System for Detection of Network Threats based on Classifiers

Bilgin Demir[1], Zoran Gacovski[1], Vladimir Pivovarov[1], Lidija Goracinova[1]

[1]*FON University, Bul. Vojvodina, bb, Skopje, Macedonia*

***Abstract* – In this paper we present a system that automatically detects and profiles threats on a real network. The realised Threat Detection System (TDS)is based on Snort software and it allows the security experts to evaluate the risk of vulnerability and to retrieve the actual number of threats that are active in the network.Algorithms are presented to determine three properties for each threat: skill, intensity of the attacks and whether the threat is a human or an autonomous computer program.**

***Keywords* – Network vulnerabilities, Intrusion detection, Classification methods.**

1. Introduction

Traditional Intrusion Detection Systems focuses on detecting attack instances on a computer network. The work presented in this paper is different, since the attack instances are used to construct a profile of the actual attacker. An attacker is named "threat" and it can be human or autonomous computer program. The system is therefore named a Threat Detection System (TDS) and it will improve the security of a computer network[1].

Security is often defined as a combination of confidentiality, integrity, and availability of assets [2]. Security means that assets (for example some piece of information) must be kept secret, we must be able to keep the integrity of that piece of information, and the asset should be available to legitimate users when they need it. In the new world based largely on the Internet, security is still important for the same reasons.

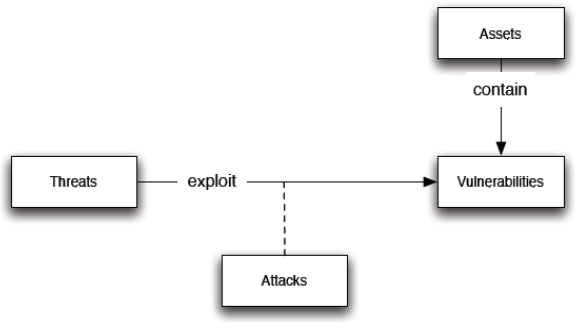

Figure 1. Definition of threat, exploit, attack, vulnerability and asset

When computer systems are connected to the Internet - the systems themselves and the information they contain are assets, and their security can be threatened by hackers, worms, viruses and botnets, which are named threats[2],[3],[7]. These threats, which can be individuals or computer programs, search for weaknesses (named vulnerabilities) of the computers and information (named assets),and try to exploit them using attacks, compromising the assets' confidentiality, integrity, availability, or a combination of those [2], [12]. The set terms are illustrated in Figure 1.

Computer system is secure when threats cannot exploit the vulnerabilities of that system. There are two approaches towards secure computer systems[2]: The first approach consists of removing the vulnerabilities of the system or adding defenses that maketheir exploitation harder. The time and skills required for a threat to compromise the system increase, and the threat will (hopefully) stop attacking voluntarily. It is impossible to locate and eliminate all vulnerabilities of a system, therefore using only this first approach is not the best manner to secure the system [2]. The second approach consists of threats detection while they are attacking the system and stopping them before they are successful in compromising it, for example by removing their access to the system and involving the police. A combination of these two approaches is best, since reacting to threats takes time that can be obtained by eliminating the vulnerabilities [2].

2. Security threats for network environments

The security of a computer network is based on three components: confidentiality, integrity, and availability. These components apply to the assets (data or systems) within the network [12]. To protect these assets, there are two basic approaches that are commonly implemented[2].

The first approach to network security is analog to building a large wall around a city; the bigger and higher the wall is - the harder it is for threats to get inside the city. It is universally accepted that there is no such thing as perfect security, which means that a threat with sufficient capabilities and resources is always able to climb over the wall.

The second approach to network security consists of monitoring the network for threats. Using firewalls, those threats can be stopped before they compromise the network.

The best network security is achieved using a combination of the two, making it hard for a threat to compromise the network and allowing the network security staff to stop them before they succeed. This combination could be modeled in Risk Management practices by defining the process of threatstopping as a control that decrease the likelihood - threat can exploit a vulnerability [11],[15].

3. System for detection of threats

This section describes the architecture and implementation of our Threat Detection System (TDS). The requirements for the system are defined in the previous sections and are the following:

1. Detect threats on a real computer network.
2. Determine the profile of these threats using threat properties that can be configured or defined by security experts.
3. Implement a skill, android, and intensity property as examples.
4. Allow a security expert to specify a threat profile and timeframe and return the matching threats.

Based on these requirements for the prototype TDS, the following decisions are made:

1. The TDS is plug-and-play. There is no time, and money required to adapt the system for a specific network. This allows the system to be used directly in a new network.
2. The TDS connects passively to the network, for example using a SPAN port on a switch or router. Therefore, the system does not influence the integrity and availability of the network in any way. An alternative would be the installation of sensors on each computer (to allow host based intrusion detection), but then the system will not be plug-and-play.
3. The TDS supports IPv4. In the future, it would be useful to include IPv6, but this is beyond the scope of this prototype.
4. The TDS supports HTTP. Nowadays, most attacks (60% of the attacks observed on the Internet) are performed on websites and limiting the scope of the prototype to this protocol demonstrates the work of TDS without the implementation of every possible protocol.

We will now describe the architecture of the Threat Detection System (TDS). First - the ideas behind the system and the function of each component will be given. The actual implementation of these components can be arranged to adapt the system to different environments.

The most important concept behind the Threat Detection System is the shift from detecting attacks to detecting the actual attackers (threats) and their properties (such as skills). Detecting threats is accomplished using the output of one or more Intrusion Detection Systems (each detecting attacks) and clustering the attacks in groups of attacks that belong to a single threat. The result of this process is a list of threats and their attacks. The Intrusion Detection Systems are not required to detect 100% of the attacks, since one attack for each threat is theoretically sufficient to detect that threat. Then, for each threat, a set of properties (like skill) is determined based on the attacks they performed or some external knowledge database. The accuracy of the properties determined for each threat depends on the amount of attacks detected for this threat; more detected attacks of a threat - increase the accuracy of the properties determined for this threat.

The TDS therefore consists of three components which are illustrated in Figure 2: one or more Intrusion Detection Systems (IDS), a threat classifier and one or more threat profilers. This architecture allows one component, for example, the implementation of the IDS, to be replaced by different IDS or to use multiple IDS concurrently. It is also possible to scale the system by installing many IDS components on separate systems,so that all report to one threat classifier[13],[14].

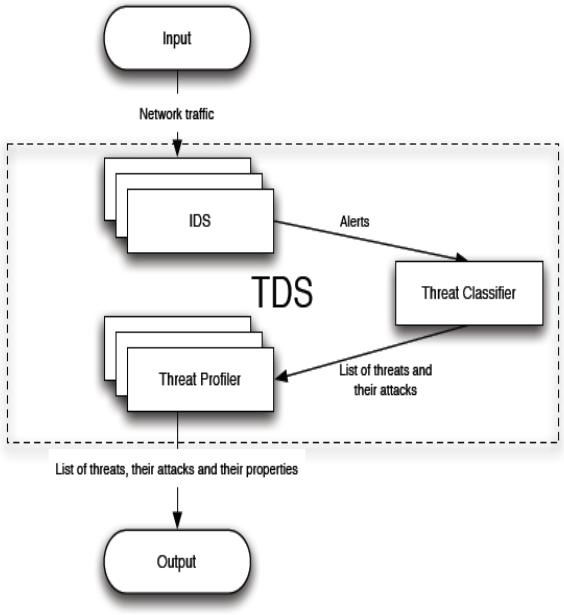

Figure 2. Components of the TDS

4. Intrusion Detection System

The Intrusion Detection System (IDS) is responsible for detecting attacks in the network traffic. Different types of IDS could be used, but they must be able to detect actual threats on actual networks. To allow the threat profiler to determine the skill of a threat, the IDS must be able to identify the attack method used for each attack. The architecture of the system does not require 100% detection rate from the IDS, since a few attacks of each threat already allow the threat to be detected and profiled [4]. When more attacks of the threat are detected, the accuracy of the threat profile increases. There are many Intrusion Detection Systems available such as Snort, BRO, Mod-security and OSSEC.

The classifier (a clustering algorithm) translates the alerts generated by the IDS into a list of threats and their attacks [16]. This translation can be performed based on (a combination of) the MAC address, the IP address, or the user name, depending on the network traffic. The architecture does not dictate a specific way of classifying the threats, but the performance of the system increases when the classifier is more accurate in grouping the alerts per threat[5],[6].

The threat profiler is responsible for determining a specific property of a threat, such as the skill. To perform its function, a threat profiler receives a list of threats and their attacks from the threat classifier component. It is important to note that only concrete properties can be determined. Abstract properties like “intent” and “motivation” seem logical properties to profile, but cannot be reliably assessed and should not be used. Subjective properties that depend on some interpretation, such as skill, can be used when they can be reliably assessed given a specific interpretation, but the interpretation must be configurable since other users of the system could disagree with the interpretation and want to change it[8],[9].[10].

The architecture of the Threat Detection System provides a novel way of intrusion detection: a shift from detecting attacks to detecting threats and their properties. The architecture allows the system to be adapted to different environments and to use third party components when available.

5. Simulation results

In this section, the testing methodology and results are presented for a final evaluation of the system. The results show if the research meets the problem definition:

Design a system to detect and profile threats in a computer network to improve risk management and network security monitoring.

An important measure to evaluate IDSs, is the rate of false positives and false negatives. Ideally, these measurements are performed using a labeled data set that is representative for the purpose of the system. In this case, a data set that contains attacks of threats which are annotated (labeled), could determine the number of users that are falsely detected as being a threat (false positive rate), and the number of threats that are not detected at all (false negative rate). The only publicly available data set that is active and provides labels for each threat and attack, contains only the net flow information and not the full contents of the network packets [15] and is therefore not suitable for this research.

To evaluate if the system provides meaningful information to security experts performing risk management and network security measurement, the results of the system are analyzed in two scenarios. First, a test is executed by placing the system in a real network for 4 days. For this network, the threats and their attacks are unknown, but all detected threats can be manually evaluated and false positives can be found, providing an indication of the false positive rate of the system. In the second test, the system is placed in a network containing only threats. The network hosts a web hacking challenge and is therefore not representative for real world scenarios (the attacks are similar, but in the real world the threats have no permission to attack the sites and therefore could behave differently). Since the threats are known and fill out a questionnaire, this test allows to test the false negative rate of the system which is not possible in a scenario that is truly representative for the real world containing unknown threats and unknown attacks.

a)Real network experiment

The first test is the baseline scenario. During the development of the system, it was placed within a real network hosting around 100 websites for the duration of 28 days to get a general idea of the systemfunctioning and setting the system parameters. In this experiment, the system is placed within the same network for four days to evaluate the performance of the TDS in a real network.

IDS results

The IDS component of the TDS is responsible for detecting the actual attacks on the network. The false positive and false negative rate of this component have direct influence on the performance of the TDS. However, the false negative rate of this component

should not have that much impact on the performance of the TDS, as long as some attacks are detected for each threat.

b)Hacker experiment

To evaluate the ability of the TDS to identify the skill of threats, an experiment is conducted by inviting hackers to compromise a website and then complete a questionnaire to identify their skill level. For each hacker, the skill level identified by the questionnaire is compared to the skill level as reported by the TDS, to calculate the correlation between them.

Question

The main question of this experiment is whether the TDS can accurately profile the threats: *What is the correlation between the skill level detected by the TDS and the skill level determined by a questionnaire?*

Hypothesis

The hypothesis of this test is: *If hackers are invited to attack a computer that is monitored by the TDS and complete questionnaire to assess their skills, then there is a strong, positive correlation between the two, which is statistically significant.*

Experiment

This experiment consists of the following steps:

1. The link to this experiment is posted on hacker forums and invites professional penetration testers to participate.
2. Participants are provided with a description of the experiment.
3. Participants are provided with the same guidelines.
4. Participants provide their e-mail address and receive a link to the website containing unique identifier.
5. Participants continue to the website containing the following vulnerabilities
    i. Accessible non-parsed dynamic script in login.php.bak.
    ii. Cross Site Scripting in the 'leave a message' box.
    iii. Filename Injection Attack in the page parameter of the URL.
    iv. SQL Injection in the error parameter of the URL.
    v. XML External Entity Attack in the XMLRPC-Ajax communication.
    vi. XSLT Injection Attack in the search field.
6. Participants have a button at the bottom of the screen "finish hacking" to continue with the questionnaire of the experiment.
7. Participants answer questionnaire containing some questions (for example: gender, skills etc.)
8. If the participant answered he performed an advanced attack (accessible non-parsed dynamic scripts, XML External Entity, XSL(T) Injection), his level is defined as "advanced". If the participant answered to have performed only one or more basic attacks (Cross Site Scripting, Filename Injection or SQL Injection), his level is defined as "basic". When no attacks are performed, his level is defined as "none".
9. The skill level for each participant is determined using the TDS.
10. The correlation between the two skill levels for the participants is calculated using Spearman's Rank Order Correlation Coefficient.

Results

The experiment is executed from 2013-12-01 14:40:43 to 2013-12-21 12:05:12. During this time, the questionnaire was submitted 95 times. Using control questions in the questionnaire, 30 participants are filtered because they provided conflicting information or did not complete the questionnaire. An example of conflicting information is a participant stating that she had no knowledge of performing SQL injection attacks but also stating that she performed an SQL injection attack. Another three participants are filtered because they submitted the questionnaire multiple times. This leaves 59 participants that provided clean data.

Table 1.Number of male and female participants.

Gender	#	Fraction
male	58	0.98
female	1	0.02

The 59 participants are randomly selected, since the experiment was announced on different public mailing-lists and forums. Table 1 and 2 illustrate the demographic properties of the data set.

Table 2. Country of participants

Country	#	fraction
Argentina	2	0.03
Bhutan	1	0.02
China	1	0.02
Estonia	2	0.03

France	4	0.07
Guyana	1	0.02
Hungary	1	0.02
India	3	0.05
Israel	1	0.02
Italy	3	0.05
Mexico	2	0.03
Netherlands	13	0.22
Philippines	1	0.02
Portugal	2	0.03
Spin	8	0.14
United Kingdom	2	0.03
United States	12	0.20

Using a unique identification number for each participant, whom they received in their e-mail, it is possible to track the IP addresses of each participant. Using this data, it can be concluded that no IP addresses are used by two or more participants. Furthermore, there are 12 participants that used 2 or more IP addresses, but only one of these participants performed attacks from 2 different IP addresses. Within this data set, this participant represents 1.7% of the dataset, which supports the assumption that hackers do not commonly use multiple IP addresses. This statement does require further research, since this experiment allowed participants to compromise the system and therefore did not require them to hide their tracks.

Table 3. Hacker experiment: Signatures and matches

signature	matches	Checklist item
1120001	18	Check for default and predictable accounts
1120003	1	Check for default and predictable accounts
1210001	3616	Check for filename injection / path traversal
1210002	142	Check for filename injection / path traversal
1220001	592	Check for SQL injection
1220002	3750	Check for SQL injection
1220003	270	Check for SQL injection
1230001	1043	Check for cross site scripting
1230003	67	Check for cross site scripting
1230004	23	Check for cross site scripting
1230005	55	Check for cross site scripting
1230007	2	Check for cross site scripting
1240001	69	Check for system command injection
1310001	5	Check for uploading of (dynamic) scripts
20210001	2752	Check for extraneous files in document root
20340001	5	Check for accessible CVS/SVN directions
20340002	23	Check for accessible CVS/SVN directions
20370001	279	Check for accessible non-parsed dynamic scripts
20650001	115	Check for brute-force username enumeration
20760001	50	Check for XSL(T) injection
20770001	20	Check for SSI injection
20790001	5	Check for dynamic scripting injection
20810001	12	Check for XML external entity parsing

The 59 participants performed a total of 12915 attacks, illustrated in Table 3. The system contained filename injection, SQL injection, cross site scripting, extraneous files, XSL(T) injection and XML external entity vulnerabilities. The table shows that many other attacks were also performed, since the participants did not know which vulnerabilities were present in the system.

An import function of the TDS is the ability to assign the correct skill to threats. Using the questionnaire to determine the skill of the participant and the skill defined by the TDS, it is possible to calculate the correlation between the two. Using Spearman's Rank Order Correlation Coefficient, the correlation is calculated to be 0.598. In a data set out of 59 participants, this is statistically significant (a correlation of 0.30 is required for the significance level of 1%).

To verify the results of the questionnaire and the results of the TDS, a sample of 10 randomly selected participants is manually reviewed. During this manual review, the HTTP requests are analyzed and attacks are annotated. Then, using the same definition of skill as used in the questionnaire and the TDS, the skill of the participant is determined by the skill level of its most difficult attack. The results of this manual review are illustrated in Table 4. This table clearly shows that the skill of the participant is not adequately assessed using the questionnaire. The result of this experiment is therefore that the TDS can assess the skill of threats more accurately than a questionnaire could do. In a practical scenario, this means that the system will provide better results than the casewhen participants filled in a questionnaire.

Table 4. Skills determined by the questionnaire, manual review and the TDS

id	questionnaire skill	manual skill	TDS skill
1	none	basic	basic
2	advanced	advanced	advanced
3	basic	advanced	advanced
4	advanced	basic	basic
5	advanced	basic	basic
6	basic	basic	basic
7	basic	basic	basic

8	basic	basic	basic
9	basic	none	none
10	none	none	none

Finally, the android profiler can be evaluated using these results. The android profiler should identify each threat as being a human but failed to do this for 13 of the 59 participants (22%). The reason for this poor performance is the way the web application is set up; when the threat enters via attack in the search field, the field is transmitted to the server after each character using XmlRpc-Requests. Each request is detected separately as an attack, and the timing between the requests is inhuman.

5. Conclusion

The system presented in this paper automatically detects and profiles threats on a real network. When placed in a new network without any changes to its configuration, one of the 47 detected threats was falsely identified as a threat (a false positive) and two of the 47 detected threats were falsely identified as being a human. When placed in a network containing only human threats, the system correctly identifies the skill of each threat and identifies them as being human 78% of the time. To achieve this, three contributions to the professional and academic world are made.

First, the signatures for an open source intrusion detection system (Snort) are developed to detect many generic types of attacks (like SQL Injection and Cross Site Scripting). Currently, signatures used in such systems can detect only specific attack instances to known vulnerabilities. The presented set of signatures allows intrusion detection systems to detect attacks on unknown vulnerabilities.

Second, an extension on this intrusion detection system using the new set of signatures is developed shifting the focus from the attacks detected on the network to the actual threats (a human or autonomous computer program) attacking the network and their properties (such as skills). This paradigm shift is a new approach to intrusion detection, and the system is therefore named a Threat Detection System (TDS). By focusing on the threats instead of the attacks, the system is not required to achieve a 100% detection rate of attacks on the network. Threats are detected when they perform at least few attacks which are detected by the system.

Finally, algorithms are presented to determine three properties for each threat: skill, intensity of the attacks and whether the threat is human or autonomous computer program. These algorithms are accompanied by guidance on how to interpret their results and how to apply them to risk management and network security monitoring. This allows experts to directly use this system without changes to the configuration.

The presented signatures for the open source intrusion detection system allow us to research real threats on real networks and allow security experts to monitor actual attacks on their computer networks. The complete Threat Detection System allows security experts evaluating the risk of vulnerability to retrieve the actual number of threats that are active on the network and are capable of exploiting this vulnerability. It also allows security experts monitoring the security of a network to focus only on threats that are capable of compromising the network. Academics can use this system as basis for their own research on threats and adapt it to detect different properties or automatically respond to certain types of threats.

Combined, these contributions improve the accuracy of risk management and the efficiency of network security monitoring. This shows the basic purposeof this paper: “To design a system that autonomously detects and profiles threats on a computer network in order to improve risk management and network security monitoring”.

References

[1] Vacca, J.(2009).Computer and Information Security Handbook Morgan Kaufmann.

[2] Richard Bejtlich. (2007). Thetao of network security monitoring, Beyond Intrusion Detection. Addison Wesley.

[3] Sommerville, I., & Stevens, P. (2007). Software Engineering: AND Using UML, Software Engineering with Objects and Components.

[4] Ross, R., Katzke, S., Johnson, A., Swanson, M., Stoneburner, G., Rogers, G., & Lee, A. (2005). Recommended security controls for federal information systems. *NIST Special Publication*, *800*, 53.

[5] Swanson, M., Wohl, A., Pope, L., Grance, T., Hash, J., & Thomas, R. (2002).*Contingency planning guide for information technology systems: Recommendations of the National Institute of Standards and Technology* (No. NIST-SP-800-34).NATIONAL INST OF STANDARDS AND TECHNOLOGY GAITHERSBURG MD.

[6] Van Der Heijden, F., Duin, R., De Ridder, D., & Tax, D. M. (2005).*Classification, parameter estimation and state estimation: an engineering approach using MATLAB*.John Wiley & Sons.

[7] Gehani, A., &Kedem, G. (2004, January). Rheostat: Real-time risk management. In *Recent Advances in Intrusion Detection* (pp. 296-314).Springer Berlin Heidelberg.

[8]ChidanandaGowda, K., &Diday, E. (1991).Symbolic clustering using a new dissimilarity measure. *Pattern Recognition*, *24*(6), 567-578.

[9] He, W., Xia, C., Wang, H., Zhang, C., &Ji, Y. (2008). Game Theoretical Attack-Defense Model Oriented to Network Security. *Computer Science and Software Engineering*.

[10] Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: a review. *ACM computing surveys (CSUR), 31*(3), 264-323.

[11] Jain, A. K., &Dubes, R. C. (1988). *Algorithms for clustering data*. Prentice-Hall, Inc..

[12] Julisch, K. (2003). Clustering intrusion detection alarms to support root cause analysis. *ACM Transactions on Information and System Security (TISSEC),6*(4), 443-471.

[13] Miller, K. D. (1992). A framework for integrated risk management in international business. *Journal of international business studies*, 23(2), 311-331.

[14] Hoo, K. J. S. (2000). *How much is enough? A risk management approach to computer security*. Stanford University.

[15] Stoneburner, G., Goguen, A., &Feringa, A. (2002). Risk management guide for information technology systems. *Nist special publication, 800*(30), 800-30.

[16] Von Ahn, L., Blum, M., & Langford, J. (2004).Telling humans and computers apart automatically. *Communications of the ACM*, *47*(2), 56-60.

Corresponding author: Zoran Gacovski
Institution: FON University – Skopje, Macedonia
*E-mail:*gacovski@mt.net.mk