hacked (Prince, 2015).

Current state-of-the-art intrusion detection and prevention systems (IDPSs) are recognized as having limitations that lessen their effectiveness in dealing with attacks, especially those of a more sophisticated nature. Conventional IDPSs are typically point-based solutions incapable of utilizing information from other sources, even if this data was available for them to use. They are limited in the attacks they can identify by the signatures stored in their database. They cannot detect zero day type attacks, attacks that use low-and-slow vectors, or other complicated attacks such as botnets. Many times an attack is only revealed by post facto forensics after some damage has already been done.

We tackle the limitations mentioned above by creating a collaborative situational-aware IDPS that uses a variety of information sources to detect attacks. The collaborative aspect comes from the fact that multiple sensors are employed, both traditional and non-traditional. Traditional sensors encompass the various hardware and software available to scan and/or monitor hosts and networks, such as Snort or Norton Antivirus or less conventional tools such as Process Explorer. Other types of IDPSs might gather information from different sensors, but some only include sensors from a single vendor, and most simply dashboard the information that is acquired. For the IDPS we are developing, incorporating specific products or vendors is not important, and the information is stored in a knowledge base for system integration. The goal is to have some combination of hardware and software that provides a clear picture of what is happening on the hosts and network. This would a network administrator to use the tools they already have available to them. By nontraditional sensors, we mean online sources in the forms of vulnerability databases, forums, blogs, etc., that contain structured and unstructured textual descriptions of potential vulnerabilities or exploits. These descriptions are often for unpatched vulnerabilities, such as those from the National Vulnerability Database (NVD), that attackers could potentially use to target affected systems.

Information gathered from these different channels is stored as facts into a knowledge base using a W3C standards based ontology that has been developed in house. In addition, high level rules, the kind that are typically used by an analyst as they look at data for evidence of attacks, are also encoded in the same ontology. The IDPS that emanates from the various sources working in conjunction is well armed to stop complex attacks such as bonets. This paper describes our overall approach and a preliminary version of this system that detects botnets. We validate the system by using a variety of cues to detect two botnets, namely Skynet and W32/Vobfus, aka W32/Changeup.

The rest of this paper is organized as follows. Section II details the background of the IDPS along with a brief botnet overview. Section III goes over the related work. Section IV describes the system setup and validation, and Section V contains the results. Section VI provides the concluding remarks.

## 2 Background

### 2.1 IDPS

Intrusion detction systems (IDSs) are a popular method for defending cyber-physical systems against various threats. Basically, they passively observe host and/or network data and show an alert when anomalous behavior is detected. An IDS will not take any action against a potential threat; instead, it stores these observations in a log file and shows an alert. Intrusion Prevention Systems (IPSs) are more active and will attempt to thwart any intruders from harming a system. Combining these two systems forms a more robust defense mechanism known as an IDPS.

In previous work, we presented a novel approach to intrusion detection and prevention by making an IDPS situationally-aware (More et al., 2012) and collaborative (Mathews et al., 2012). The essence of this IDPS is to take the data gathered from traditional sensors and add to that the data gained from nontraditional sensors, like online forums or Twitter feeds, analyze this data, and create additional facts and information expressed using an ontology we created. In combination with rules, also expressed using terms

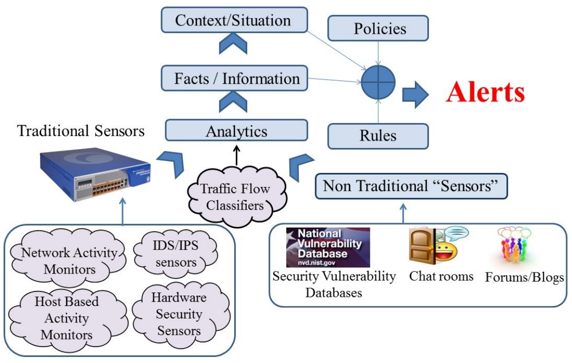

Context/Situation
Policies
Facts / Information
Alerts
Traditional Sensors
Analytics
Rules
Traffic Flow Classifiers
Non Traditional "Sensors"
Network Activity Monitors
IDS/IPS sensors
Host Based Activity Monitors
Hardware Security Sensors
National Vulnerability Database
Security Vulnerability Databases
Chat rooms
Forums/Blogs

Figure 1: Our architecture supports situation awareness by integrating and analyzing data extracted from multiple sources, both traditional and nontraditional. The results are represented in a knowledge-based system and reasoned over to detect potential attacks. Taken from (Mathews et al., 2012).

from this ontology, that are normally used by analysts to detect attacks post facto, the system can infer if a situation is indicative of an attack. The use of heterogeneous and diverse data sources can provide a wider picture of the system activities and information related to potential attacks that are being discussed, and thus, better help detect zero day attacks for which signatures do not exist.

The designed framework can be split into two main sections. The first involves the gathering of data from different channels such as online web sources, logs from existing IDS/IPS systems, host-based activity monitors, network activity monitors, and hardware security sensors. The second section employs an ontology to model both the intrusions in terms of their defining characteristics and the various hosts and networks of a system in terms of their properties and behavior. This information is stored in a knowledge base as a set of rules, and a reasoner is used to extract additional information from this knowledge base. The system architecture designed for this collaborative situation-aware IDS (More et al., 2012) is depicted in Figure 1. The components of the second section will be described in the next few paragraphs.

The ontology used in this work is one focused on cybersecurity concepts and is an extension of a previous ontology developed in house (Undercoffer et al., 2003). The classes present in this current ontology allow an attack to be represented in terms of its means and consequences and a system to be represented by its host and network properties. The means of an attack is the method in which it is executed, such as a buffer overflow. The consequence would be the end result, which could be denial of service (DoS) for the previous means. Examples of system properties modeled by the ontology include operating systems, processes running, and IP addresses. As stated in our previous work, the knowledge base is built up by encoding the information as Web Ontology Language (OWL (Bechhofer et al., 2004)) and Resource Description Framework (RDF (Manola et al., 2014)) assertions. We serialize these using Notation-3 (N3 (Berners-Lee and Connolly, 2014)) triples of the form (subject predicate object) that asserts that the relation $p$ holds between $s$ and $o$ (Mathews et al., 2012).

The reasoning logic component takes the output from the various data channels, knowledge base assertions, rules, and information representation properties of the ontology to infer the presence of an attack. This reasoner allows for additional facts to be inferred, either from rules, or from the structure of the ontology. Take for example a triple that describes a network activity such as "ComputerA sendsPacketTo ComputerB". As humans, we understand that this can also be expressed as "ComputerB receivedPacketFrom ComputerA", but with the use of the reasoner and an owl:inverseOf statement relating sendsPacketTo and receivedPacketFrom, the system knows this as well.

### 2.2 Botnets

Most of the top-rated IDPSs implemented to defend computer networks will not be able to stand up to newly published attacks, especially the more complicated ones such as botnets. A botnet consists of a botmaster, or botmasters in some cases, that gains control of its victims through various methods, turning these compromised computers into its bots, short for robots (Feily et al., 2009; Bailey et al., 2009). One property of botnets that distinguishes them from other types of malware is their use of command and control (C&C) servers. The bots will continue to receive instructions from its botmaster while the botnet lives on, usually through a backdoor. Different botnets have different architectures and communication protocols, and these can in fact change during the course of the attack. In the past, these attacks have utilized a more centralized structure for communication amongst the members of the botnet, while more recent designs have implemented a decentralized architecture using peer-to-peer (P2P) communication.

Skynet, discovered in December 2012, was a complex botnet with different mechanisms in place to add stealth and fly under the radar while performing malicious activities including execution of a distributed DOS, illegal Bitcoin generation, and procurement of online login credentials (Constantin, 2012). As seen with other botnets, Skynet is a variant of another botnet, in this case Zeus. The supposed inventors of this attack had possibly created nearly one million dollars in Bitcoins using a modified Zeus banking trojan and infested more than 12,000 computers by the time they were caught in December of 2013 (Kumar, 2013; Constantin, 2012). Its use of Tor to hide its Internet Relay Chat (IRC) C&C communication makes identifying the servers and bots more difficult. Tor is a specialized software that provides traffic encryption and anonymity by dispersing its users' traffic over several relays, thus hiding the location of its users and making traffic analysis a challenge (Guarnieri, 2012). The IDPS described in this paper would be able to detect this attack using the combination of traditional and nontraditional sensors as described without having components specifically designed for botnet detection.

## 3 Related Work

In prior work, we described our efforts in creating a situational-aware IDPS framework and also presented the details for a network traffic based classifier that shows promise for detecting malicious traffic (Mathews et al., 2012; More et al., 2012). Sharma et al. (Sharma et al., 2013) designed and validated a framework that contains several modules for monitoring various parameters that indicate the occurrence of data exfiltration, examples of which include network usage and DLLs called. Monitoring these parameters would allow a system to be profiled in terms of its normal behavior for all layers, i.e., from hardware up to application.

There have been many papers published that focus on botnet detection and even a few survey papers that cover the different detection techniques. The paper by Feily et al. (Feily et al., 2009) provides an overview of botnet detection methods while covering the basics of the botnets including the terminology, characteristics, and infection life cycle. The authors of this survey state that botnets can be classified according to their command and control architecture, which they also emphasize is the defining property of these attacks that also provides anonymity for the botmaster. The four detection methods they summarize and then go on to compare and contrast are signature based, anomaly based, DNS based, and mining based. The evaluation criteria included determining whether the detection method could identify unknown bots, whether it was protocol and structure independent, if it could deal with encrypted command and control channels, the ability to detect these botnets in real-time, and the accuracy of each method. According to the authors, the most promising botnet detection techniques are those based on a data mining approach and another that utilizes a DNS based method. Data mining detection techniques try to identify the C&C traffic that is required for these attacks. DNS based methods seek to identify DNS traffic that occurs during a botnet's life cycle. We tried a pure data mining approach to identify various botnets, but the results showed that different botnets exhibit different patterns and implementing additional sensors rather than just a network traffic analyzer would stand a better chance at detection.

While some detection techniques focus on one specific type of botnet, others try to analyze different kinds in order to find similar characteristics. Algorithms that are created as a result of studying botnets of a specific type can sometimes be restrictive in the sense that they can only be applied to those specific botnets, as mentioned in the botnet survey paper (Bailey et al., 2009). They described the changes botnets have gone through while still having the end goal of turning their targets into zombies, or bots.

Other researchers have studied cross-analysis botnet results. For example, Shin et al. studied one botnet that uses auto-self propagating techniques and two that are non-auto-self-propagating (Shin et al., 2011). Their experiments examined properties like the geographical distribution of infected networks and the remote accessibility of networks to detect botnets of both similar and different propagation type. Due to the recognition of these attacks as a serious concern and their complex attack vectors, many research efforts result in elaborate algorithms or system components that are designed to detect and possibly deter botnet attacks. They usually do not work well on other types of attacks. Examples of these types of systems include the works of Gu et al. (BotMiner, BotSniffer, BotHunter, and BotProbe) (Gu et al., 2009). Many of these algorithms also require the use of deep packet inspection (DPI), which looks at the payload information of a packet and can be considered intrusive and raises privacy concerns. Our approach demonstrates how a collaborative, situational aware IDPS could detect different kinds of attacks including, but not limited to, botnets.

Allowing an IDPS to understand the context or situation in which an attack can occur can go a log way in preventing different kinds of attacks from occurring, not just the ones whose signature is in a database. While situational awareness might be applied to more physical domains, the potential of its implementation in the cyber realm is starting to be realized. The authors of idsNETS (Mancuso et al., 2012) realized this and are developing a system to test how users with varying degrees of cyber security knowledge understand this concept by seeing how they defend a system/network against multiple attacks, some more harmful than others. Their work focuses on studying studying situational awareness instead of its implementation.

Squicciarini et al. have also realized the usefulness of a situational aware intrusion detection system (Squicciarini et al., 2014). What we call an attack, they refer to as a case or incident. Their system, ReasONets, incorporates information from host and network sensors, which results in information similar to what is gathered in previous work done by us (Mathews et al., 2012), (Sharma et al., 2013). These authors also use Snort, stress the importance of a reasoner and knowledge base, and incorporate DNS blacklists as examples of sources of additional information, similar to our previous work (More et al., 2012), (Mathews et al., 2012).

## 3.1 Data Mining Approach to Botnet Detection

After our initial research lead us to believe that a data mining approach would be best, we decided to try studying different botnet pcap files to identify various patterns that exist. A pcap file contains network packet data that is captured while the network is running. The botnets studied were W32/SDbot (McRee, 2006), Kelihos/Hlux botnet (Mila, 2013), and Skynet/Tor (Mila, 2012).

In order to apply data mining techniques to the botnet pcap files, we used two free and open-source products. Wireshark is a popular network protocol analyzer that can capture traffic on a network and open previously obtained pcap files (Wireshark, 2015). RapidMiner is a data mining software that provides a user-friendly GUI environment for applying data mining and machine learning techniques and visualizing the output (RapidMiner, 2015). The fields/attributes chosen to be exported were the source IP address, destination IP address, protocol, packet length, destination port, source port, and ExpertInfo.

The ExpertInfo field that is produced by Wireshark is similar to other intrusion detection systems alerts. There are five values that an individual instance can have: NULL, Chat, Note, Warn, Error. NULLs occur for normal traffic while Chats occur with traffic that might be considered normal but there is something unusual, but not alarming, regarding the workflow. Notes occur for packets that are suspicious but not strongly so. The Warn value is seen for activity perceived as out of the ordinary and should be taken as a warning. An Error value is the worst rating indicating a serious problem.

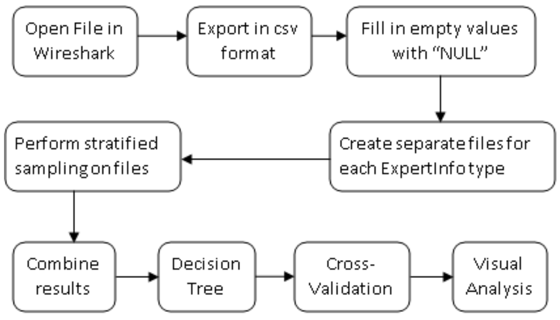

Open File in Wireshark
Export in csv format
Fill in empty values with "NULL"
Create separate files for each ExpertInfo type
Perform stratified sampling on files
Combine results
Decision Tree
Cross-Validation
Visual Analysis
Figure 2: Block diagram of methodology

Figure 2 represents the block diagram showing how each botnet pcap file is preprocessed and analyzed. Looking at the confusion matrix for each file, Chats seem to be the most difficult to classify properly. The NULL class has the best results with very few misclassified instances appearing since they were hand-selected specifically so that only normal traffic would be included. The Note, Error, and Warn classes were mixed; half the time the instances were classified correctly for the most part, and the other half they were not.

When looking at the resulting decision tree and observing the paths from the root to the leaf nodes along with the intermediate attributes, there is no discernible pattern for the attribute values. The Protocol and Source and Destination Port attributes were in all but one of the decision trees. The Destination IP Address attribute appeared only in one tree, and the Source IP Address did not appear at all. The main pattern observed was for the PacketLength attribute value, which varied for the different branches of each tree. However, a split was often seen on the value 57. When looking at the files for each botnet and ordering by PacketLength, it was seen that in many cases a value of 54 bytes would have TCP ZeroWindow listed as part of the Info column. This occurs when the sender is told that the destination cannot accept more information for the moment. While this could be used as part of a rule/policy to detect botnets, this pattern alone would not catch them since it can also be observed in nonbotnet traffic. No obvious pattern or trend was perceived regarding the depth or number of nodes in the tree and the accuracy. This failure to detect patterns at the syntax level in attributes that would enable classification provides support to our overall approach of using semantically rich representations and reasoning to detect attacks.

## 4 Current System Setup and Validation

### 4.1 Current Setup

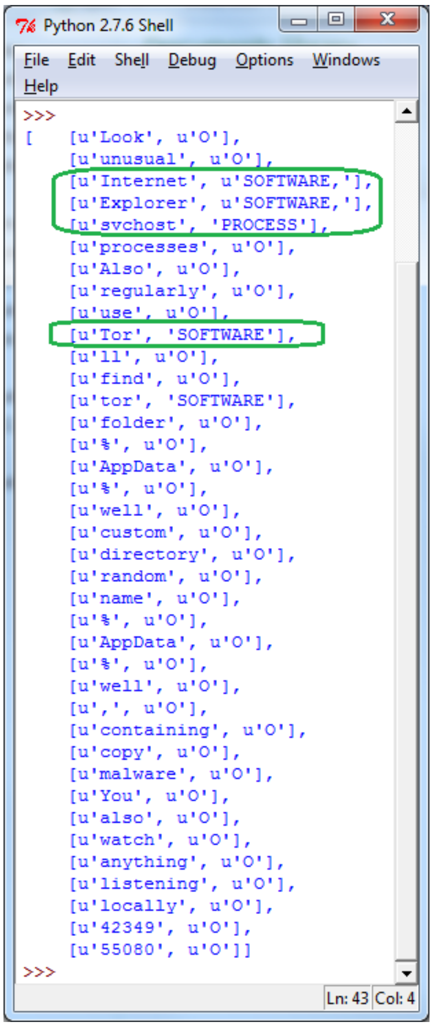
We tested the prototype system described above against a sophisticated botnet attack using the various components we have created or utilized for the IDPS. Actual binaries were desired to recreate the attack since, along with the shortcomings listed in the previous section, it is difficult to ascertain the timing of events in a pcap that someone else has created. In other words, the particular times or more accurate knowledge of the packets where important events take place is not always easily determined. Also, it is often the case that the exact system configuration, such as the operating system and what vulnerable applications were installed, is not mentioned. Instead, we found the executables for the Skynet botnet, also knows as Trojan.Tbot, from the contagio site (Mila, 2012).

The text processing module described in (Mathews et al., 2012) and the host parameter monitoring system described in (Sharma et al., 2013) were utilized for system validation. Even though the Skynet botnet was discovered in December of 2012, the attack was still infecting victim computers when the alleged creators were arrested one year later (Kumar, 2013). Virtual machines (VMs) were set up to run the botnet in a sandboxed environment with the modules of the IDPS described in this paper in place. Additionally, Symantec and Microsoft write-ups of the W32/Vobfus attack were found and sent through the text processing module. A summary of the results will be provided below.

### 4.2 Text Processing Module Results

As described earlier, textual sources can provide discussions on new attacks where formal signatures are not yet available from AV providers or vendors. Consider the text below, which is a response taken from an entry on the Reddit website to the question of how to detect Skynet.

> Look for unusual Internet Explorer and svchost processes. Also if you don't regularly use Tor you'll find a "tor" folder in %AppData% as well as a custom directory with a random name in %AppData% as well, containing a copy of the malware. You can also watch for anything listening locally on 42349 and 55080 (botherder, 2012).



```
>>>
[    [u'Look', u'O'],
     [u'unusual', u'O'],
     [u'Internet', u'SOFTWARE,'],
     [u'Explorer', u'SOFTWARE,'],
     [u'svchost', 'PROCESS'],
     [u'processes', u'O'],
     [u'Also', u'O'],
     [u'regularly', u'O'],
     [u'use', u'O'],
     [u'Tor', 'SOFTWARE'],
     [u'll', u'O'],
     [u'find', u'O'],
     [u'tor', 'SOFTWARE'],
     [u'folder', u'O'],
     [u'%', u'O'],
     [u'AppData', u'O'],
     [u'%', u'O'],
     [u'well', u'O'],
     [u'custom', u'O'],
     [u'directory', u'O'],
     [u'random', u'O'],
     [u'name', u'O'],
     [u'%', u'O'],
     [u'AppData', u'O'],
     [u'%', u'O'],
     [u'well', u'O'],
     [u',', u'O'],
     [u'containing', u'O'],
     [u'copy', u'O'],
     [u'malware', u'O'],
     [u'You', u'O'],
     [u'also', u'O'],
     [u'watch', u'O'],
     [u'anything', u'O'],
     [u'listening', u'O'],
     [u'locally', u'O'],
     [u'42349', u'O'],
     [u'55080', u'O']]
>>>
```

Figure 3: Output from Reddit post sent through Security NER

Our text processing module can analyze this description and extract information that can aid in asserting the fact that host process modification indicates the presence of a botnet. Passing the response text through our NER (named entity recognizer) results in the identification of the relevant processes and software as shown in Figure 3. In our ontology, 'showsInfectionSigns' and 'opensPort' are properties of the class 'Process'; 'DistributedCode' is a subclass of 'MaliciousCodeExecution', which is a subclass of the 'Means' class; and 'DistributedCode' has the object property of 'resultsIn' with the class 'BotnetAttack', which is a subclass of 'Consequence'. A rule that accounts for this threat, like the one in Figure 4, could say that if a host-based sensor detects a process flagged as showing infections signs (as detected by the process monitoring module), and there are open ports 42349 and 55080 (as shown by some

```
[ IDPS:webText_Reddit001  IDPS:hasProcess              ?Process
  IDPS:webText_Reddit001  IDPS:hasPort                 ?Port
  IDPS:webText_Reddit001  IDPS:hasTerms                ?OtherTechnicalTerms
  IDPS:scannerLog_2       IDPS:hasProcess              ?Process
  IDPS:hostSensor_1       IDPS:hasProcess              ?Process
  IDPS:hostSensor_1       IDPS:showsInfectionSigns     "True"
  IDPS:scannerLog_2       IDPS:opensPort               ?Port
  IDPS:hostSensor_1       IDPS:anomalousDataOutFlow    "True"]
=>
[ IDPS:System_001         isUnderAttack                "botnet"
  IDPS:attack             hasMeans                     MaliciousCodeExecution
  IDPS:attack             hasConsequence               BotnetAttack
  IDPS:attack             hasConsequence               DenialofService
  IDPS:attack             hasConsequence               LossofIntegrity ]
```

Figure 4: This rule, serialized as N3, asserts RDF triples describing a potential attack based on the presence of triples representing the state of the system and recent events.

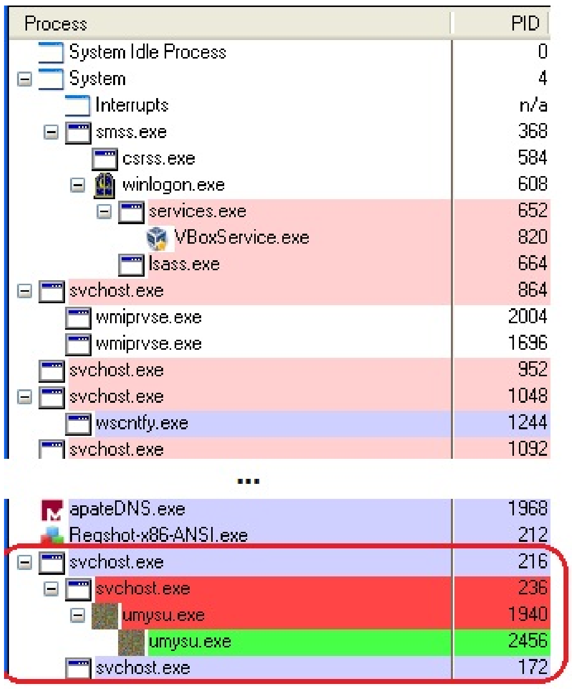
network monitoring tool), and that there is an abnormal amount of data flow observed in behavior such as a large spike in network activity or connection attempts to/from previously unseen IP addresses (as detected by Wireshark), a botnet attack is likely occurring.

Several parameters need to indicate abnormal activity in order for the IDPS to throw suspicion on this botnet. It is important that all of the indicators are present to reduce the possibility of false positives. One instance of a low level alert might not be of interest to anyone, but if all the indicators are present, this warrants at least additional investigation by a network administrator.

In order to identify the malicious behavior of a botnet, several tools and applications need to be in place. These are used by the host and network parameter monitors mentioned earlier. Detecting Skynet would involve the use of host monitors to detect process behavior and registry modifications and network monitors to detect suspicious port openings and abnormal traffic behavior.

Figure 5 shows a screenshot of ProcessExplorer, which has been incorporated into the host parameter monitor, as it detects the disguised svchost.exe process that is created after one of the Skynet files is executed. During later executions of the botnet inside a new VM, the disguised process was sometimes shown as IEXPLORER.exe. Symantec had posted a writeup of Trojan.Tbot in their Security Response section where the describe the technical details of this trojan including modifications seen on an infected system (Spasojevi, 2012). Their analysis of the botnet listed the creation of a rogue svchost.exe process, but interestingly did not mention anything about the Internet Explorer process. False negatives would most likely occur if their signature to detect Skynet needed the discovery of the disguised process and didn't include IEXPLORER.exe.

Figure 6 shows a screenshot of Regshot which compares two snapshots of a host's registry to detect

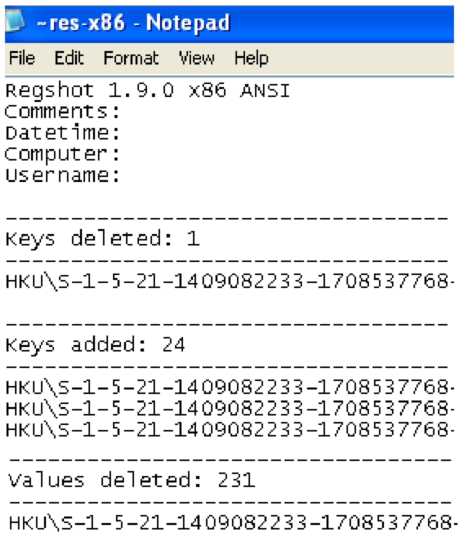


```
Process                              PID
  System Idle Process                  0
⊟ System                               4
    Interrupts                       n/a
  ⊟ smss.exe                         368
      csrss.exe                      584
    ⊟ winlogon.exe                   608
      ⊟ services.exe                 652
          VBoxService.exe            820
        lsass.exe                    664
⊟ svchost.exe                        864
    wmiprvse.exe                    2004
    wmiprvse.exe                    1696
  svchost.exe                        952
⊟ svchost.exe                       1048
    wscntfy.exe                     1244
  svchost.exe                       1092
...
  apateDNS.exe                      1968
  Regshot-x86-ANSI.exe               212
⊟ svchost.exe                        216
  ⊟ svchost.exe                      236
    ⊟ umysu.exe                     1940
        umysu.exe                   2456
    svchost.exe                      172
```

Figure 5: Process Explorer showing disguised svchost.exe

any modifications including keys that were deleted and added, values that have been deleted and added, and values that have been modified. There were 382 modifications noted while the VM was allowed to run, and this was without a Tor client or Bitcoin account set up. While this might not be a big deal considering registry values can change with legitimate or normal activity, it is still something that could be of interest when other suspicious host or network behavior is observed. Netcat was used to listen on ports 42349 and 55080 and showed a connection attempt on port 42349.

Posts from Microsoft's Malware Protection Center and Symantec provided information on the W32/Vobfus, aka W32/Changeup, malware including the different attack vectors and operating systems affected (Young et al., 2009; Diaz and Estavillol, 2010). Victim computers are infected through removable media and network drives where an autorun.inf file, an AutoRun configuration file, is set to execute a copy of the worm once the drive is accessed. The worm will copy itself and the corresponding autorun.inf file to any removable and network drives found on the newly compromised machine. It will then attempt to download additional malware from remote locations through a few specific ports to different executables under the %UserProfiles% folder with random file names. Registry keys are modified so that the malware runs each time the computer is restarted.



```
Regshot 1.9.0 x86 ANSI
Comments:
Datetime:
Computer:
Username:

----------------------------------
Keys deleted: 1
----------------------------------
HKU\S-1-5-21-1409082233-1708537768-

----------------------------------
Keys added: 24
----------------------------------
HKU\S-1-5-21-1409082233-1708537768-
HKU\S-1-5-21-1409082233-1708537768-
HKU\S-1-5-21-1409082233-1708537768-

----------------------------------
Values deleted: 231
----------------------------------
HKU\S-1-5-21-1409082233-1708537768-
```

Figure 6: RegShot showing indication of registry modification

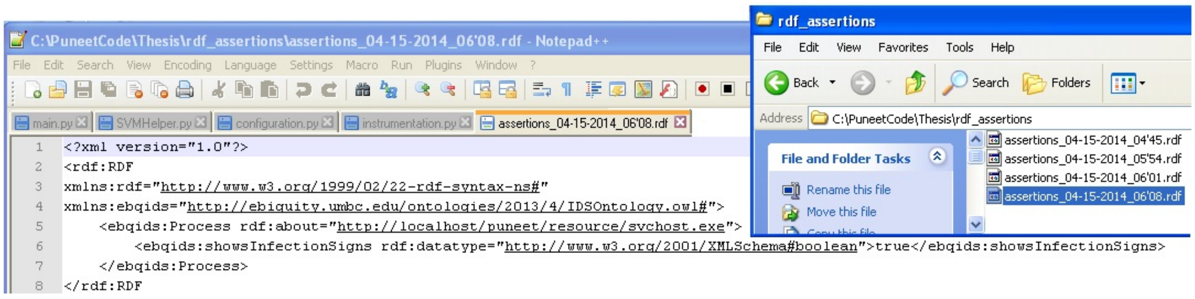
The worm may also attempt to modify registry keys so that its location is hidden and Windows updates are disabled. The output from sending the Microsoft technical information text through the NER is shown in Figure 7.

## 4.3 Host and Network Modules Results

The IDPS outlined in this paper has the different host and network sensors along with the modules to analyze them in place to flag an alert when suspicious activity is discovered. This should help alleviate the burden of network administrators who might monitor these different tools individually. In Figure 8, the window on the right shows the RDF assertions generated during the course of the attack. The window on the left shows the highlighted assertion regarding the disguise of the svchost.exe process.

VirusTotal provides an online service where files and URLs will be sent through several antivirus and antispamware engines and blacklists and provides a report on any suspicious content (VirusTotal, 2015). The results of the old VirusTotal scans uploaded onto the contagio site show that many of the Skynet files had a low detection rate when passed through several IDSs/IPSs a few weeks after the attack was discovered in December 2012 (Mila, 2012). This indicates that relying on one sensor to detect all attacks is not ideal. When running the bontnet files against the site as of April 2015, some of the malware detection engines still do not flag the files as malicious.

Having forensic style rules that can detect various complicated threats and are not limited to one partic-


C:\PuneetCode\Thesis\rdf_assertions\assertions_04-15-2014_06'08.rdf - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
main.py
SVMHelper.py
configuration.py
instrumentation.py
assertions_04-15-2014_06'08.rdf
rdf_assertions
File Edit View Favorites Tools Help
Back
Search
Folders
Address C:\PuneetCode\Thesis\rdf_assertions
File and Folder Tasks
Rename this file
Move this file
assertions_04-15-2014_04'45.rdf
assertions_04-15-2014_05'54.rdf
assertions_04-15-2014_06'01.rdf
assertions_04-15-2014_06'08.rdf

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:ebqids="http://ebiquity.umbc.edu/ontologies/2013/4/IDSOntology.owl#">
    <ebqids:Process rdf:about="http://localhost/puneet/resource/svchost.exe">
        <ebqids:showsInfectionSigns rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">true</ebqids:showsInfectionSigns>
    </ebqids:Process>
</rdf:RDF
```

Figure 8: RDF assertions generated

```
>>>
[   [u'The', u'O'],
    [u'worm', u'O'],
    [u'writes', u'O'],
    [u'Autorun', u'MEANS,'],
    [u'configuration', u'MEANS,'],
    [u'file', u'MEANS,'],
    [u'named', u'O'],
    [u'autorun.inf', 'PROCESS'],
...
    [u'Win32Vobfus', u'MEANS,'],
    [u'changes', u'MEANS,'],
    [u'following', u'MEANS,'],
    [u'registry', u'MEANS,'],
    [u'entries', u'MEANS,'],
    [u'prevent', u'MEANS,'],
    [u'changing', u'MEANS,'],
    [u'hidden', u'MEANS,'],
    [u'files', u'MEANS,'],
    [u'folders', u'MEANS,'],
    [u'displayed', u'O'],
...
    [u'Downloads', u'MEANS,'],
    [u'runs', u'MEANS,'],
    [u'malware', u'MEANS,'],

    [u'The', u'O'],
    [u'remote', u'CONSEQUENCES,'],
    [u'hosts', 'PROCESS'],
    [u'address', u'O'],
    [u'hardcoded', u'O'],
    [u'variants', u'O'],
    [u'binary', u'O'],
    [u',', u'O'],
    [u'varies', u'O'],
    [u'malware', u'CONSEQUENCES,'],
    [u'author', u'CONSEQUENCES,'],
    [u'releases', u'CONSEQUENCES,'],
    [u'new', u'O'],
    [u'binaries', u'CONSEQUENCES,'],
```

Figure 7: Output from Microsoft technical information on W32/Vobfus sent through Security NER

ular threat would give an IDPS an advantage in discovering attacks whose specific signatures have not yet been created. Take for example the rule created after extracting text from the Reddit post regarding Skynet. Since the attack behavior for W32/Vobfus, as described above, is similar to that observed in Skynet, the same rule constructed earlier should be able to detect this additional attack as well.

## 5 Conclusion

In this paper, we substantiated the need for a collaborative, situational-aware IDPS to detect sophisticated attacks. The IDPS outlined is capable of gathering data from traditional as well as nontraditional sources, extracting the relevant information, and storing that information in a knowledge base as facts. When used in conjunction with domain expert rules put in place, malicious host and network activity will have a better chance of being detected.

Initial experiments were conducted using several pcaps, but a few significant drawbacks made us switch to finding actual binaries. The complex attack of choice was the Skynet botnet, aka Trojan.Tbot. Old VirusTotal scans of the botnet files reveal that upon discovery of the attack in December of 2012, many of the popular IDS/IPS engines failed at identifying them as a threat. The text processing module is capable of extracting the necessary information to create a useful rule to detect botnets, not just specific to Skynet. Several applications, such as ProcessExplorer and registry monitors, that have been incorporated into the IDPS described in this paper would detect the host and network changes that are checked by the rule created to identify the attack.

As part of our future work, we will study the number of false positives and false negatives generated by our IDPS compared to other systems and modify our system as needed. Any bottlenecks that occur while gathering information from the different sensors will need to be dealt with. Adding a confidence metric or assigning weights to the different rules in the knowledge base is also a possible future contribution.

## REFERENCES

Bailey, M., Cooke, E., Jahanian, F., Xu, Y., and Karir, M. (2009). A survey of botnet technology and defenses. In *Conference For Homeland Security, 2009. CATCH'09. Cybersecurity Applications & Technology*, pages 299–304. IEEE.

Bechhofer, S., Van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., and Stein, L. A. (2004). OWL Web Ontology Language Overview. http://www.w3.org/TR/owl-features.

Berners-Lee, T. and Connolly, D. (2014). Notation3 (N3): A readable RDF syntax. http://www.w3.org/TeamSubmission/n3/.

botherder (2012). Skynet, a Tor-powered botnet straight from Reddit. http://www.reddit.com/r/netsec/comments/14etfq/skynet_a_torpowered_botnet_straight_from_reddit/.

Constantin, L. (2012). Tor network used to command Skynet botnet. http://www.pcworld.idg.com.au/article/444088/tor_network_used_command_skynet_botnet/.

Diaz, Jr, E. and Estavillol, P. (2010). Win32/Vobfus. http://www.microsoft.com/security/portal/threat/encyclopedia/Entry.aspx?Name=Win32%2fVobfus.

Feily, M., Shahrestani, A., and Ramadass, S. (2009). A survey of botnet and botnet detection. In *Emerging Security Information, Systems and Technologies, 2009. SECURWARE'09. Third International Conference on*, pages 268–273. IEEE.

Gu, G., Yegneswaran, V., Porras, P., Stoll, J., and Lee, W. (2009). Active botnet probing to identify obscure command and control channels. In *Computer Security Applications Conference, 2009. ACSAC'09. Annual*, pages 241–253. IEEE.

Guarnieri, C. (2012). Skynet, a Tor-powered botnet straight from Reddit. http://community.rapid7.com/community/infosec/blog/2012/12/06/skynet-a-tor-powered-botnet-straight-from-reddit.

Kumar, M. (2013). Alleged Skynet Botnet creator arrested in Germany. http://thehackernews.com/2013/12/alleged-skynet-botnet-creator-arrested.html/.

Mancuso, V. F., Minotra, D., Giacobe, N., McNeese, M., and Tyworth, M. (2012). idsnets: An experimental platform to study situation awareness for intrusion detection analysts. In *Cognitive Methods in Situation Awareness and Decision Support (CogSIMA), 2012 IEEE International Multi-Disciplinary Conference on*, pages 73–79. IEEE.

Manola, F., Miller, E., and McBride, B. (2014). Rdf 1.1 Primer. http://www.w3.org/TR/rdf11-primer/.

Mathews, M. L., Halvorsen, P., Joshi, A., and Finin, T. (2012). A collaborative approach to situational awareness for cybersecurity. In *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2012 8th International Conference on*, pages 216–222. IEEE.

McRee, R. (2006). http://holisticinfosec.org/toolsmith/files/nov2k6/toolsmith.pcap.

Mila (2012). Dec. 2012 Skynet Tor botnet / Trojan.Tbot samples. http://contagiodump.blogspot.com/2012/12/dec-2012-skynet-tor-botnet-trojantbot.html.

Mila (2013). Trojan Nap aka Kelihos/Hlux status update by DeepEnd Research and samples. http://contagiodump.blogspot.com/2013/02/trojan-nap-aka-kelihoshlux-status.html.

More, S., Matthews, M., Joshi, A., and Finin, T. (2012). A knowledge-based approach to intrusion detection modeling. In *Security and Privacy Workshops (SPW), 2012 IEEE Symposium on*, pages 75–81. IEEE.

Prince, B. (2015). Obama administration proposes giving courts more power to issue botnet injunctions. http://www.securityweek.com/obama-administration-proposes-giving-courts-more-power-issue-botnet-injunctions.

RapidMiner (2015). http://rapidminer.com/.

Sharma, P., Joshi, A., and Finin, T. (2013). Detecting data exfiltration by integrating information across layers. In *Information Reuse and Integration (IRI), 2013 IEEE 14th International Conference on*, pages 309–316. IEEE.

Shin, S., Lin, R., and Gu, G. (2011). Cross-analysis of botnet victims: New insights and implications. In *Recent Advances in Intrusion Detection*, pages 242–261. Springer.

Spasojevi, B. (2012). Trojan.Tbot. http://www.symantec.com/security_response/writeup.jsp?docid=2012-120716-2955-99.

Squicciarini, A. C., Petracca, G., Horne, W. G., and Nath, A. (2014). Situational awareness through reasoning on network incidents. In *Proceedings of the 4th ACM conference on Data and application security and privacy*, pages 111–122. ACM.

Thuraisingham, B., Hamlen, K. W., Khan, L., and Masud, M. M. (2008). Data mining for security applications. In *Embedded and Ubiquitous Computing, IEEE/IFIP International Conference on*, volume 2, pages 585–589. IEEE.

Undercoffer, J., Joshi, A., and Pinkston, J. (2003). Modeling computer attacks: An ontology for intrusion detection. In *Recent Advances in Intrusion Detection*, pages 113–135. Springer.

VirusTotal (2015). https://www.virustotal.com/en//.

Wireshark (2015). http://www.wireshark.org/.

Young, E., Honda, H., and Bell, H. (2009). W32.Changeup. http://www.symantec.com/security_response/writeup.jsp?docid=2009-081806-2906-99.