Attack of the Zombies: Discovering and Mitigating Bots and Botnets

By

Paul Albert

Submitted to
the Faculty of the Information Technology Program
in Partial Fulfillment of the Requirements
for
the Degree of Bachelor of Science
in Information Technology

© Copyright 2010 Paul Albert

The contents of this document are under copyright of the author. It may not be reproduced and distributed in whole or in part without the written permission of the author.

_____________________________________________________  __________________
Paul M. Albert         Date

_____________________________________________________  __________________
John Nyland, Faculty Advisor      Date

_____________________________________________________  __________________
Hazem Said, Ph.D., Department Head     Date
# Table of Contents

1 Statement of Problem  
2 Solution  
3 User Profiles  
4 Design Protocols  
  4.1 Hardware  
  4.2 Software  
5 Proof of Concept  
  5.1 Starting Snort IDS  
  5.2 Accessing BASE  
  5.3 Attacker Machine Attacking Target  
  5.4 Barnyard2 Gathering Information from Snort Log File  
  5.5 BASE Displaying Alerts  
  5.6 Updating IDS Rules  
  5.7 Script for Installing IDS  
6 Timeline  
7 Budget  
8 Deliverables  
9 Testing  
10 Risk Management Plan  
11 Conclusion
List of Illustrations

Figure 1 - Diagram of a botnet 3
Figure 2 - Diagram of solution 5
Figure 3 - Flowchart of IDS operating procedures 7
Figure 4 - Flowchart of functioning IDS system 8
Figure 5 - Snort IDS started 11
Figure 6 - Barnyard2 started 11
Figure 7 - BASE running 12
Figure 8 - Attacker machine performing a port scan on the Windows XP 1 machine 12
Figure 9 - Attacker machine performing a port scan on the Windows XP 2 machine 13
Figure 10 - Barnyard2 catching Snort alerts 13
Figure 11 - BASE displaying alert statistics at a high-level view 14
Figure 12 - BASE displaying alert statistics in detail 14
Figure 13 - BASE displaying alert statistics by classification 15
Figure 14 - BASE displaying detailed packet information on an alert 15
Figure 15 - BASE displaying a graph on number of alerts vs. the specific source TCP port 16
Figure 16 - BASE displaying a graph on number of alerts vs. a specified time period 16
Figure 17 - Oinkmaster script starting 17
Figure 18 - Oinkmaster script ending 17
Figure 19 - IDS install script starting 18
Figure 20 - IDS install script ending 18
Figure 21 - Timeline of the project 19
Table 1 - Software used in solution

Table 2 - Budget of the project

Table 3 - Risk management plan for the project
Abstract

Bots, botnets, and other malicious cyber attacks have increasingly been on the rise over the past decade. These types of attacks can lead to identity theft, spam, spyware, adware, distributed denial of services (DDoS) attacks, and many other attacks. Detection of bots and other cyber attacks can be difficult to detect due to the many variants of malicious software, and the way the programming code is written. There is a need to have an easy method for detecting and mitigating bots and other malicious internet traffic. This project creates an easy-to-implement, accurate, and cheap solution for discovering, mitigating, and reporting bots and botnet activity on a network. This project takes advantage of an open source intrusion detection system (IDS) called Snort, along with other open source software to meet its requirements. Scripting in Perl is also utilized to allow for easy installation, updating, and other purposes. An open source reporting tool is used to provide a dashboard for analysis of alerts. Depending on the size of the network this project is placed on, it can be used to target only botnet activity, or other types of cyber attacks. Proper implementation of this project allows for an efficient way of detecting, reporting, alerting, and mitigating bots, botnet activity, and other cyber attacks.
1 Statement of Problem

The presence of botnets on the internet is causing an enormous problem on a growing number of computer networks around the world. In their first quarter of 2009 threats report, McAfee Avert Labs reported a nearly 50% increase in computers operating as zombies (8). Botnets (short for robot network) are dangerous because of their command and control style of operation. The concept of a botnet operation is rather simple. As shown in figure 1, the attacker (known as a bot herder) has a server setup typically at another location. The bot herder then infects one or more machines with malicious software that then typically scans networks and looks for other machines to infect. The bot herder can infect as many machines as they wish, and the machines that are infected are often referred to as zombies. Once machines are infected, the bot herder can send commands to their command and control server. Once a command is sent, the server proceeds to send commands to all of the machines it has infected. A botnet can consist of as few as two machines to millions of machines.

There are many types of attacks that are commonly deployed by botnets. Below is a list of the most common uses (6):

- Spreading Itself
- Distributed Denial of Service Attacks (DDoS)
- Identity Theft
- Spam
- Spyware
- Adware
- Click Fraud
- Hosting files
Detection of bots or botnet activity on a network can be very difficult due to the many variants of malicious software. The bot herder can make the source code change as the bot spreads. Virus scanners can only detect bots for which their programmers have defined a signature. If a new bot has been created with different source code, virus scanners cannot detect the bot. With this type of polymorphic coding, bots will continue to be an increasing problem.

In 2009, a group of researchers from the University of California took over the Torpig which had hundreds of thousands of zombies under its control. The researchers observed the data being reported to the command and control server for a period of ten days. One particular piece of data reported was information obtained from financial institutions. Over a ten day
period, the Torpig bot obtained 8310 different pieces of financial data (11). This research shows the dangerous impact of botnets on computer networks throughout the world. There needs to be an easy method of detecting and mitigating botnets. This project creates an easy-to-implement, accurate, and cheap solution for discovering and reporting botnets on a network.

2 Solution

The solution for this problem is to implement an intrusion detection system (IDS) on a network that reports and alerts when suspected bot activity is detected. Snort, an open source IDS/IPS originally developed in 1998, is used for the solution (10). Snort can perform real time network traffic analysis and logging, while also having the ability to be an intrusion prevention system (IPS) (10). If someone discovers unique traffic that he or she wishes to detect in the future, custom rules can be created and imported into Snort. A Snort rule is a specific line of code that Snort interprets when analyzing network traffic. If Snort analyzes network traffic that matches the rule, an alert is triggered. Using the IPS features in Snort is essential for mitigating bots and botnet activity on the network.

This implementation can be scaled down to include only IDS rules that relate to bot activity for larger sized networks and can feature all the rules if on a smaller network. Accuracy, reliability, and the need for alerting on only bot activity is the reason for this stripped-down implementation. Using Snort IDS with its included bot rules and other rules created in the open source community allows for constant reliability and accurate results.

MySQL is used to store the IDS data, and Basic Analysis and Security Engine (BASE) to report on the data. BASE is a web-based application that analyzes and queries the alerts that come from Snort IDS (2). Scripts were made to help automate the setup process, along with
automatically importing the open source community rules for the IDS. This project can be used on a standard desktop computer or a server. Utilizing these tools with customizations for a network allows for a way to detect, report, and alert on bots and bot activity on a given network. Figure 2 illustrates using Snort IDS/IPS on a router or gateway. On larger networks Snort should be placed in a mirrored port on a switch.

![Figure 2 - Diagram of solution](image-url)
3 User Profiles

Depending on the size of the network, there may be one or more types of users. A system administrator is needed for the ownership of the process and reporting to managers if needed. The system administrator can also perform the maintenance that needs to be completed on the IDS system. Maintenance consists of system upgrades, configurations, and repair work. Analyzing BASE and the IDS can be completed by an analyst within the IT department. Analysis of both Snort and BASE can be completed by analyzing the logs through BASE, and other system logs that may exist. Also, a person(s) needs to receive and respond to alerts. Responding to alerts may consist of analyzing the alert, and determining what action to take as a result of the specific network activity. There are six needs for this project:

- Maintenance of IDS and BASE
- Analysis of IDS and BASE
- Receiving and responding to alerts
- Knowledge to implement solution
- Knowledge to troubleshoot IDS and BASE
- Knowledge to script in Perl

4 Design Protocols

Snort operates in the background when the operating system is booted. BASE reports all results from the Snort log file via a piece of software called Barnyard2. The analyst(s) responsible for analyzing and reporting on the IDS and BASE will log into BASE through a web browser. Once logged in, the user analyst can navigate through multiple sections to view alerts
or events that have occurred. BASE reports in detail on each intrusion attempt or malicious traffic. Snort can be configured to send an alert to the administrator when an intrusion occurs. Perl scripts are used to assist with the initial install and configuration of Snort, Barnyard2, and Base. Having these scripts allows for easy installation and setup, thus relieving the amount of knowledge that an IT employee would normally have to have to set up this type of IDS system. This setup also requires a couple of files to be changed when starting Snort and Barnyard, so there is a script for changing these files when the system starts. Figure 3 displays a flowchart for the operating procedures of the project.

![Flowchart of IDS operating procedures](image)

**Figure 3 - Flowchart of IDS operating procedures**

The project is based on three main pieces of software plus the scripting piece. When the operating system is started, Snort and Barnyard2 is started via a script. Apache web server is
started when the operating system starts, and BASE is then available through the web browser. When Snort detects bad traffic it logs that information to a log file. Barnyard2 reads that log file and translates it for BASE to read. BASE is configured to take that information and display it on a web interface. BASE can also graph information in a variety of ways. Figure 4 depicts how the IDS system functions.

Figure 4 - Flowchart of functioning IDS system
4.1 Hardware

The Snort intrusion detection system does not require a lot of processing power and resources. One of the goals in this project is to be able to run this IDS/IPS solution on almost any machine. With the use of VirtualBox, the only hardware needed is a single computer or server. A gigabit Ethernet controller is suggested on larger sized networks as it will be needed to handle more traffic. If this IDS/IPS solution is used on a switch, it is necessary to plug the machine into a mirrored port so it can view all the packets.

4.2 Software

One of the goals of this project is to create an accurate and cheap solution. To achieve this goal, open source software is being utilized. Most of the Linux based software has various other dependencies that have to be installed along with the software. The Windows XP machines are patched to match a real world scenario. Snort can be scaled down to allow for specifically targeting bots and botnet activity on the network. For this to occur, most of the rules are taken out of the rules directory within the Snort directory. Snort can also be used in inline mode to act as an intrusion prevention system. Table 1 shows a table of the software that is used:
Table 1 - Software used in solution

<table>
<thead>
<tr>
<th>Software Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>VirtualBox 3.1.2</td>
</tr>
<tr>
<td>Windows XP SP3</td>
</tr>
<tr>
<td>Ubuntu 9.10</td>
</tr>
<tr>
<td>Snort 2.8.4.1</td>
</tr>
<tr>
<td>Barnyard2 2.1.8</td>
</tr>
<tr>
<td>Basic Analysis Security Engine (BASE)</td>
</tr>
<tr>
<td>Oinkmaster 2.0</td>
</tr>
<tr>
<td>MySQL</td>
</tr>
<tr>
<td>Perl</td>
</tr>
</tbody>
</table>

5 Proof of Concept

Using VirtualBox to demonstrate the project proved to be extremely helpful. Three machines were used to demonstrate the project effectiveness. An Ubuntu machine acting as the IDS, Windows XP machine acting as the victim, and a BackTrack 4 machine acting as an attacker were all put on a network and assigned IP addresses. The screenshots in Figures 8 and 9 depict the attacker machine performing a port scan on the victim machine, and then show Snort IDS alerting on the port scan. Figures 10 and 11 show BASE displaying the alerts from Snort IDS.
5.1 Starting Snort IDS

Figure 5 below shows Snort IDS starting, and Figure 6 shows Barnyard2 starting. A script starts both of these programs.
5.2 Accessing BASE

Figure 7 shows BASE functioning.

![Figure 7 - BASE running](image)

5.3 Attacker Machine Attacking Target

Figure 8 shows the attackers machine (BackTrack 4 – IP: 192.168.1.9) running a port scan on the Windows XP 1 machine (192.168.1.7).

![Figure 8 - Attacker machine performing a port scan on the Windows XP 1 machine](image)
Figure 9 shows the attackers machine (BackTrack 4 – IP: 192.168.1.9) running a port scan on the Windows XP 2 machine (192.168.1.10)

5.4 Barnyard2 Gathering Information from Snort Log File

Figure 10 shows Barnyard2 catching the Snort alerts. You can see the rules that were triggered when nmap ran a port scan on the victim machine.
5.5 BASE Displaying Alerts

Figure 11 shows BASE displaying the alerts in a high-level view.

Figure 11 - BASE displaying alert statistics at a high-level view

Figure 12 shows BASE displaying details of the alerts that were triggered by the port scan.

Figure 12 - BASE displaying alert statistics in detail
Figure 13 shows BASE displaying the alerts triggered by the different scans and attacks by classification.

Figure 14 shows BASE displaying detailed packet information on a specific alert.
Figure 15 shows BASE displaying a graph based on the number of alerts vs. the specific source TCP port.

![Figure 15 - BASE displaying a graph on number of alerts vs. the specific source TCP port](image1)

Figure 16 shows BASE displaying a graph based on the number of alerts vs. a specified time period.

![Figure 16 - BASE displaying a graph on number of alerts vs. a specified time period](image2)
5.6 Updating IDS Rules

Figure 17 shows the script for Oinkmaster starting and updating the IDS rules.

![Figure 17 - Oinkmaster script starting](image)

Figure 17 - Oinkmaster script starting

Figure 18 shows the script for Oinkmaster executed successfully.

![Figure 18 - Oinkmaster script ending](image)

Figure 18 - Oinkmaster script ending
5.7 Script for Installing IDS

Figure 19 shows the script starting for the IDS install, along with other software.

![Figure 19 - IDS install script starting](image1)

Figure 19 - IDS install script starting

Figure 20 shows the script for the IDS install executed successfully.

![Figure 20 - IDS install script ending](image2)

Figure 20 - IDS install script ending
6 Timeline

Research on this project began at the beginning of fall quarter and lasted roughly three months. Additional research was performed throughout the second quarter to fulfill any additions, problems, or concerns with the solution. The technical part of the project was started on in the second quarter, and finished in the third quarter. Testing is ongoing to ensure everything is working efficiently and to make any adjustments that may be necessary. The major milestone throughout this whole project is completing everything up to and including the scripting. Completion of the scripting allows sufficient time for final testing and other assignments that must be completed for this project. Figure 21 is a timeline showing tasks that need to be completed, along with the proposed time frame for each task.

![Figure 21 - Timeline of the project](image-url)
7 Budget

Because open source software is utilized, the budget is relatively low compared to commercial products that are available. With the MSDN Academic Alliance, Windows XP was free. A subscription to Snort is also an option that allows for more frequent updates that are maintained by Sourcefire VRT. Table 2 shows the cost for each piece of hardware and software being used in this project.

Table 2 - Budget of the project

<table>
<thead>
<tr>
<th>Item</th>
<th>Estimated Cost</th>
<th>Actual Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>VirtualBox 3.1.2</td>
<td>Free</td>
<td>Free</td>
</tr>
<tr>
<td>Desktop Computer</td>
<td>Free</td>
<td>$1,000.00</td>
</tr>
<tr>
<td>Windows XP Professional</td>
<td>Free</td>
<td>$299.99</td>
</tr>
<tr>
<td>Ubuntu 9.10</td>
<td>Free</td>
<td>Free</td>
</tr>
<tr>
<td>Snort IDS</td>
<td>Free</td>
<td>Free</td>
</tr>
<tr>
<td>Snort Subscription</td>
<td>$29.99</td>
<td>$499.00</td>
</tr>
<tr>
<td>MySQL</td>
<td>Free</td>
<td>Free</td>
</tr>
<tr>
<td>BASE</td>
<td>Free</td>
<td>Free</td>
</tr>
<tr>
<td>Total</td>
<td>$29.99</td>
<td>$1,798.99</td>
</tr>
</tbody>
</table>

8 Deliverables

Below is a list of deliverables to be completed by the project deadline.

- VirtualBox Install
  - Installation of VirtualBox on the project computer
  - Image of multiple Windows Operating Systems
  - Image of Linux Operating System

- Intrusion Detection System Setup
  - Installation of Snort IDS along with MySQL
  - Configuration of IDS and required software

- Basic Analysis and Security Engine (BASE) Setup
  - Installation of BASE
  - Configuration of BASE
- Mitigation
  - Recommendations based on reports that are generated

- Scripting
  - Writing of scripts to assist with installation and maintenance of IDS

- Testing
  - Test the functionality of the IDS, BASE, scripts, and alerting functions

9 Testing

VirtualBox is being used to host the virtual environment, and hosts one Ubuntu machine, two Windows XP machines, and one BackTrack machine. I first placed malicious software and bots on the Windows XP machines. When all necessary machines had a bot successfully installed, BASE showed the malicious activity, thus showing that the setup was successful and the IDS detected malicious network traffic. The IDS/IPS solution was integrated in the virtual environment, and was tested to ensure the scripts worked to install the IDS and retrieved updates. Efficient detection, reporting, alerting, and prevention are necessary to ensure a proper implementation of the IDS/IPS solution.

10 Risk Management Plan

Table 3 below explains the risk management plan for this project.

<table>
<thead>
<tr>
<th>Risk Description</th>
<th>Risk Level</th>
<th>Risk Mitigation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problems with configuration of any software</td>
<td>Medium</td>
<td>Designate more time to researching and fixing any issues</td>
</tr>
<tr>
<td>Scripts not working correctly</td>
<td>Medium</td>
<td>More time may have to be dedicated to fixing issues, and research of the issue.</td>
</tr>
<tr>
<td>Data loss or corruption</td>
<td>High</td>
<td>Backup data on a regular basis</td>
</tr>
<tr>
<td>Not keeping up with timeline</td>
<td>Medium</td>
<td>Analyze and update timeline weekly</td>
</tr>
</tbody>
</table>
11 Conclusion

Botnets are an increasing threat to almost every computer network in the world. A single bot herder can control tens of thousands of machines and launch a wide variety of dangerous attacks. Bot herders have the ability to create or change the source code of existing known bots to avoid virus protection, allowing for many bots to go undetected on networks and computers. There is a need for an accurate, cheap and easy-to-implement solution to detect, mitigate, and report on bot and botnet activity. Proper implementation of Snort IDS/IPS, MySQL, BASE, and scripting allows for an efficient way of detecting, reporting, alerting, and mitigating bots and botnet activity.
References


