

# A Multi-Layered Approach to Botnet Detection

Robert F. Erbacher      Adele Cutler      Pranab Banerjee      Jim Marshall  
Dept. of Computer Science    Dept. of Math and Stats      SDL      SDL  
Utah State University      Utah State University    USU Research Foundation    USU Research Foundation  
Robert.Erbacher@usu.edu    Adele.Cutler@usu.edu    Pranab.Banerjee@sdl.usu.edu    Jim.Marshall@sdl.usu.edu

**Abstract** – The goal of this research was to design a multi-layered architecture for the detection of a wide range of existing and new botnets. By not relying on a single technique but rather building in the ability to support multiple techniques, the goal is to be able to detect a wider array of bots and botnets than is possible with a single technique. The open architecture and API will allow any techniques designed by other researchers to be integrated. The goal is to use signature type techniques to detect well-known bots and botnets and data mining techniques to detect new classes and variants; i.e. anomaly or misuse detection.

**Keywords** – Botnet Detection, Software Architecture, Signature-Based Detection, Data Mining.

## 1 Introduction

Bots and botnets [12][19] are an existing and growing threat to the global cyber community. These malicious codes are used for a variety of nefarious purposes and have essentially become a major technology as well as a financial threat to the cyber security of government, industry and academia. The difficulty in detecting botnets derives from the rapidity with which botnets change and adapt, often specifically to avoid detection. This has resulted in a wide range of different types and sub-types of bots. A study on the current extent of botnets was reported recently by Rajab et al. [14].

Given the extent of the threats of botnets and the difficulty of detection we have designed a multi-layered architecture for the detection of botnets. Unlike virus scanners that require regular updates, the proposed solution has the ability to detect new threats as they emerge. The software architecture uses an extensible approach allowing

for the easy addition of new detection modules as botnet threats evolve and techniques are refined. The software architecture creates a solution that will mitigate the bot ‘arms race’ that is occurring today. Table 1 examines fundamental requirements for such a bot detection tool and how our proposed solution fulfills these requirements.

## 2 Background

There literally exists an army of bot writers and bot attackers in the world today. Antivirus companies such as Norton [18] and McAfee [17] are incorporating bot detection into their antivirus tools. However, since bots can and are being dynamically updated by the bot controllers, these detection strategies are failing for the most part. For instance, as soon as an anti-virus company updates their signature files to identify a variation of a bot, the bot controller updates the bot to change the signature. This has resulted in a wide array of both distinct classes of bots and bot variations. Consequently, bot defenses need to be organized and developed within a flexible, structured architecture that is aimed at involving antibot software writers in developing tools that will be used to update and defeat any new threat by integrating into a coordinated environment.

The designed architecture has the following attributes:

- Hierarchical and secure
- Multilayered
- Combination of standard existing tools (firewalls and antiviral s/w) of “old direct” methods (signature recognition) with “new indirect” data mining-methods
- Open to being constantly enhanced with new antibot modules. The multilayer architecture is purposely designed such that the kernel is closed and secured and

Need	Proposed Solution
Automatically scan networks and nodes detecting bots and botnets	Automated data mining such as Random Forest and neural networks on network data
All operations, code and methods will operate within legal and ethical boundaries	No ‘hack-back’ or other offensive mitigation approaches will be taken or recommended, no automated mitigation as a response to bot detection will be implemented
Mitigation approaches will be recommended once a bot or botnet has been detected	Upon bot/botnet detection, a mitigation strategy will be recommended to the user based on existing bot information and data mining (Random Forest) data classification
Minimize impact on network, system performance and or operations	Code will be streamlined with minimal impact on resources, system administrator will have ultimate control of which processes run and when/where they execute, this provides fine grain control over effectiveness vs. performance tradeoffs.
Primary operator of the code is the System Administrator	Tool will be designed for System Administrator use.

at the same time allows the modification, patching, and enhancing of the system via a predefined open API.

Other integrated detection environments exist such as OSSIM [15] and Prelude [16]. However, they are designed as general IDS environments and are not focused on the unique and adaptable issues of botnets. Current botnet and other malware detection schemes rely on a signature approach that needs to be updated, distributed, and installed on each node of the network once a new threat emerges. Thus, they are not effective against unknown and against the adaptive nature of modern botnets. This approach is only valid if the threat is contained and not exposed to the network or node in question.

Through the application of the multi-layered approach we are not relying on any single technique. This will help ensure the detection on not only known bots but also new variations and new classes of bots [6].

### 3 System Overview

Our solution to botnet detection consists of a multi-layered approach implemented within a client-server software architecture, allowing for extensibility and expandability. The core of the system is an automated process using statistical data mining techniques, such as Random Forests, applied to network data. These processes execute on a server with access to network traffic and/or on an individual node within the system. Layers of bot specific detection techniques are incorporated. The architecture provides the system administrator the flexibility to launch antbot actions on all or a select number of nodes in the system.

Since individual bots may not be detectable through particular detection strategies, a multi-layered approach using a variety of techniques are applied to ensure the greatest likelihood of detection. The core of this process is the automated data analysis and manual detection techniques. The automated data analysis techniques would include statistical and neural network based data mining such as: Random Forests, Artificial Neural Networks, and Support Vector Machines. This strategy is illustrated in Figure 1.

#### 3.1 Logical Structure of the System

Figure 2 illustrates the logical structure of the proposed solution. The logical demarcation of the major components include: Data Sources, Agents, Sensors, Analytical Data Storage, and the Data Mining Subsystem.

*Data Sources* include all sources of data for bot detection, which include network traffic data, system process information, and file system information. Of particular importance is the lack of reliance on any single type of data but rather support for all source data that may contain portions of data indicative of the existing of bots and botnets.

*Agents* gather specific information in the network and write it to the appropriate log-files or send it as network packets to *Sensors*. *Agents* may be passive or active. *Agents* gather information from all the available network data

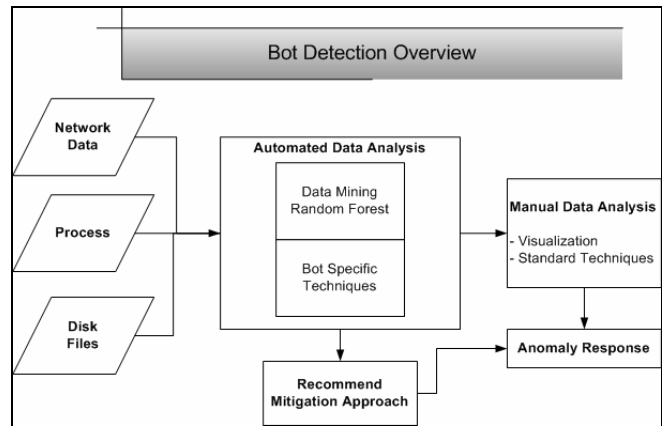


Figure 1: Bot Detection Strategy Overview

sources: network traffic data, system process information, and file system information.

*Sensors* monitor data in the packets sent by *Agents* and information in the *Analytical Data Storage* to compare them against *Alarm Patterns*. If an alarm pattern is met, the appropriate signal is sent to the Mitigation subsystem.

*Agents* and *sensors* are separated logically due to their different nature. In general, there may be no direct one-to-one relationship between agents and sensors. Moreover, there may be some agents with no connection to any sensor at all, e.g. an active sensor that initiates definite actions to entrap some bot. However, these roles are not necessarily separated physically. There is an option of keeping agent and sensor in one software module. The decision to separate them physically or not depends on a variety of factors, such as optimization of network loading.

*Analytical Data Storage* is the collection of uniformly stored log files and bot detection related data for the purpose of updating the threat patterns and training the data mining algorithms.

*Data Mining* subsystem analyzes information from *Analytical Data Storage* to update the training of the data mining algorithms and create or modify alarm patterns. The Data Mining subsystem can be considered stand alone from other components of the system. Other components are in constant connection and work together as a coordinated attack to detect bots and botnets. The data mining subsystem is activated periodically as it feeds other components with updated and trained data mining modules.

Some, but not all sensors will incorporate data mining algorithms. Logically, data mining-based sensors consist of two parts: algorithm and data to adjust the algorithm. The algorithm is a fixed trained data mining method. However, it might require some parameters to be adjusted (coefficients, patterns, etc.). The algorithm is produced by the data mining subsystem after some data mining method is researched, examined, and a trained usable instance of it becomes available. The updated data mining instance will then be uploaded to the appropriate sensor for modification and execution. Multiple algorithms differing in complexity and effectiveness can be implemented in the sensors.

The designed architecture will be effectively organized as a client-server system. The two main characteristics of this approach are:

- Hierarchical system that is server driven
- Optimization of system loading resulting in minimal impact on system resources

## 4 Bot Detection Algorithms and Techniques

The proposed algorithms used for bot and botnet detection are automated and consist of data mining and bot specific techniques. Bot detection activity can be characterized as either proactive and/or reactive. Proactive analysis is based upon conducting definite actions to find and identify potential problems that could be manifested in the future. Reactive analysis identifies manifestation of existing problems and determines their cause through diagnosis.

The proposed architecture uses a proactive method to bot detection. All of the existing, well-known antibot capabilities are limited to reactive analysis, which is bot signature detection. This approach is well examined and widely used, however its limitations result in:

- Insufficient reliability due to deviations in existing bots and botnets
- Obvious delay in responding to new bots and bot threats

An alternative approach to the direct inspection of bot signatures is detecting indirect circumstantial manifestations of bot activities such as bot data in network traffic. It was proposed in [12] that intelligent techniques based on behavioral analysis are the most promising direction for bot detection. For this reason we explore more extensively the implications of data mining techniques.

### 4.1 Data Mining

Data mining is the “nontrivial extraction of implicit,

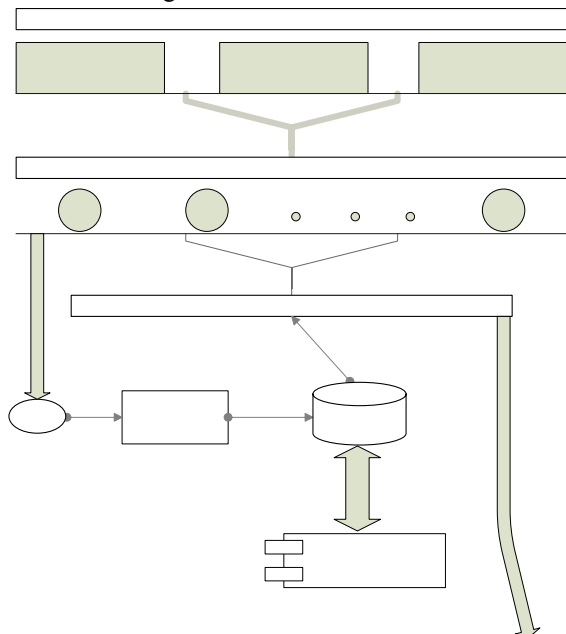


Figure 2: Logical Structure of the System

previously unknown, and potentially useful information from data” [10]. Methods of data mining include statistical data analysis, pattern recognition, artificial neural networks, support vector machines, etc. The goal with the integration of data mining methods is to provide a mechanism for the detection of new classes and variants of bots. The software architecture is designed to support multiple approaches implemented in the sensors all under the bot detection software architecture.

Two primary models of analyzing events to detect threats are:

- **Misuse detection model** [11]: the system detects intrusions by looking for activity that corresponds to known signatures or patterns of intrusions or vulnerabilities;
- **Anomaly detection model** [2][3][8][9]: the system detects threat or intrusion by searching abnormal behavior of the network. “Abnormal” behavior is detected as deviation from “normal” behavior predefined by the appropriate templates.

Both misuse and anomaly detection should be performed by the data mining algorithms implemented in the sensors. This will ensure maximum detection of novel classes and variants of bots. The proposed multilayer architecture of the antibot system provides advantages such as the easy attachment or detachment of different modules that have proved their suitability or unsuitability of bot detection. The effectiveness of different data mining methods can be examined and the most effective can be attached to the system. In the future, new methods can be added, and previously attached ones can be modified or patched as bot threats adjust and transform.

The following sections provide a detailed explanation of the most effective data mining algorithms that should be deployed with the architecture. The data mining algorithms under consideration are as follows:

1. Random Forests (RF)
2. Artificial Neural Networks (ANN)
3. Support Vector Machines (SVM)

Other methods may be evaluated and tested based on the effectiveness of the above algorithms, speed of execution, and current bot threats. Each of these data mining techniques uses very different techniques and algorithms and thus each will behave completely differently in the identification of bots and botnets. Using these techniques in conjunction will allow for a wider array of bots to be detected, limit the ability for bots to evade detection, and provide additional feedback as to the nature of threat.

#### 4.1.1 Statistical Data Mining - Random Forests

Random Forests [4] is an accurate multi-class prediction algorithm that can be used to predict either a categorical or continuous response.

RF works by fitting many classification trees and classifies a new instance by putting it down each tree in the forest and predicting that class getting the most votes from the forest. The predicted class for a new item is the most frequently predicted class over the collection of trees.

Random forests provide unique results that can assist in the analysis of identified threats, including:

- **Variable importance measures.** The variable importance measures are useful for understanding the data and selecting variables to use for mapping.
- **Intrinsic proximities.** RF provides a measure of proximity between each pair of cases.
- **Outlier detection.** Outlier detection aids in identification of errors, anomalies, etc.
- **Noise resilience** will allow RF to detect Bots and Botnets within typically noisy network traffic data.

These characteristics can aid identification of the class of botnet being identified and appropriate mitigation strategies.

#### 4.1.2 Data Mining – Artificial Neural Networks (ANN)

Another data mining technique that should be included into the software architecture is artificial neural networks [5]. ANNs can be considered one of the promising tools for detecting bot threats and attacks, both for misuse detection model and for anomaly detection model. ANN is an interconnected assembly of simple processing elements, units or nodes, whose functionality is loosely based on the animal neuron. The processing ability of the network is stored in the inter-unit connection strengths, or weights, obtained by a process of adaptation to, or learning from, a set of training patterns. Trained ANN could be used as a “black box” with input (pattern to be recognized) and output (class to which the pattern belongs). Neural nets can be trained in two main ways: supervised training and unsupervised training.

The most significant characteristics of ANNs are:

- Ability of self training
- Ability to find hidden interdependencies in raw input data. For algorithmically unsolvable tasks this allows the forecast result with given accuracy.

#### 4.1.3 Data Mining – Support Vector Machines (SVM)

Another data mining technique that would provide value is Support Vector Machines [7]. SVM is one of the most modern, most popular and most prospective of the statistical data mining methods. SVMs improve the standard methods of finding optimal separating hyperplanes. This makes it possible to construct linear decision surfaces in feature space which correspond to non-linear decision surfaces in input space. One more advantage of SVMs is that for training a particular SVM, a very limited vectors subset of the whole training set of examples is used (called support vectors).

## 4.2 Bot Specific Techniques

In conjunction with data mining, bot specific techniques will be used for bot and botnet detection. Bot specific techniques include specific detection techniques targeted towards known bots or bot functions, e.g., key logging, IRC, or HTTP. One of the many possible approaches is the incorporation of antibot bots on client nodes in a network for the sole purpose of detecting other bots. Capabilities should be included such as detecting probing hosts or “bot spoofing” with typical bot command sequences in order to acquire a response. Additionally, tools will identify changes

on a system, in real-time, indicative of an inappropriate modification typical of bots. These tools should identify the potential compromise rapidly, as soon as a bot begins installing and modifying the OS configuration. A variety of tools can be integrated as techniques are discovered to combat bots. Specific tools include but are not limited to the items shown in Table 2. These tools will be bundled into the antibot bot and/or into the server tools.

Server-based probing to send known commands to local hosts	Client-based system probing for examination of local host	Real-time open port monitor
Real-time process monitoring	Real-time security monitor	Disk examination
Real-time registry monitoring	Real-time system load monitor	Process examination
Real-time disk monitoring	Real-time disk usage monitor	Examine Registry
Real-time process DLL monitoring	Real-time network usage monitor	

One of the bot specific techniques is signature analysis, which is the most popular method of detecting malicious activities. It is the main method used in well known tools and applications by Symantec, McAfee, Trend Micro, Sophos and others. The main purpose of the signature analysis is comparing the ongoing network events against the known attack signatures.

Two main approaches exist to detect bot activities with signature analysis of the network traffic: analysis of the headers of network packets and analysis of the packets content. Theoretically, full analysis of all the traffic might be the best method; however it is never used due to the obvious dramatic decrease of the network productivity. In addition, it is often useless when encrypted data is used.

#### 4.2.1 General Algorithms

Let us consider using the signature analysis of the network traffic to detect bot activities and to detect known bots. Signatures of known bots and known bot attacks are stored in the Analytical Data Storage of the system. Each signature contains distinguishing characteristics of network packets that might be sent to/from a bot.

The appropriate pair Agent/Sensor monitors the ongoing traffic to find the known signatures in it. An alert is issued when a bot signature is met. This approach determines bot manifestations very accurately. However, it can be applied only for the bots with known signatures.

#### 4.2.2 Example of Use

Signatures of known bots and known bot attacks are placed into the Analytical Data Storage of the system. The appropriate set of rules is placed into the Analytical Data Storage. Each rule contains the description of particular characteristics of an infected packet and an action that is assigned to such packet (e.g. logging this event in some file; sending alert to the administrator of the network, etc.). The rule can also initiate some additional activities such as analyzing the contents of the packet.

Traffic scanner (Agent/Sensor pair) is placed in the network. This scanner compares headers of network packets against the rules describing the known bot signatures and performs the actions given in the rules.

The algorithm described above provides thorough control over network traffic, with minimal, unnoticeable impact on network productivity.

Signature analysis can be combined with data mining methods to achieve the results unrealizable by each method being used separately.

### 4.3 Bot /Botnet Mitigation

Upon the identification of a bot, the system will generate an alert identifying as to what system was affected and the best remediation and mitigation strategy. It is here that data mining, particularly RF, will demonstrate its added benefit as not only will the system identify the most important parameters used in differentiating the data elements, but will also provide the ability to derive attribution information. Thus, if it was RF that provided the results then the importance parameter will aid identification of the exact type of bot and the attribution information will aid identification of the compromised system. This capability is not available with other techniques.

Once a bot or botnet has been detected, the classification information is sent to the user and to the mitigation subsystem. An appropriate mitigation strategy will be recommended by the system listing all actions necessary to mitigate a bot attack. No automated mitigation approach will be implemented; user intervention will be required. The recommended operations include but are not limited to the following:

- Physically disconnecting infected computers from the network
- Considering an option of immediate blocking all outbound traffic to external networks
- Implementing filters on internal routers, firewalls and other networking equipment as appropriate to isolate infected segments and to monitor network traffic to ensure internal containment or identify how this infection is spreading and which hosts are infected
- Monitoring all network traffic in order to address possible multifaceted attacks
- Reviewing appropriate log files to attempt to identify the first system infected and what the attack vector was
- Removal of bot and botnet from the system
- Notification of users and external cyber support groups per policy
- Reinstall OS of infected systems (from Ghost image)
- Fully follow all BOT packet streams, for analysis and additional detection
- Contact ISP or Network provider (company, organization, etc.) of BOTNet offenders
- Perform additional forensics on affected systems (possible additional exploitation).

### 4.4 BOT Classification and Detection Approach

There is no industrial standard for bot classification. However, all noted sources use the same approach to classify bots. In general, this approach can be reduced to the following:

- Bots are divided into two main groups “good” bots and “bad” or malware bots
- According to the Honeynet.org group and to the main producers of antivirus software, malware bots can be divided into 9 main classes:
  1. Lisp IRC Bots
  2. Click bot or hitbot
  3. Agobot/Phatbot/Forbot/XtremBot
  4. SDBot/RBot/UrBot/UrXBot
  5. mIRC-based Bots - GT-Bots
  6. DSNX Bots
  7. Q8 Bots
  8. Kaiten
  9. Perl-based bots.

Based on the design of the proposed system, the table on the following 2 pages ranks each of the bot types in order of severity (most severe to least), describes the type of threat and how it works, and identifies how to detect and mitigate the bot/botnet. There are multiple variations to each of these bot types; in fact each of these classes can be split into numerous families. Portions of this table were derived from [19]. Alternative metrics have been proposed by Akiyama et al. [1]. A more organized taxonomy based on typical features and characteristics of bots has been proposed by Trend Micro [13].

The following method was used to evaluate severity:

$$SeverityIndex = \sum_{i=1}^n k_i * P_i$$

Where,

$n$  - number of parameters that most influence bot severity;

$P_i$  - value of the parameter #  $i$  (10-scored);

$k_i$  - weighting coefficient of the parameter #  $i$ .

Three main parameters are proposed:

- Spreading ( $P_1$ ) – how widespread is the bot
- Destructiveness ( $P_2$ ) – level of harm of the given bot
- Ability to be distributed ( $P_3$ ) – parameter that characterizes the ability of the bot to propagate.

### 5 Bot Detection Example

As an example, the following illustrates how the system would detect a particular bot, such as Agobot. It is widely known and has thousands of modifications and even families of ensuing bots such as Phatbot, Forbot, and XtremBot.

<b>Bot class</b>	<b>Severity Index (1 – 25) 25 is highest</b>	<b>How it works</b>	<b>OS</b>	<b>Kind of threats</b>	<b>Signature</b>	<b>Data sources</b>	<b>Detection method or tool</b>	<b>Mitigation</b>
Agobot/ Phatbot/ Forbot/ XtremBot/	17.4	Uses IRC-port and P2P net (Phatbot) for messaging. Spreads using numerous vulnerabilities in OS, applications, via P2P applications such as Kazaa, Grokster, and Bear Share, and via network shared drives.	Windows	Releases confidential info (steals the CD keys of several popular computer games, steals Windows product ID) Unauthorized remote access to computer Kills processes, belonging to antivirus and firewall software	Signatures of existent bots are usually available at the specialized web sites (e.g. <a href="http://www.lurhq.com/phatbot.html">http://www.lurhq.com/phatbot.html</a> ) and at the web sites of the known antivirus products (e.g. Symantec, Sophos, McAfee, etc.). However, the bots belonging to the Agobot class obtain dozens of new derivatives each day, and moreover, some versions use Polymorphic Encryptor Engine to encrypt the code. All the above means that signature analysis is ineffective for this class.	Registry settings Executables in system folders of the Windows (please see Chapter 3.3 above for the details) Network traffic	Data Mining methods: Neural Networks, SVM, Expert Systems, etc.	Close IRC ports, and P2P ports,  Block attempts to get access to admin accounts
SDBot/ RBot/ UrBot/ UrXBot	15.3	Uses IRC-port to receive commands Spreads exploiting vulnerabilities in Windows operating systems and via network shared drives.	Windows	Unauthorized remote access to computer (Executing programs, Opening files Downloading files, Redirecting information sent to a local port to a remote port , sending system information from the local host, such as operating system, processor speed, free ram, etc. ) File deletion	Signatures exist, but ineffective - There are dozens of new derivatives each day	Registry settings Executables in system folders of the windows Network traffic	Data Mining methods: Neural Networks, SVM, Expert Systems, etc.	Close IRC ports
mIRC-based Bots - GT-Bots	13	Uses mIRC as a core. Uses IRC- channel. Spreads using e-mail attachment or downloads via the hacker's site.	Windows	DDoS, Files installation and deletion	Existence of m-IRC scripts as well as of the m-IRC software	Files (existence of mIRC scripts). Network traffic (high volume of traffic)	Bot signature analysis (looking for mIRC scripts), Data Mining methods: Neural Networks, SVM, Expert Systems, etc.	Block excess traffic Close IRC-port
DSNX Bots	12.4	IRC-Channel Spreads using e-mail attachment or downloads via the hacker's site	Windows	Allows for unauthorized access to a computer (Create a proxy server on the infected machine; Delete, download,	Some signatures of existent bots available at the web sites of the known antivirus products (e.g. Symantec, Sophos, McAfee).	Network traffic (data of the IRC protocol) Files (looking for the signatures)	Bot signature analysis, Data Mining methods: Neural Networks, SVM, Expert Systems, etc.	Close IRC-port

<b>Bot class</b>	<b>Severity Index (1 – 25) 25 is highest</b>	<b>How it works</b>	<b>OS</b>	<b>Kind of threats</b>	<b>Signature</b>	<b>Data sources</b>	<b>Detection method or tool</b>	<b>Mitigation</b>
				execute, files; Flood a specified IP address; Load program plugins; Log keystrokes; Perform port scan on local network; Redirect TCP traffic to a remote site; Terminate and uninstall the program; Visit URLs)				
Click bot or hitbot	10	Uses IRC-port to communicate with hacker Spreads using e-mail attachment.	Windows	Click Frauds DDoS attacks	Signatures of existent bots are usually available at the web sites of the known antivirus products (e.g. Symantec, Sophos, McAfee). However, the bots belonging to the Clickbot/Hitbot class are very easy to implement from scratch, so signature analysis might be ineffective for them.	Network traffic (high volume of http traffic) File system (definite signatures can be found in files) Working applications and System Processes (watching for users' interaction with applications)	User Intension Analysis	Close IRC-port
Q8 Bots	6.9	IRC-Channel	Unix/Linux	DDoS (SYN-flood and UDP-flood). Execution of arbitrary commands	Has core algorithm (926 lines of C code)	File system (known C code of the kernel). Network traffic (excess activity)	Bot signature analysis, Data Mining methods: Neural Networks, SVM, Expert Systems, etc.	Close IRC-port
Kaiten	6.1	Uses IRC-Channel	Unix/Linux Windows	DDoS attacks Download files from a Web site of the hacker's choice Run commands or files of the hacker's choice	Current signatures can be found in antiviral databases, however, the bot can be modified, and the signatures will change	File system (known signatures) Network traffic (known commands in the IRC-port Registry Settings (for Win32,Kaiten)	Bot signature analysis Data Mining Methods: Neural Networks, SVM	Close IRC-port
Lisp IRC bots	4.3	Lisp commands to process operations Uses IRC-port for C&C communications	Windows Unix/Linux (rarely)	DDoS attacks	Lisp command el-irc library exists on computer	File system (find lisp commands or libraries) Network traffic (find appropriate commands in traffic)	Signature analysis Data Mining methods: SVM, Neural Network	Close IRC-port
Perl-based bots	3.3	Uses IRC-Channel for C&C communications Has limited basic set of commands	Unix/Linux	DDoS attacks	There are no constant signatures of this bot, since the bot itself is very small, and consists of several hundred lines of code that are usually rewritten anew.	File System (existence of suspicious perl-instructions that use IRC commands) Network traffic (existence of definite IRC-commands)	Data mining methods: SVM, Neural Network	Forbid IRC connections from Perl-code

The following procedure can be used to detect such bots:

- Examine registry folders HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run and HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices to compare all the files mentioned there for starting, with those listed in the system folders (Windows\System, Windows\System32). Quite often the malicious files of such bots are named similar to the system files, e.g. svchostt.exe vs. svchost.exe.
- Find whether connections to P2P networks are open
- Find whether many password-selection tries to get access to default administrative shares (e.g. IPC\$, admin\$, C\$, D\$, E\$ and print\$) are being held
- Use sniffing to check whether the software meeting the requirements above has a connection to IRC channels(s).

## 6 Conclusions

The key advantage of the architecture designed in this research is that it allows for the integration of wide ranging techniques. We do not limit the architecture to supporting only a single type or class of detection algorithms. By allowing algorithms from other researchers to be integrated through the open architecture we allow for the greatest possible detection strategy.

We examined some specific techniques that should be included in such an architecture. This includes the bot-specific techniques as well a range of data mining techniques. Each of the different data mining techniques has advantages and disadvantages as far as the detection of bots goes.

Additionally, the architecture examines all sources of available data in a client-server architecture. While many analysis techniques will run on the individual hosts with the isolated local data, additional techniques run on the server over collated network-wide data. This allows for both rapid detection and robust detection with multiple levels of fail-safe to ensure critical events or correlations are not missed.

Additionally, the client-server architecture allows network administrators to control the extent to which bot detection is performed on each individual host as well as the server. This can be adjusted dynamically dependent on the current level of threat in the environment.

Finally, the described architecture provides an extensive set of capabilities for managing bot detection, including:

- Detection of bot activities
- Mitigation of bot threats and attacks
- Notification of users about current bot threats
- Ability to extend the system with new antibot modules
- Ability to upgrade previously installed antibot modules
- Ability to be adjusted and tuned to meet the exact requirements of network users
- Provide status of the current state of the network and network traffic
- Provide status of the current state of the processes on nodes in the network
- Provide status of the current state of file system
- Unification of all the gathered data

## 7 References

- [1] Mitsuaki Akiyama, Takanori Kawamoto, Masayoshi Shimamura, Teruaki Yokoyama, Youki Kadobayashi, and Suguru Yamaguchi, "A proposal of metrics for botnet detection based on its cooperative behavior," *Proceedings of the SAINT 2007 Internet Measurement Technology and its Applications to Building Next Generation Internet Workshop*, January 2007.
- [2] James R. Binkley and Suresh Singh, "An Algorithm for Anomaly-based Botnet Detection," Computer Science, PSU, *USENIX SRUTI: '06 2nd Workshop on Steps to Reducing Unwanted Traffic on the Internet*, July 7 2006.
- [3] James R. Binkley, "Anomaly-based Botnet Server Detection," Computer Science, PSU, *FLOCON CERT/SEI*, Vancouver WA, October 2006.
- [4] L. Breiman, "Random forests," *Machine Learning*, 2001, 45(1), 5-32.
- [5] Robert J. Brown, "An Artificial Neural Network Experiment," *Dr. Dobbs Journal*, April 1987.
- [6] Evan Cooke, Farnam Jahanian, and Danny McPherson, "The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets," *Proceedings of the 2005 Usenix Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI '05)*, June 2005.
- [7] Corinna Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning*, 20, 1995.
- [8] D. Denning, "An Intrusion-Detection Model," *IEEE Transactions on Software Engineering*, 13(2), Feb. 1987.
- [9] Jonathon W. Donaldson, "Anomaly-based Botnet Detection for High-Speed Networks," Thesis, Rochester Institute of Technology, Rochester, New York.
- [10] W. Frawley and G. Piatetsky-Shapiro and C. Matheus, "Knowledge Discovery in Databases: An Overview," *AI Magazine*, Fall 1992, pp. 213-228.
- [11] P. Helman and G. Liepins, "Statistical Foundations of Audit Trail Analysis for the Detection of Computer Misuse," *IEEE Transactions on Software Engineering*, 19(9), September, 1993.
- [12] Sandvine, "Dynamic Botnet Detection," *Sandvine White Paper*, June 2006.
- [13] Trend Micro, "Taxonomy of Botnet Threats," *Trend Micro White Paper*, November 2006.
- [14] Moheeb Abu Rajab, Jay Zarfoss, Fabian Monrose, Andreas Terzis. A Multifaceted Approach to Understanding the Botnet Phenomenon. *In Proceedings of ACM SIGCOMM/USENIX Internet Measurement Conference (IMC), Oct., 2006. Rio de Janeiro, Brazil.*
- [15] Open Source Security Information Management (OSSIM), <http://www.ossim.net/>
- [16] Prelude IDS, <http://www.prelude-ids.org/>
- [17] McAfee, <http://www.mcafee.com/us/>
- [18] Norton Antivirus, <http://www.norton.com/>
- [19] Know your Enemy: Tracking Botnets, <http://www.honeynet.org/papers/bots/>