An Intelligent Firewall to Detect Novel Attacks

An Integrated Approach based on Anomaly Detection Against Virus Attacks

InSeon Yoo and Ulrich Ultes-Nitsche
Department of Electronics and Computer Science,
University of Southampton, Southampton, SO17 1BJ, UK.
{isy01r, uun}@ecs.soton.ac.uk

Abstract. We report in this paper on research in progress concerning the integration of different security techniques. A main purpose of the project is to integrate a smart detection engine into a firewall. The smart detection engine will aim at not only detecting anomalous network traffic as in classical IDSs, but also detecting unusual structures in data packets that suggest the presence of virus data. We will report in this paper on the concept of an intelligent firewall that contains a smart detection engine for potentially malicious data packets.

1 Introduction

Internet security breaches are growing. Viruses are one of the major causes of the rising number of security breaches. Nowadays some companies, which recognize the importance of security, adopt security systems such as firewalls, intrusion detection systems, and virus monitors. It is not easy to use all security systems owing to high cost. Furthermore, it is not enough to protect a company's system with only a single security system, because each security system has different features in present days, in which particularly DoS (Denial of Service) and virus attacks are becoming serious.

We report in this paper on an ongoing project[14] which is to integrate a smart detection engine into a firewall, creating what we call an *intelligent firewall*. Integrating techniques of different security systems appears to offer interesting possibilities. This project is fulfilling such a task. One of the main features of the intelligent firewall will be an anomaly detection capability, which will be achieved by applying AI techniques to detect unusual network traffic (an intrusion detection capability) and to detect unusual content of data packets (a capability to identify unknown potential viruses).

2 Background

2.1 Increasingly Serious Attacks

A virus is a piece of code that adds itself to other programs and cannot run independently. As Microsoft Windows became popular, windows viruses and windows-application-derived viruses using VBA (Visual Basic for Applications) spread widely. A common way of windows virus dissemination is through emails. In addition, worms are programs that can run by themselves and propagate a

Mária Bieliková (Ed.): SOFSEM 2002 Student Research Forum, pp. 59–64, 2002.

fully working version of themselves to other machines. The recent important one was Code Red. The Code Red worm is a malicious self-propagating code [1] that spreads surreptitiously through a hole in certain Microsoft software. Code Red, which leaves computers open to hijacking, has caused a lot of traffic being sent, clogging the bandwidth on the Internet. An infected system will show an increased processor and network load. The worm could easily permit hackers to take control of hundreds of thousands of infected machines. The worm has this magnifying effect on network traffic during attacks on internal networks.

DoS (Denial of Service) attacks can interrupt services by flooding networks or systems with unwanted traffic. A service will be denied because the network/system is overwhelmed. Distributed systems based on the client/server model have become increasingly popular. Using this scheme, DDoS (Distributed Denial of Service) attacks are also getting escalated. In DDoS form, an attacker controls a number of handlers. A handler is a compromised host with a special program running on it. Each handler is capable of controlling multiple agents. An agent is a compromised host, which is responsible for generating a stream of packets that is directed toward the intended victim.

2.2 Present Network Security Systems

Virus monitors examine network traffic, aiming to prevent malicious code from entering network nodes by detecting *known* malicious-code patterns. Apparently, they can detect only known viruses. New viruses will only become detectable after their pattern characteristics have been analysed and are made available. Current IDSs(Intrusion Detection Systems) do not prevent an intrusion from happening; they only detect and report it. When a virus associated with a DoS (Denial of Service) attack spreads through the Internet (e.g. the CodeRed), virus monitors and IDSs should co-operate to prevent such an attack. However, although virus monitors and IDSs are installed, new virus information needs to be updated constantly. Moreover, firewalls are used to guard and isolate connected segments of inter-networks. "Inside" network domains are protected against "outside" un-trusted networks, or parts of a network are protected against other parts [10]. These firewalls use only TCP/IP headers - no payload information is used - to detect whether data packets are safe or not.

3 Applying Ideas From Intrusion Detection Systems

3.1 Anomalous Pattern Recognition

IDSs very frequently run a process known as anomaly detection. An IDS based on this paradigm will constantly monitor network traffic and compare the stream of network packets with what it perceives as *normal* network traffic. As soon as it observes an anomalous pattern in the traffic it will throw a warning for the network administrator to deal with it. Anomaly detection appears to be applicable not only to intrusion detection but also to virus monitoring [9], now not being applied on the level of the full network traffic, but to single data

packets: The improved virus monitor will examine data packets as usual. But besides checking against known malicious-code patterns, it will check whether it sees a pattern that it perceives as *potentially malicious* and will react accordingly, e.g. by creating a warning of some sort.

3.2 Neural Network Models

Nowadays most practical IDSs are signature-based systems. These systems work based on predefined descriptions of attack signatures. Various data-source and type-of-pattern recognition techniques are still used. However, many known attacks can be easily modified to present many different signatures. If not all variations are in the database, a known attack may be missed. Some more modern approaches try to exploit neural-network-based techniques. To explore attack spaces, [11] uses a hierarchy of back propagation (BP) neural networks (NN) for the protocol and the self-organizing map (SOM) technique for the anomaly classification. In an experiment, however, only artificially generated data was used. [6] uses a classical feed-forward multi-layer perceptron network: a back propagation neural network and time delay neural network to program-based anomaly detection. Lastly, to identify and classify network activity based on limited, incomplete, and nonlinear data sources, [2] presents an analysis of the applicability of neural networks. In the neural network architecture, multi-layer perceptron (MLP) and hybrid forms (MLP + SOM) are adapted.

4 Need of the Intelligent Firewall

The intelligent firewall is our proposal for increasing the strength of firewalls. It will have the ability to examine entire data packets and to apply standard as well as intelligent detection techniques to identify misuse.

4.1 Data Packet Detection

In order to identify viruses/worms in e-mail attachments, one exploits that a virus has a unique character, the *virus signature*. In addition to the usual network control ability of a firewall, data packet detection is necessary in the intelligent firewall to identify packets containing malicious data: Unique characters will also appear in the data packets. For instance, the beginning of the Code Red's attack packet looks like the following [7]:

```
GET/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN%u9090%u6858%ucbd3
```

4.2 Dynamic Packet Handling Ability

Although a firewall is able to control a network and maintain its connectivity, it handles packets only statically. Through open ports, a firewall would not

inspect/control packet willingly. According to the analysis of distributed denial of service attack tools, it is well known how to use that TFN [3], TFN2K, Trinoo [4] and Stacheldraht [5]. These programs not only use TCP and UDP but also ICMP packets. Moreover, because the programs use ICMP_ECHOREPLY packets for communication, it will be very difficult to block attacks without breaking most Internet programs that rely on ICMP. Since TFN, TFN2K and Stacheldraht use ICMP packets, it is much more difficult to detect them in action, and packets will go right through most firewalls. The current only sure way to destroy this channel is to deny all ICMP_ECHO traffic into the network. Furthermore, the tools mentioned above use any port randomly; it is hard to prevent a port from malicious attacks in advance using the fixed port close scheme in current firewalls. Therefore, to prevent degradation of service on the network and to deny this kind of malicious packet, dynamic packet handling on the level of firewalls is crucial.

4.3 Intelligent Detection Engine

To prevent malicious self-propagating virus attacks from entering intranets, dynamic filtering of data packets is compulsory. Having the ability of anti-virus systems and IDSs, a malicious-data-packet-detection engine needs to be developed. Like a firewall, this system would have the ability to accept or deny packets. Equivalent to an intrusion detection system, it would examine a packet's content, flags, and headers to make a decision. As we examine virus packets filtered by MailScanner, which is running in the Southampton ECS intranet, the front parts of these packets display similar patterns. And indeed, if we capture "good packets", which have already been filtered by the ECS firewall and MailScanner, these good packets are different. Most front parts of the good packets display a variety of patterns. In order to capture the investigated packets we have used the *libpcap* [8] packet capturing ability of *snort* [12].

However, in some of the good packets, we could see patterns very similar to the ones possible in packets containing malicious code. These were packets sent by Microsoft Servers to NetBios and DNS lookup services. For example, port 137 is reserved for the NetBIOS name service and port 138 is reserved for the NetBIOS datagram service. The subsequent packet was assumed to contain the signature of the "BAT 911/Chode" worm even though it was a benign packet:

```
05/24-13:10:13.082716 152.78.70.46:137 -—> 152.78.70.127:137
UDP TTL:128 TOS:0x0 ID:47635 IpLen:20 DgmLen:78
Len: 58
.....................................................................
0x0030: 00 00 00 00 00 00 20 45 45 46 44 46 44 45 46 43 ..............EEFDFDEFC
0x0040: 41 43 41 43 41 43 41 43 41 43 41 43 41 43 41 43 ACACACACACACACAC
0x0050: 41 43 41 43 41 42 4C 00 00 20 00 01                ACACABL ..    ..
```

Except these Microsoft Server packets, all other packets we have captured had a considerably different front part from data packets containing virus code. This observation suggests that malicious and benign packets look sufficiently different to consider an anomaly-based virus detection approach possible, which could identify new, *unknown* viruses.

5 Potential Approaches for the Detection Engine

Intending to accomplish the detection engine sketched above, which will be capable of identifying new viruses and generating warnings correspondingly, we consider several approaches. Each approach obviously has its own advantages and disadvantages; we will therefore decide on which approaches to deploy in the detection engine only after thorough examination and experiments. The first approach to consider is the Bayesian Belief Network (BBN). A BBN is a special type of diagram, or graph, together with an associated set of probability tables. Nodes of the BBN represent entities and their attributes; the arcs describe the relationships between these entities. BBNs are beneficial to model uncertain events and arguments about them. Moreover, if Bayesian probability is applied to propagate, it produces useful probability estimates for uncertain outcomes.

The second approach is Bayesian estimation. As an aspect of incremental learning, learning the parameters of a model can be addressed by Bayesian estimation. This has the following advantages: incorporation of prior knowledge and constraints, incorporation of models of time evolution, choice of optimality criteria, and, for linear Gaussian models, the Bayesian estimator is optimal under almost any rational optimality criterion. However, it has also disadvantages: a prior distribution must be known, and closed-form solutions are often hard to find.

Thirdly, the data mining approach describes the discovery of useful summaries of data based on relations, patterns, and rules that exist in the data. Pattern extraction and discovery as well as feature extraction capabilities of data mining processes seem to offer interesting possibilities for our detection engine.

Finally, there is the Neural Network approach. As the anomaly of data packets cannot be reduced explicitly to matching exact patterns, we need a mechanism that is capable of distinguishing between "good" and "bad" patterns in data packets based on (incomplete) knowledge about the general structure of these packets. With the aim to detect "bad" patterns, we take much account of five aspects: pattern classification, competitive learning, unsupervised learning, good performance in noise and error, and flexible time delay. Possession of such features can be found in neural-network-based detection. We will closely investigate 5 neural network models to deal with the 5 aspects of pattern classification as mentioned above: Self-Organizing Maps (SOM), Hopfield Networks, Hamming Networks, Probabilistic Neural Networks (PNN), and Time-Delay Neural Networks (TDNN). A detailed experimental analysis of these different models in regard to their suitability for the detection engine is part of future work in this project.

6 Conclusion and Future Research

Currently we have produced a firewall layout using libpcap [8] and snort [12] modules. In order to capture packets from the datalink layer we have used libpcap, and to decode data packets we have modified parts of snort. Our goal is to

build a smart detection engine and integrate it into the prototype firewall we have built. This engine will be useful if its detection rate is higher than that of traditional IDSs' anomaly detection rate with an acceptably low rate of false positives. Please recall that the smart detection engine will not only aim at detecting anomalous network traffic as in classical IDSs, but also to detect unusual structures in data packets that suggest the presence of virus data.

With intent to accomplish this goal we are investigating the above-mentioned different classification techniques in detail to identify suitable candidates to implement and experiment with in our firewall prototype. Problems to tackle include the performance of the decision process (to deal with real-time network traffic) and optimisations to avoid the creation of too many false positives. The selection of suitable techniques will require sufficient experimental evidence of their applicability. Such experiments form the next step in the discussed project.

References

1. CERT Advisory CA-2001-23 : Continued Threat of the "Code Red" Worm. 2002. http://www.cert.org/advisories/CA-2001-23.html
2. James Cannady and James Mahaffey: The application of artificial neural networks to misuse detection: initial results. In: Proceedings of the 1st International Workshop on Recent Advances in Intrusion Detection (RAID 1998), 1998.
3. D. Dittrich: The "Tribe Flood Network" distributed denial of service attack tool. October 1999. http://staff.washington.edu/dittrich/misc/tfn.analysis.
4. D. Dittrich: The DoS Project's "trinoo" distributed denial of service attack tool. October 1999. http://staff.washington.edu/dittrich/misc/trinoo.analysis.
5. D. Dittrich: The "stacheldraht" distributed denial of service attack tool. December 1999. http://staff.washington.edu/dittrich/misc/stacheldraht.analysis.
6. Anup K. Ghosh and Aaron Schwartzbard: A study in using neural networks for anomaly and misuse detection. 1999. http://www.docshow.net/ids/usenix_sec99.zip
7. L-117: The Code Red Worm, U.S. Department of Energy. CIAC (Computer Incident Advisory Capability), July 19, 2001. http://www.ciac.org/ciac/bulletins/l-117.shtml
8. LIBPCAP: The Tcpdump Group. http://www.tcpdump.org
9. M. Swimmer: Review and Outlook of the Detection of Viruses using Intrusion Detection Systems. In: Proceedings of the 3rd International Workshop on Recent Advances in Intrusion Detection (RAID 2000).
10. Christoph Ludwig Schuba: On the Modeling, Design and Implementation of Firewall Technology. Purdue University, PhD thesis, 1997. ftp://ftp.cerias.purdue.edu/pub/papers/christoph-schuba/schuba_phddis.pdf
11. Susan C. Lee and David V. Heinbuch: Training a neural network-based intrusion detector to recognize novel attacks. In: IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, July, 2001, 31(4) pp.294-299.
12. SNORT: The Open Source Network Intrusion Detection System. http://www.snort.org
13. G. Tesauro, J.O. Kephart, and G.B. Sorkin: Neural Networks for Computer Virus Recognition. In: IEEE Expert, 11(4): 5-6, August 1996.
14. Ulrich Ultes-Nitsche and InSeon Yoo: An Integrated Network Security Approach - Pairing Detecting Malicious Patterns with Anomaly Detection. In: Proceedings of the 2nd ISSA 2002. To be Published.