e-Business and Telecommunication Networks

e-Business and Telecommunication Networks

edited by

João Ascenso

ISEL,
Lisbon, Portugal

Luminita Vasiu

University of Westminster,
London, UK

Carlos Belo

IST/IT,
Lisbon, Portugal

and

Mónica Saramago

INSTICC,
Setúbal, Portugal

Springer

A C.I.P. Catalogue record for this book is available from the Library of Congress.

ISBN-10 1-4020-4760-6 (HB)
ISBN-13 978-1-4020-4760-2 (HB)
ISBN-10 1-4020-4761-4 (e-book)
ISBN-13 978-1-4020-4761-9 (e-book)

Published by Springer,
P.O. Box 17, 3300 AA Dordrecht, The Netherlands.

www.springer.com

Printed on acid-free paper

Printed in the Netherlands

TABLE OF CONTENTS

PART 1 – GLOBAL COMMUNICATION INFORMATION SYSTEMS AND SERVICES

PART 2 – SECURITY AND RELIABILITY IN INFORMATION SYSTEMS AND NETWORKS

PREFACE

This book contains the best papers of the First International Conference on E-business and Telecommunication Networks (ICETE 2004), held in Setúbal (Portugal) and organized by INSTICC (*Institute for Systems and Technologies of Information, Communication and Control*) in collaboration with the School of Business of the Polytechnic Institute of Setúbal, who hosted the event.

This conference represents a major initiative to increase the technical exchanges among professionals, who work on the e-Business and Telecommunication Networks fields, and who are deploying new services and technologies into the lives of ordinary consumers. The major goal of this conference is to bring together researchers and developers from academia and industry working in areas related to e-business, with a special focus on Telecommunication Networks. This year, four simultaneous tracks were held, covering different aspects, including: "*Global Communication Information Systems and Services*", "*Security and Reliability in Information Systems and Networks*", "*Wireless Communication Systems and Networks*" and "*Multimedia Signal Processing*". The sections of this book reflect the conference tracks.

ICETE 2004 received 202 paper submissions from 43 different countries, from all continents. 110 papers were published and orally presented as full papers, i.e. completed work, and 44 papers were accepted for poster presentation. The full paper acceptance ratio confirms our confidence that ICETE 2004 has achieved a high quality standard that we will strive to keep and enhance in order to ensure the success of the next year ICETE edition.

Additionally, the ICETE conference included a number of invited talks, including keynote lectures and technical tutorials. These special presentations made by internationally recognized experts have definitely increased the overall quality of the Conference and provided a deeper understanding of the Telecommunication Networks field. Their contributions have been included in a special section of this book.

The program for this conference required the dedicated effort of many people. Firstly, we must thank the authors, whose research and development efforts are recorded here. Secondly, we thank the members of the program committee and the additional reviewers for their diligence and expert reviewing. Thirdly, we thank the invited speakers for their invaluable contribution and for taking the time to synthesise and prepare their talks. Finally, we thank the workshop chairs whose collaboration with ICETE was much appreciated.

João Ascenso
School of Technology of Setúbal, IPS, Setúbal, Portugal

Luminita Vasiu
Middlesex University, WITRC, London, U.K.

Joaquim Filipe
School of Technology of Setúbal and INSTICC, Setúbal, Portugal

Carlos Belo
Institute of Telecommunications and IST, Lisbon, Portugal

CONFERENCE COMMITTEE

Conference Chair
Joaquim Filipe, Escola Superior de Tecnologia de Setúbal, Portugal

Programme co-Chairs
Carlos Belo, Instituto de Telecomunicações, Portugal
Luminita Vasiu, Middlesex University, U.K.

Program Committee Chair
João Ascenso, Escola Superior de Tecnologia de Setúbal, Portugal

Secretariat
Mónica Saramago, INSTICC, Portugal

Programme Committee:
Acharya, A. (USA)
Ahmed, K. (THAILAND)
Al-Sharhan, S. (KUWAIT)
Ansari, N. (USA)
Asatani, K. (JAPAN)
Assuncão, P. (PORTUGAL)
Barn, B. (UK)
Bedford, A. (AUSTRALIA)
Bella, G. (ITALY)
Benzekri, A. (FRANCE)
Boavida, F. (PORTUGAL)
Bonyuet, D. (USA)
Boutaba, R. (CANADA)
Broadfoot, P. (UK)
Cappellini, V. (ITALY)
Cheng, T. (SINGAPORE)
Cheung, K. (CHINA)
Choras, R. (POLAND)
Clarke, R. (UK)
Cohen, R. (ISRAEL)
Comley, R. (UK)
Constantinides, T. (UK)
Correia, M. (PORTUGAL)
Correia, P. (PORTUGAL)
Devetsikiotis, M. (USA)
Elmirghani, J. (UK)
Fang, Y. (USA)
Faria, S. (PORTUGAL)
Figueiredo, M. (PORTUGAL)
Gaspary, L. (BRAZIL)
Georghiades, C. (USA)
Giannakis, G. (GREECE)
Goldszmidt, G. (USA)
Goulart, C. (BRAZIL)
Granai, L. (SWITZERLAND)
Granville, L. (BRAZIL)
Greaves, D. (UK)
Gritzalis, S. (GREECE)
Kang, C. (KOREA)
Hamdi, M. (CHINA)
Hanzo, L. (UK)
Harris, R. (AUSTRALIA)
Helal, S. (USA)
Hoang, N. (SINGAPORE)
Hong, D. (KOREA)
Hu, J. (AUSTRALIA)
Huston, G. (AUSTRALIA)
Isaías, P. (PORTUGAL)
Jagodic, M. (SLOVENIA)
Jahankhani, H. (UK)
Jain, A. (INDIA)
Jefferies, N. (UK)
Júnior, E. (BRAZIL)
Kahlil, I. (AUSTRALIA)

Karmouch, A. (CANADA)
Kihl, M. (SWEDEN)
Kollias, S. (GREECE)
Kos, M. (CROATIA)
Kunt, M. (SWITZERLAND)
Kuo, G. S. (TAIWAN)
Landfeldt, B. (AUSTRALIA)
Lee, M. (AUSTRIA)
Lewis, L. (USA)
Liu, K. (UK)
Lloyd-Smith, B. (AUSTRALIA)
Lorna, U. (UK)
Loureiro, A. (BRAZIL)
Lu, S. (USA)
Magedanz, T. (GERMANY)
Magli, E. (ITALY)
Mahmoud, Q. (CANADA)
Makki, K. (USA)
Malek, M. (USA)
Malumbres, M. (SPAIN)
Man, H. (USA)
Marshall, A. (UK)
Marshall, I. (UK)
Mascolo, S. (ITALY)
Matsuura, K. (JAPAN)
McGrath, S. (IRELAND)
Merabti, M. (UK)
Mirmehdi, M. (UK)
Morikawa, H. (JAPAN)
Navarro, A. (PORTUGAL)
Nordholm, S. (AUSTRALIA)
Obaidat, M. (USA)
Ohtsuki, T. (JAPAN)
Osadciw, L. (EUA)
Pach, A. (POLAND)
Perkis, A. (NORWAY)
Petrizzelli, M. (VENEZUELA)
Pigneur, Y. (SWITZERLAND)
Pinnes, E. (USA)
Pitsillides, A. (CYPRUS)
Plagemann, T. (NORWAY)
Podvalny, S. (RUSSIA)
Preston, D. (UK)
Queluz, P. (PORTUGAL)
Ramadass, S. (MALAYSIA)
Raychaudhuri, D. (USA)
Regazzoni, C. (ITALY)
Reichl, P. (AUSTRIA)
Reis, L. (PORTUGAL)
Rodrigues, A. (PORTUGAL)
Rosales, C. (MEXICO)
Roth, J. (GERMANY)
Roztocki, N. (USA)
Sanadidi, M. (USA)
Schulze, B. (BRAZIL)
Sericola, B. (FRANCE)
Skarbek, W. (POLAND)
Specialski, E. (BRAZIL)
Steinmetz, R. (GERMANY)
Suda, T. (USA)
Sun, L. (UK)
Sure, Y. (GERMANY)
Tarouco, L. (BRAZIL)
Tirri, H. (FINLAND)
Toh, C. K. (USA)
Ultes-Nitsche, U. (SWITZERLAND)
Valadas, R. (PORTUGAL)
Vidal, A. (SPAIN)
Waldron, J. (IRELAND)
Weghorn, H. (GERMANY)
Weigel, R. (GERMANY)
Wilde, E. (SWITZERLAND)
Wilson, S. (USA)
Wu, G. (USA)
Wu, W. X. (UK)
Yasinsac, A. (EUA)
Yeo, B. (SINGAPORE)
Yin, Q. (SINGAPORE)
Youn, H. (KOREA)
Yu, W. (USA)
Yuan, S. (TAIWAN)
Zhang, J. (USA)

Invited Speakers

Luminita Vasiu, Middlesex University, U.K.
Manu Malek, Institute if Technology, USA
Henry Tirri, Nokia Research Fellow/Nokia Research Center, Finland
Nirwan Ansari, New Jersey Institute of Technology, USA
Mohamed Atiquzzam, Uiversity of Oklahoma, USA

Invited Speakers

DATA MINING TECHNIQUES FOR SECURITY OF WEB SERVICES

Manu Malek and Fotios Harmantzis
Steven Institute of Technology, Castle Point on the Hudson, Hoboken, NJ 07030, USA
Email: {mmalek, fharmant}@stevens.edu

Keywords: Security services, Security attack, Denial of service, Intrusion detection, Security safeguards.

Abstract: The Internet, while being increasingly used to provide services efficiently, poses a unique set of security issues due to its openness and ubiquity. We highlight the importance of security in web services and describe how data mining techniques can offer help. The anatomy of a specific security attack is described. We then survey some security intrusions detection techniques based on data mining and point out their shortcomings. Then we provide some novel data mining techniques to detect such attacks, and describe some safeguard against these attacks.

1 INTRODUCTION

Cyberspace is used extensively for commerce. For years banks and other financial organizations have conducted transactions over the Internet using various geographically dispersed computer systems. Businesses that accept transactions via the Internet can gain a competitive edge by reaching a worldwide customer base at relatively low cost. But the Internet poses a unique set of security issues due to its openness and ubiquity. Indeed, security is recognized as a critical issue in Information Technology today. Customers will submit information via the Web/ Internet only if they are confident that their private information, such as credit card numbers, is secure. Therefore, today's Web/Internet-based services must include solutions that provide security as a primary component in their design and deployment.

Web services generally refer to web-based applications that make it possible for enterprises to do transactions on the web and for users to share documents and information with each other over the Web. The standard that makes it possible to describe the communications in some structured way is Web Services Definition Language (WSDL). WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information (http://www.w3.org/TR/wsdl).

But openness and integration have their price. Without adequate security protections and effective security management, these features can be used to attack the availability and integrity of information systems and the networks connecting to them. Here we highlight a few typical ways an attacker may gain illegal access to an information system, or to make it unavailable to legitimate users. We identify the profiles or signatures for the sequence of actions an attacker may perform to perpetrate such attacks. We use data mining techniques to discover such attack profiles to detect the attacks.

Data mining refers to a technique to intelligently and automatically assist humans in analyzing the large volumes of data to identify valid, novel, and potentially useful patterns in data. It offers great promise in helping organizations uncover patterns hidden in their data that can be used to predict the behavior of customers, so that they can better plan products and processes. Data mining takes advantage of advances in the fields of artificial intelligence (AI) and statistics. Both disciplines help in pattern recognition and classification. Other disciplines used in data mining include rule-based and case-based reasoning, fuzzy logic, and neural networks. The techniques used in data mining include rule induction, clustering, projection, and visualization (e.g., see (Berry, M. and L. Gordon, 1997) for details).

J. Ascenso et al. (eds.), e-Business and Telecommunication Networks, 3–10.

This paper provides a glimpse at the cyberspace security situation, and offers some techniques to manage the security of web services. The paper describes some security attacks, and provides some techniques to detect and defend against them. In Section 2, we present some statistics related to security attacks to highlight the urgency of the issue. Some typical security vulnerabilities and attacks are discussed in Section 3. In Section 4, we provide a survey of data mining applications in intrusion detection and point out their shortcomings. We then define attack signatures and outline how to use them in conjunction with data mining techniques for efficient intrusion detection. Section 5 summarizes the paper.

2 BACKGROUND

Based on data provided by CERT/CC, the number of incidents and vulnerabilities for cyber attacks have increased exponentially during the period 1998 to 2002 (CERT/CC). Figure 1 shows that intrusions were relatively few in the early1990s, but there has been a major increase since 2000. About 25,000 intrusions were reported in the Year 2000 (CERT/CC). Keep in mind that not all enterprises that suffer security breaches report them. The line moving upward in this figure shows various types of threats, starting with very simple ones in the early '90s, like password guessing. The sophistication of attacks increased with self-replicating codes, such as viruses, then password cracking (where the cryptographic password is broken), and on to the more sophisticated threats shown. Against this rising sophistication in threats, we have easy availability of hacking tools: hackers no longer have to be experts in computer science or security; they could use available tools. For example, a tool such as nmap (www.insecure.org/nmap/nmap-fingerprinting-aticle.html) can be used to find all the open ports, a first stage in an attack. This combination of decreasing knowledge required of the attackers and the increasing sophistication of the attacks is giving rise to major security concerns.

According to the Federal Computer Incidence Response Center (FedCIRC), the incident handling entity for the federal government, 130,000 government sites totaling more than one million hosts were attacked in 1998 (NIST ITL Bulletin, 1999). Also, a 1999 survey conducted by the Computer Security Institute (CSI) and the Federal Bureau of Investigation (FBI) revealed that 57% of organizations cited their Internet connections as a frequent point of attack, 30% detected actual network intrusion, and 26% reported theft of proprietary information (CSI, 2002). A similar survey in 2002, showed that 90% of the 507 participating organizations detected computer security breaches within the past 12 months, 74% cited their Internet connections as a frequent point of attack, but only 34% reported security intrusions to law enforcement agencies. These numbers must be considered observing that they relate to only known attacks and vulnerabilities. However, they do indicate the magnitude of the problem.

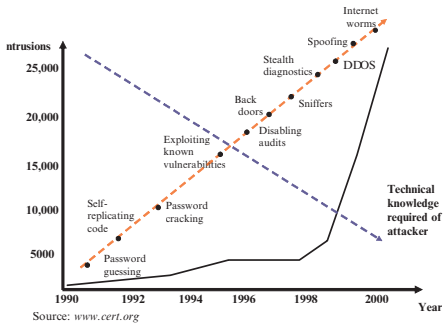

Figure 1: Security vulnerabilities and threats.

A key to preventing security attacks is to understand and identify vulnerabilities, and to take corrective action. A threat to computing systems or communication network is a potential violation of security unauthorized, illegitimate, malicious or fraudulent purposes. An attack is the implementation of a threat using the system vulnerabilities. Vulnerability is a weakness in the security system that might be exploited to launch an attack. Finally, a control is a protective measure – an action, device, procedure, or technique – that reduces vulnerabilities.

Table 1 sows the top 10 security vulnerabilities as reported periodically by The SANS Institute (The SANS Institute, 2003). The reasons for the existence of these vulnerabilities include: buggy software design and development, system administrators being too busy to install security patches in a timely manner, and inadequate policies and procedures. Another factor is that due to the ubiquity of the Internet, vulnerabilities are quickly and widely published.

Table 1: Top 10 security vulnerabilities (The SANS Institute, 2003).

Vulnerabilities of Windows Systems	***Vulnerabilities of Unix Systems***
1. Internet Information Services (IIS)	*1. Remote Procedure Calls (RPC)*
2. Microsoft Data Access Components (MDAC)– Remote Data Services	*2. Apache Web Server*
3. Microsoft SQL Server	*3. Secure Shell (SSH)*
4. NETBIOS – Unprotected Windows Networking Shares	*4. Simple Network Management Protocol (SNMP)*
5. Anonymous Logon – Null Sessions	*5. File Transfer Protocol (FTP)*
6. LAN Manager Authentication – Weak LM Hashing	*6. R-Services – Trust Relationships*
7. General Windows Authentication – Accounts with No Passwords or Weak Passwords	*7. Line Printer Daemon (LPD)*
8. Internet Explorer	*8. Sendmail*
9. Remote Registry Access	*9. BIND/DNS*
10. Windows Scripting Host	*10. General Unix Authentication – Accounts with No Passwords or Weak Passwords*

The motive of attackers could be anything from pure joy of hacking to financial benefit. The attackers are either highly technically capable, or they sometimes breaks into the network by trial and error. Disgruntled employees have more access rights to enterprise computer networks compared to outside attackers. According to the CSI/FBI 2002 Survey (CSI, 2002), 60% of attacks in the US were inside attacks (attacks that originated inside the institutions) and 40% were outside attacks.

3 SOME TYPICAL SECURITY ATTACKS

As mentioned, a security attack occurs when an attacker takes advantage of one or more security vulnerabilities. To improve security, one needs to minimize security vulnerabilities. In this section we present some typical security attacks, point out the vulnerabilities abused to perpetrate the attacks. Some safeguards against these attacks will be described in the next section. An attack that we deal with specifically in this section and in Section 4 is the HTTP GET attack.

3.1 Denial-of-Service Attack

In a Denial-of-Service (DoS) attack, the attacker attempts to use up all the victim system's resources like memory or bandwidth. When the attack is successful, legitimate users can no longer access the resources and the services offered by the server will be shut down. According to the 2002 CSI/FBI survey (CSI, 2002), 40% of all attacks are DoS attacks.

An attack can be directed at an operating system or at the network. The attacker may send specially crafted packets that crash remote software/services running on the victim server. It will be successful if the network is unable to distinguish between legitimate traffic and malicious or bogus traffic. Some common DoS attacks follow.

ICMP Flooding and Smurf Attack
These are both ICMP-based attacks. Flooding with ICMP packets slows down the victim server so that it can no longer respond quickly enough for the services to work properly. If packets are sent with forged IP addresses, the victim server not only has to allocate system resources to receive, but to reply to packets to addresses which do not exist. The Smurf attack uses a similar idea: the attacking machine sends Echo requests with broadcast IP addresses, thus not only the victim server but the attached network will be flooded by a large amount of ICMP traffic.

SYN Flooding
SYN flooding exploits the weakness of the TCP Three-way Handshake (Comer, D., 2000). In a normal TCP connection request, the source sends a SYN (synchronization) packet to the destination to initiate the connection; then waits for a SYN ACK (synchronization acknowledged) packet from the destination. The connection is established when the destination receives a FIN ACK (finishing acknowledged) packet from the source. In the SYN flooding attack, the attacker sends a large number of SYN packets, often from bogus IP addresses, to the victim server, which adds the entry to the connection queue and replies with SYN ACKs. As the source addresses are incorrect or non-existent, FIN ACKs

will never be received by the victim server, so the last part of the Three-way Handshake never completes and the connection queue of the victim server fills up.

Badly-formed Packets
In this type of attack, the attacker sends badly-formed IP packets, e.g., packets that consist of invalid fragments, protocol, packet size, or header values, to the victim server. Once the destination TCP stack receives such invalid packets, the operating system must allocate resources to handle them. If the operating system cannot handle the misbehavior, it will crash. An example of this is the Ping-of-Death attack (www.insecure.org/sploits/ping-o-death.html) which causes buffer overflow in the operating system. In this attack, the attacker sends a larger than standard ICMP (Internet Control Message Protocol) packet, such as a ping, in fragments to the target server. Since the allowed maximum size of such a packet is 65,535 bytes, the server allows a corresponding buffer space to collect the fragments. A clever attacker may create a ping with many fragments destined to a target server. The server receives the fragments and starts to reassemble them. When reassembled, the buffer will overflow, leading to program termination, overwriting other data or executable code, kernel dump, etc. More than 50% of attacks on servers are due to buffer overflow (CERT/CC).

Distributed Denial of Service Attack
With the speed and power of computing resources today, an attacker may not be able to simply use one computer to craft a DoS attack. In the Distributed Denial of Service (DDoS) attack, many computers may be hijacked by the attacker as agents (zombies) to simultaneously flood a victim system's resources. A typical way to recruit zombie computers is for the perpetrator to send viruses to multiple computers, or to break into computer systems and load them with DDoS programs. Each infected system then finds other vulnerable systems and loads them with the programs, etc. The perpetrator uses the first system that was overtaken to instruct all the other compromised systems to launch the attack simultaneously.

3.2 HTTP GET Attack

For many web applications, a client should be able to send information to the server. HTML 2.0 and later versions support the Form element within an HTML document to allow data to be sent to web servers (www.w3c.org). One of the attributes of Form is Method which indicates how data is submitted to the web server. Valid choices for the Method attribute are GET and POST. In METHOD = GET the values inputted by the user are concatenated with the URL, separated by a special character (usually ?) fields are separated by &; space is represented by + For example, the following URL: http://www.gadgets, com?customer = **John + Doe** & address = **101 + Main + Street** & cardno = **1234567890** & cc=; **visacard** indicates that the customer's name and address with the customer's credit card number are to be sent to the web server at www.gadgets.com A savvy user (attacker) may be able to use this feature to get access to proprietary information if appropriate security mechanisms are not in place. The following scenario, adopted from Ref. (McClure, S. et al., 2003), is an HTTP GET attack on a typical web server which has some vulnerabilities.

The server http://www.acme.com runs Apache 1.3.12 on a Linux operating system. Firewalls prevent all but HTTP traffic via ports 80 (HTTP default port) and 443 (SSL port). Perl CGI scripts are used for the online store. A visitor to this site first begins browsing through the www.acme.com site, viewing the site's main page and a few images on it. The visitor notices that for the last selection (viewing the picture of a sunset), the URL in the browser window shows: http://www.acme.com/index.cgi?page=sunset.html. Following this pattern, the visitor (now attacker) issues a request for index.cg by typing the following URL: http://www.acme.com/index.cgi?page=index.cgi

Now, if the program does not validate the parameters passed to the index.cgi script, the filename passed as a parameter from the URL is captured by the CGI script, appended to the absolute path, and causes to open the index.cgi script as requested. Consequently, the browser display shows the source code of the index.cgi script!

At this point the attacker realizes that this technique can be further exploited to retrieve arbitrary files from the server. So the attacker may send the following request through the browser: http://www.acme.com/index.cgi?page=/../../etc/passwd

If permissions are not set properly on the /etc/passwd file, its contents will be displayed by the browser, providing the attacker with user's password information. The attacker could now execute arbitrary commands on the server, for example, by sending http://www.acme.com/index.cgi?page=|ls+-la/%0aid%0awhich+xterm| (% plus the hex character

0a indicates line feed) requesting ls –al (to show a file list of the server's root directory) id (the effective user id of the process running index.cgi) which xterm (path to the xterm binary code, to gain interactive shell access to the server and the attacker could gain full interactive shell-level access to the web server.

Note again that the vulnerabilities: the program does not validate the parameters passed to the index.cgi script, and permissions are not set properly on the /etc/passwd file.

4 DATA MINING TECHNIQUES FOR INTRUSION DETECTION

In this section we review server logs, introduce attack signatures, and present our main contribution: how security attack signatures are used in conjunction with data mining to detect security intrusions. More specifically, we first describe the relevance and importance of the different log files that are available; we then define specific patterns in the log files for an attack (the individual log records as well as their sequence/order) as the attack signature; and use data mining to search and find such patters for attack detection. The efficiency and speed of the overall process can even lead to attack prediction capabilities.

4.1 Logs

Every visit to a Web site by a user creates a record of what happens during that session in the server's log. A busy site may generate thousands of log entries per hour, compiled in various log files. A log file entry contains items like the IP address of the computer requesting the Web page, the date and time of the request, the name and the size of the file requested. Logs vary by the type of server and the file format. Following are some typical logs and what they record:

- Access Log records every transaction between server and browsers (date, time, domain name or IP address, size of transaction, …).
- Referrer Log records the visitor's path to the site (the initial URL from which the visitor came).
- Agent Log records the type and version of the browser.

For secure systems, the standard logs and directories may not be sufficient and one must employ additional logging tools, e.g., information about which computer is connecting to which services on the system. There are many programs under the heading of IP loggers available for this purpose, e.g., EnviroMon (http://www.interwld.com/pico/subs/pico_Environ_IP_Logging.htm) and ippl http://packages.debian.org/unstable/net/ippl.html).

4.2 Mining Logs

The data available in log files can be "mined" to gain useful information. Data mining offers promise in uncovering hidden patterns in the data that can be used to predict the behavior of (malicious) users. Using data mining in intrusion detection is a relatively new concept. In (Lee, W. and Stolfo, S. J., 1998), the authors outline a data mining framework for constructing intrusion detection models. The central idea is to utilize auditing programs to extract an extensive set of features that describe each network connection or host session, and apply data mining to learn rules that capture the behavior of intrusions and normal activities. Detection models for new intrusions are incorporated into an Intrusion Detection Systems (IDS) through a meta-learning (or co-operative) learning process. The strength of this approach is in classification, meta-learning, and association rules.

In (Almgren, M. et al., 2000), the authors present an intrusion detection tool aimed at protecting servers. However, their method does not effectively handle all matches of the signature (e.g., of the 404 type: document not found). Attacks that have no matching signature and are sent by a previously unknown host may be missed.

Ref. (Forrest, S. et al., 1996 represents a first attempt to analyze sequences of system calls issued by a process for intrusion detection. The authors introduce a method based on sequences of Unix system calls at the process level for anomaly detection resulting in intrusion detection. They address sendmail, lpr, and ftpd processes and obtain some good results in terms of false-positives. Ref. (Hofmeyr, S.A., 1998) also uses sequences of system calls for intrusion detection. The authors choose to monitor behaviour at the level of privileged processes. Their proposed approach of detecting irregularities in the behavior of privileged programs is to regard the program as a black-box, which, when it runs, emits some observable behavior. Privileged processes

are trusted to access only relevant system resources, but in cases where there is a programming error in the code that the privileged process is running, or if the privileged process is incorrectly configured, an ordinary user may be able to gain super-user privileges by exploiting the problem in the process. The system-call method, however, is specific to processes and cannot detect generic intrusion attempts, e.g., race condition attacks, session hijackin g(when one user masquerades as another), and cases in which a user violates policy without using privileged processes.

Artificial intelligence techniques have also been applied to help in decision making for intrusion detection. In (Frank, J., 1994), the author presents a survey of such methods and provides an example of using feature selection to improve the classification of network connections. In (Liu, Z. et al., 2002), the authors present a comparison of some neural-network-based method and offer some "classifiers" for anomaly detection in Unix processes. All the techniques based on artificial intelligence, however, suffer from lack of scalability: they work only for small size networks and data sizes.

4.3 Attack Signatures

We use attack signatures in combination with data mining to not only detect, but predict attacks. An attack signature encapsulates the way an attacker would navigate through the resources and the actions the attacker would take. For example, in a denial-of-service attack, the attacker may send a large number of almost simultaneous TCP connect requests from one or more IP addresses without responding to server acknowledgements.

To illustrate a specific attack signature, let us look at the log lines stored by the web server in the HTTP GET attack example described in the previous section. The log line in the Access Log corresponding to the visitor's (attacker's) first attempt is

A. 10.0.1.21 – [31/Oct/2001:03:02:47] "GET/HTTP /1.0" 200 3008 where 10.0.1.21 is the visitor's IP address, followed by date and time of visit, the Method and the Protocol used. The number 200 indicates the "normal" code, and 3008 indicates the byte size of the file retrieved. The following log line corresponds to the visitor's selection of the sunset picture:

B. 10.0.1.21 – [31/Oct/2001:03:03:18] "GET/sunset. jpg HTTP/1.0" 200 36580 and the following log line corresponds to the visitor's first attempt at surveillance of the site (issuing a request for index.cgi):

C. 10.0.1.21 – [31/Oct/2001:03:05:31] "GET/index. cgi?page= index.cgi HTTP/1.0" 200 358 The following log lines correspond to the visitor (by now, attacker) attempting to open supposedly secure files:

D. 10.0.1.21 – [31/Oct/2001:03:06:21] "GET/index. cgi?page=/../../etc/passwd HTTP/1.0" 200 723

E. 10.0.1.21 – [31/Oct/2001:03:07:01] "GET/ index. cgi?page=|ls+-la+/%0aid%0awhich+xterm|HTTP/ 1.0" 200 1228

This pattern of log lines from the same source IP address can be recognized as a signature of an HTTP GET attack. In the above example, the sequence of log lines A-B-C-D-E, A-C-D-E, B-C-D-E, or C-D-E constitutes the signature of this HTTP GET attack. Even some individual log lines from a source IP address could provide tell-tale signs of an impending HTTP GET attack. For example, the existence of a "pipe" (i.e.,) in the URL, as in log line E above, would indicate that the user is possibly trying to execute operating system commands.

In our research, we try to establish signatures for various types of attacks. Note the importance of good comprehensive attack signatures in detecting attacks. Incomplete signatures result in false-positive or false-negative detection. Another point is the use of data mining to detect attack signatures. One can imagine the tremendous amount of data collected by web services, resulting in multi-tera-byte databases. With such large amounts of data to analyze, data mining could become quite computationally expensive. Therefore, efficiency becomes a major issue. Currently, we are continuing our efforts to identify ways data should be efficiently analyzed in order to provide accurate and effective results.

In our research, we use the Rule Induction Kit (RIK) and Enterprise Data-Miner (EDM) tools (http://www.data-miner.com) to detect and mine attack signatures. The RIK package discovers highly compact decision rules from data, while the EDM software kits implement the data-mining techniques presented in (Weiss, S. and Indurkhya, N., 1997) and includes programs for (a) data preparation (b) data reduction or sampling, and (c) prediction. Our selection of this tool package was based on criteria related to efficiency (speed, especially when it comes to large amounts of data, as is the case with log files), and portability (multiple platforms), as well as extensibility (where the user can compose new methods with the existing building blocks).

Based on this software platform, we are able to create a sophisticated data mining methodology for efficient intrusion detection.

4.4 Security Safeguards

Safeguards are applied to reduce security risk to an acceptable/desirable level. They may be Proactive to prevent security incidents, or Reactive, to protect information when an incidence is detected. In either case, they must be cost effective, difficult to bypass, and with minimal impact on operations. Examples of safeguards are: avoidance (keeping security incidents from occurring, e.g., by removing vulnerabilities), limiting access (e.g., by reducing the number of entry points where attacks may originate), transference (shifting risk to someone else, e.g., via insurance or outsourcing), and mitigation (minimizing the impact of an incidence, e.g., by reducing its scope or improving detection).

One of the major safeguards is to detect and reduce/remove vulnerabilities. The main reasons for existing vulnerabilities are buggy software design and development, or system administration problems. Existence of bugs in software are due to

- programming for security not being generally taught,
- good software engineering processes not being universal, as well as
- existence of legacy code.

The system administration problems are due to inadequate policies and procedures, or the system administrators being too busy with many machines to administer, too many platforms and applications to support, and too many updates and patches to apply.

For the attack examples given in the previous section, we can offer some rather simple safeguards. For attacks that are based on making multiple requests and ignoring the server acknowledgments, such as ICMP Flood and Smurf Attack, and SYN Flooding, one could employ a timer: if the response does not arrive within a reasonable time, the request could be dropped and the resources freed. For attacks that are based on buffer overflow, one could use operating systems written in "safe" languages that perform range checking (like Java). The HTTP GET attack could be prevented by making sure that programs validate the parameters passed to them, and that file permissions are set properly.

5 CONCLUSIONS

We have first set the stage emphasizing the magnitude of the security problem, raising awareness and focusing on the impact of security. We have detailed two attacks: Denial of Service and the HTTP GET attack, and defined the signature of the latter. The application of data mining techniques for detecting attacks was described. The novelty of our approach is in determining the relevance/importance of different log records, defining intelligent signatures, and using efficient data mining techniques. Preliminary results have been encouraging.

There is significant work still to be done, e.g., improving the effectiveness of attack signatures, developing distributed algorithms for detection/prediction, and improving the efficiency of pattern searching. We are currently working on these issues.

REFERENCES

http://www.w3.org/TR/wsdl

Michael J. A. Berry and Gordon Linoff, Data Mining Techniques, Wiley Computer Publishing, 1997

Computer Emergency Response Team/Coordination Center (CERT/CC) at Carnegie Mellon University's Software Engineering Institute, http://www.cert.org/

www.insecure.org/nmap/nmap-fingerprinting-article.html

NIST ITL Bulletin, "Computer attacks: what they are and how to defend against them," May 1999.

CSI, "2002 CSI/FBI Computer Crime and Security Survey," http://www.gocsi.com/.

The SANS Institute (http://www.sans.org/top20/), May 2003

Douglas Comer, Internetworking with TCP/IP Vol. 1: Principles, Protocols, and Architecture (4th Edition), Prentice Hall, 2000

www.insecure.org/sploits/ping-o-death.html

www.w3c.org?

S. McClure, S. Shah, and S. Shah, Web Hacking: Attacks and Defenses, Addison Wesley, 2003

http://www.interwld.com/pico/subs/pico_Environ_IP_Logging.htm

http://packages.debian.org/unstable/net/ippl.html

W. Lee and S. J. Stolfo, "Data Mining Approaches for Intrusion Detection," Usenix Security Symposium, San Antonio, Texas, July 1998

Jeremy Frank, "Artificial Intelligence and Intrusion Detection: Current and Future Directions," June 1994 (http://citeseer.nj.nec.com/frank94artificial.html)

Zhen Liu, German Florez, and Susan Bridges, "A Comparison of Input Representation in Neural Networks: A Case Study in Intrusion Detection," Proc. International Joint Conference on Neural Networks, May 12-17, 2002, Honolulu, Hawaii. http://www.data-miner.com

S. Weiss and N. Indurkhya, Predictive Data Mining: A Practical Guide, Morgan Kaufmann, 1997.

Magnus Almgren, Herve Deba, and Marc Dacier, "A Lightweight Tool for Detecting Web Server Attacks," http://www.ce.chalmers.se/~almgren/Publications/almgren-ndss00.pdf

S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A Sense of Self for Unix Processes," Proc. 1996 IEEE Symp. Security and Privacy, Los Alamitos, CA, pp. 120-128, 1996

S. A. Hofmeyr, A. Somayaji, and S. Forrest, "Intrusion Detection using Sequences of System Calls," Journal of Computer Security Vol. 6, pp. 151-180, 1998.

TOWARDS AN ALTERNATIVE WAY OF VERIFYING PROXY OBJECTS IN JINI

Nikolaos Papamichail and Luminita Vasiu
School of Computer Science, Middlesex University, London, UK
Email: n.papamichail@mdx.ac.uk, l.vasiu@mdx.ac.uk

Keywords: Jini Security, Proxy Trust Verification.

Abstract: Jini networking technology represents an exciting paradigm in distributed systems. Its elegant approach in computer networking possesses immense advantages, but also generates security problems. Extensive research has been undertaken and existing security methodologies have been applied to provide a safe execution environment. However the unique nature of Jini has made it hard for traditional security mechanisms to be applied effectively. Part of the problem lies within the downloaded code and in the lack of centralised control. Current solutions are based on assumptions; therefore they are inadequate for enforcing the security requirements of the system. The goal of our research is to increase the security of the Jini model without altering its initial characteristics. We present our preliminary research efforts in providing an alternative, fault tolerant security architecture that uses a trusted local verifier in order to evaluate and certify the correctness of remote calls.

1 INTRODUCTION

Jini networking technology (Sun Microsystems Inc.2003a; http://www.jini.org/) presents an exciting paradigm in distributed computing. Based on the Java programming language, it allows the development of spontaneous networked systems. Users and applications are able to dynamically locate one another and form on-the-fly communities. Unlike traditional systems that rely on a fixed protocol and central administration, Jini requires no further human intervention once being set up. It employs strong fault-tolerance mechanisms that do not attempt to eliminate or hide the fact that network failures may happen. On the contrary it provides a programming model and an infrastructure that allow developers to recognise and isolate any faults that might occur.

When Jini was made publicly available, no security has been taken into consideration. The Java language alone was not adequate to cope with the security required in a distributed setting. Although some solutions have been proposed, Jini lacked a generic security model that could be applied to counter any threats that might arise. The Davis project (http://davis.jini.org/) presents such a security model that has been recently incorporated into the latest Jini release. The security model is based on well known and proven techniques to enforce the basic requirements for network security. However, some of the mechanisms that Jini employs are unique in distributed computing. Additionally, neither any real world applications that make use of the model nor a formal evaluation of it have appeared yet. Thus any assumptions about the correctness of the design and the degree of security provided might prove to be mistaken. The purpose of our research is to examine the security model employed by Jini technology for any potential security faults and propose appropriate modifications. In this paper we focus in the algorithm responsible for verifying trust in Jini proxy objects.

The rest of the paper is organised as follows. Section 2 presents an overview of the Jini programming model and infrastructure, particularly the components that constitute a Jini system and other mechanisms relevant to Jini operation. Section 3 presents some security problems related to proxy objects, Lookup Services and Jini Services while Section 4 presents an overview of the current Jini security model, the Davis Project, and a critical approach to its proxy verification algorithm. Section 5 presents an outline of two proposed solutions to the issues related with proxy object verification and the advantages that they may possess. Section 6 presents related work and some concluding thoughts are drawn in Section 7.

J. Ascenso et al. (eds.), e-Business and Telecommunication Networks, 11–18.

2 BACKGROUND

Jini (Sun Microsystems Inc., 2003a; http://www.jini.org/) is a distributed system based in Java that allows the establishment of spontaneous network communities or federations. To make that possible, Jini provides the following:

An infrastructure that enables devices, human users and applications to dynamically discover one another without any prior knowledge of their location or of the network's topology and form dynamic distributed systems. The infrastructure is composed of a set of components based on Jini's programming model. Parts of the infrastructure are the discovery join and lookup protocols and the Lookup Service (Sun Microsystems Inc., 2003a). A programming model that is used by the infrastructure as well as by services. Besides service construction, the programming model provides interfaces for performing leasing as well as event and transaction handling.

Services that are employed inside a federation and provide some functionality. Services exploit the underlying infrastructure and are implemented using the programming model.

2.1 Services

Every entity that participates in a Jini system and provides some functionality is perceived as a service. No separation is made regarding the type or the characteristics of the service. A service could be either a hardware device, a piece of software or a human user. Jini provides the means for services to form interconnected systems, and each one separately to offer its resources to interested parties or clients. The separation between a service and a client, however, is sometimes blurred, as sometimes a Jini service may act both as a service and a client.

A word process application, for example, is perceived as a service by any human user that writes a document, although the same application acts as a client whenever it uses a device such as a printer. The latter is again a Jini service, thus for the infrastructure the word application is now its client.

2.2 Proxy Objects

In order for services to participate in a Jini system they must create an object that provides the code by which they can be exploited by potential clients, the proxy object. The proxy object contains the knowledge of the service's location and the protocol that the service implements. It also exposes an interface that defines the functions that can be invoked. A client is able to make use of a service only after the correspondent service's proxy object is downloaded to the client's local space. By invoking functions defined in the proxy interface, clients are able to contact and control services. Clients need only to be aware of the interface that the proxy implements and not of any details of the proxy implementation.

2.3 Lookup Service

The Lookup Service (LUS) is a special kind of service that is part of the Jini infrastructure. It provides a mechanism for services to participate in a Jini system and for clients to find and employ these services. The Lookup Service may be perceived as a directory that lists all the available services at any given time inside a Jini community. Rather than listing String based entries that point back to the location of a service, the Lookup Service stores proxy objects registered by Jini services.

2.4 Discovery Join and Lookup

Relevant to the use of Lookup services are three protocols called discovery, join and lookup (Sun Microsystems Inc. 2003a). Discovery is the process where an entity, whether it would be a service or a client, is trying to obtain references to a lookup service. After a reference has been successfully obtained, the entity might register a proxy object with the Lookup service (join), or search the Lookup Service for a specific type of service (lookup). The discovery protocol provides the way for clients and services to find available Lookup Services in the network, and for Lookup Services to announce their presence.

3 JINI SECURITY ISSUES

Typically security is concerned with ensuring the properties of confidentiality, integrity, authentication and non-repudiation (Menezes et al., 1996):

- Confidentiality ensures that information remains unseen by unauthorised entities
- Integrity addresses the unauthorised alteration of data
- Authentication is the verification of identity of entities and data
- Non-repudiation prevents an entity from denying previous commitments or actions

These properties are generic and apply to a wide variety of systems. Inside Jini, no prior knowledge

of the network's infrastructure is assumed. For that reason, Jini is not only bound to security problems related to distributed systems, but also to any additional issues that the spontaneity of the environment invokes. The following components present different security requirements and they will be examined separately.

3.1 Proxy Object Issues

Nothing should be able to alter the state of the proxy object, either by intention or by fault. That means that the integrity of the proxy object must be ensured (Hasselmeyer et al., 2000a). Since the proxy object is downloaded from an unknown location in the network, neither the source nor the intentions of the proxy object can be verified. Therefore, even the act of downloading the proxy of a service is considered by itself a security risk. Moreover, the proxy is responsible for performing the communication between the client, and the service that the proxy represents. Therefore the integrity and confidentiality of the communication has to be preserved, since the communication link might be intercepted, altered, or simulated by someone with malicious intentions. The privacy and anonymity of the client may be abused, because the client can not be ensured that the proxy does indeed provide the functionality it claims (Hasselmeyer et al., 2000a). On the other hand it has to be verified that any data that needs to be supplied to the proxy object, for the interaction with the service to take place, reaches the appropriate service (JAAS).

3.2 Lookup Service Issues

The Lookup Service lacks any mechanism for authenticating services (Schoch et al., 2001). That means every service can discover the Lookup Service and register its proxy. Malicious proxies may register and pretend they provide some functionality, while they don't. Moreover, every client can search the Lookup Service and find which services are provided. Some services may require only registered users to access them. Therefore access control mechanisms need to be imposed. Additionally, clients might encounter unfairness while searching the Lookup Service for available services (Hasselmeyer et al., 2000a). There is no way a client of a service can be assured that he received the best available service from the Lookup Service. The fact that every service can register and even re-register with the Lookup Service can lead to "man-in-the-middle" attacks (Schoch et al., 2001). A malicious service just has to re-register its proxy with the same service ID as the original one. Every time a client tries to access the required service, the Lookup Service may provide him with the new, malicious proxy. The client is unaware of the change, as the new proxy looks like it implements the same interface as the original one.

3.3 Service Interaction issues

In order for an interaction between two services to take place, the service acting as a client must first locate the provider of the desired service, via the process of discovery, and then download its corresponding proxy object. However, in a spontaneous environment like Jini, hundreds of services may be present at the same time and many of them may provide the same functionality. No standard names or address for recognising individual services exist, besides a unique service ID that is assigned by the Lookup Service. However, it is dependent upon the provider of each service to decide whether or not the assigned ID will be stored and used in any future transactions. Therefore clients have to be able to authenticate the services they access (Eronen et al., 2000). Similarly, the service provider has to be able to authenticate clients that try to use its resources and call its provided functions.

Another aspect in the service interaction is different access levels (Kagal et al., 2001). An obvious solution to this problem is the integration of access control lists. Every user could be identified by a unique username and password that would grant him or deny certain permissions. However, new problems arise, like the distribution of the appropriate keys and the way that the permissions are to be decided.

4 THE DAVIS PROJECT

The Davis project (http://davis.jini.org/) is an effort led by Sun Microsystems' project team responsible for the development of Jini. The purpose is to satisfy the basic Jini requirements for security, by providing a security programming model that would be tightly integrated with the original Jini programming model and infrastructure. Part of the requirements (Scheifler, 2002) has been to avoid changing any existing application code by defining security measures at deployment time. Also to extend the security mechanisms provided by the Java programming language, such as the Java Authentication and Authorisation Service (JAAS).

The Davis project has been integrated with the original release of Jini networking technology (Jini specifications archive – v 2.0) resulting in the

release of the Jini starter kit version 2 (Sun Microsystems Inc., 2003a).

4.1 Constraints

In order to support a broad variety of applications and requirements, the security model dictates that both service providers and their clients should specify the type of security they require before any interaction between them takes place. Decisions upon the type of the desirable security are expressed by a set of constraints that have the form of Java objects. Any service that wishes to incorporate security in its current implementation has to implement a proxy object that implements a well-known interface (Sun Microsystems Inc., 2003b). The interface defines a method for clients and services to set constraints to the proxy object. If the proxy implements that interface, all the imposed constraints apply to every single call through any method defined by the proxy. The basic constraints are the equivalent of Boolean constants that allow decisions upon the type of security required to be specified in proxy objects. Typically service providers specify the constraints during the proxy creation, while clients set the constraints after the proxy object has been downloaded. Constraints specify only what type of security is expected but not how this is implemented.

The security model dictates that constraints imposed by services and clients are combined to a single set of constraints. If any of them contradict with each other then no calls are performed. It is possible, however, that alternative constraints are defined. This provides an elegant way for all parties participating in a Jini interaction to have direct control over the security imposed.

4.2 Object Integrity

There are two mechanisms that the current security model employs to provide integrity for the code of proxy objects. Both assume that the http protocol is used. The first mechanism is http over SSL (https) (Rescorla, 2000), the standard protocol for providing web site security in terms of server authentication, confidentiality and integrity. The other is a custom defined protocol called HTTPMD (Scheifler, 2002; Sun Microsystems Inc., 2003b) The proxy object consists of code which is downloaded by clients, and data which is downloaded from the service. Therefore to ensure total integrity these mechanisms have to apply to both the location where the proxy object is downloaded from and the location of the object's codebase. Along with integrity, the https protocol provides confidentiality and encryption, resulting in additional overhead when these are not required. In these cases the HTTPMD protocol is used. The location of objects, including their code, is specified by a normal http URL. Attached to it is a cryptographic checksum of the contents of the code, a message digest (Rivest, 1992). By computing the checksum of the downloaded data and code and comparing it with the attached message digest, clients are ensured that integrity has been preserved, since any modification in the contents would result in a different message digest.

4.3 Proxy Trust Algorithm

In terms of deciding whether a client trusts a proxy object downloaded by an unknown source, the current model (http://davis.jini.org/) employs the procedure described below. It is assumed that the client has already downloaded a proxy object from somewhere but it can not yet trust neither the proxy object not its correspondent service. Initially the client performs an object graph analysis of the proxy object. By checking recursively all the classes that the object is composed of it can be determined whether these classes are local or not. If the classes are local, in perspective to the client, then the proxy object is considered trusted. This is accepted on the basis that all local code is considered trustworthy. In the case where the proxy object is not fully constructed of local classes, the following components take part in the proxy trust verification algorithm:

1. Proxy object
This is the object that implements the server's functionality. It is downloaded by the client, traditionally from the Lookup Service and it contains the knowledge of how to communicate back with the server. All remote calls to the server are passing through this object and this is the object that needs to be verified.

2. A 'bootstrap' proxy
If the object graph analysis proves that the classes used for the construction of the proxy object are not local relatively to the client, the client uses the initial proxy object to request another object called the 'bootstrap' proxy. The bootstrap proxy should be only consisted of local classes (relevant to the client). The purpose is that clients can trust an object that only uses local code to run. The 'bootstrap' proxy is also used to authenticate the server to the client, as well as to provide him with the verifier described next.

3. A proxy Verifier
The Verifier is an object sent to the client by the

server, using the 'bootstrap' proxy. It checks the downloaded proxy object in order to verify whether the server trusts the initial proxy object or not.

A client obtains a proxy object from the network using Jini discovery and lookup mechanisms. The client examines whether the proxy object is using local code (relative to the client). Since this is normally not the case the client has to verify whether the proxy object can be trusted. The way that this is performed by the current security model is illustrated in Figure 1.

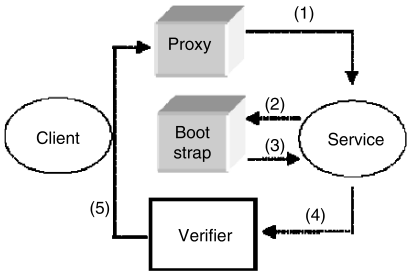

Figure 1: The proxy trust authentication employed by the current security model.

In order to verify that the proxy originates from a legitimate service, the client has to contact the same service and ask the service whether the proxy should be trusted. Since there is no way to directly contact the service, the client places a call through the proxy it can not yet trust, asking for a 'bootstrap' proxy (1). The bootstrap proxy has to use only local classes, relative to the client, in order to be considered trustworthy. After the bootstrap proxy is downloaded to the client's local address space (2), and the locality of the classes that compose the bootstrap object is verified, the client performs a call through it (3). Part of the call is to request from the service to authenticate. After the service has authenticated successfully, it passes a verifier object to the client (4). Finally the verifier is used to verify the legitimacy of the initial proxy object (5).

4.4 Critical Review of the Proxy Trust Algorithm

A number of potential problems might arise from the verification algorithm described above. The first is that clients have to rely on an object downloaded from an unknown source (the proxy object) to obtain the bootstrap proxy. In order for the latter to occur, clients have to place a remote call through an untrusted object. Since the functionality that the proxy object implements is unknown, clients may unintentionally execute an operation that presents a security risk in case the proxy is a malicious object. The second problem is that the service provider has to have some knowledge of the type of classes that are local to the user. If the bootstrap proxy is not consisted entirely by local classes, relevant to the client, the client would not utilize it to obtain the verifier.

A third type of problem is related to the way and type of checks that the verifier performs to the proxy object. There is no standardised set of tests that could be performed, since these are left for the service providers to implement. The method suggested is that the verifier carries the code of the proxy object. By checking the equality of the code that the verifier carries with the code of the proxy object, it is possible for a service to identify the correctness of the proxy object. However, there is no way to ensure whether the checks performed are adequate or if any checks are performed at all.

Therefore a 'lazy' verifier that just confirms the correctness of proxies without performing any checks might incorrectly identify a malicious object as a legitimate one.

Finally faults might occur if a service provider updates the implementation of the proxy object without updating the implementation of the verifier too. In that case legitimate proxy objects would not be able to be identified correctly, since the equality check would fail. Therefore the service provider might unintentionally cause a denial of service attack not initiated by a malicious client, but by himself.

5 AN ALTERNATIVE WAY OF VERIFYING PROXY TRUST

Instead of relying on the untrusted proxy object downloaded from an unknown source to obtain a proxy verifier, clients might be able to protect themselves from malicious proxy objects by using their own local verifier. The verifier is generated locally by clients before any participation in a Jini federation takes place. In order to create the verifier, clients specify their security requirements such as authentication, confidentiality and integrity. These requirements are injected to the verifier and might vary for different scenarios. Specification of the security requirements is similar to the concept of constraints specified by the current Jini security model (http://davis.jini.org/). This permits the specification of application independent security requirements and allows better interoperability with the current security model. The difference is that the

client requirements are not injected into a downloaded proxy, but into the locally generated verifier.

The notion of a locally generated verifier is central to all of the proposed solutions. The operations that the verifier performs, however, are different in every variation of the algorithm. The entities employed in all the solutions proposed here case are the following:

- Client: The entity that wishes to use a service. Clients need to be protected from any potential hazards.
- Proxy object: Typically the object that is downloaded by clients and used to access services. Presents the major source of incoming threats.
- Local Verifier: An entity generated by clients before any interaction with downloaded objects takes place. Used to either verify proxy objects or isolate clients from them.
- Service: The entity that lies somewhere in the network and provides some functionality. Services supply proxy objects and should be considered untrusted.

In every proposed solution it is assumed that a service has already discovered an available Lookup Service and registered its proxy object. The client is ready to perform discovery and lookup in order to obtain a proxy object from the same Lookup Service.

5.1 Proxy Verification Based on a Local Generated Verifier

In order to verify that the downloaded proxy object can be trusted, the following process is performed:

1. Before any discovery process takes place, the client generates a local verifier
2. Client's security requirements are injected to the verifier by the client
3. The client performs discovery of the Lookup Service and downloads a service proxy object
4. Before any interaction with the proxy takes place, the proxy object is passed to the verifier
5. The verifier performs a series of security checks according to the client requirements and makes a decision on behalf of the client about the trustworthiness of the proxy object
6. In case the verifier has decided that the security requirements are satisfied, the client interacts with the service through the proxy object as defined by Jini programming model.

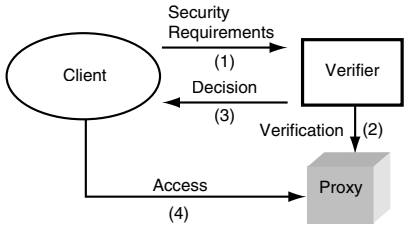

Figure 2: Proxy trust verification by a local verifier.

The described process is illustrated in Figure 2 Initially the client generates the verifier and specifies the security requirements (1). The verifier performs a series of tests to verify trust in the proxy object (2). The result of the verification procedure is expressed as a decision and the client gets notified (3). If proxy has been considered to be trustworthy, the client is allowed to contact the proxy object (4) and access the related service. Comparing this solution with the default proxy verification algorithm, in both algorithms the client is responsible for specifying the type of security required. However, the entity that is responsible for enforcing these requirements is not an untrusted proxy object anymore, but a locally generated verifier. The type of checks performed and the way these are carried out is much more transparent from the client's point of view. Moreover, clients do not have to rely on a verifier object downloaded from a service since the process of such object verifying the initial proxy object is not clear to the client.

Therefore the problem of a service generated verifier that performs no actual check to the proxy object, resulting in the verification of a faulty proxy, is eliminated.

Service providers also do not need to worry about having to provide a bootstrap proxy and a verifier. The only entity that services need to expose is the default proxy object. Absence of a bootstrap proxy eliminates the need for services to implement an object based on the assumption that it would consist of classes that the client already has. Moreover, the current algorithm dictates that every time the implementation of a proxy object changes the verifier object has to change as well, since proxy verification is based on equality checking. Finally by eliminating the need for services to produce two additional objects (the bootstrap proxy and the verifier), administration burden is removed from the service provider.

5.2 Restricting Proxy Object in a Controlled Environment

1. Before any discovery process takes place, the client generates a local verifier
2. Client injects to the verifier the security requirements and the maximum amount of local resources permitted for use by proxy objects
3. The client performs discovery of the Lookup Service and downloads a service proxy object
4. The verifier provides a controlled environment for the proxy object to run. Besides performing security checks to the proxy object, the verifier ensures that the proxy does not use more resources than specified. All requests to and from the proxy object pass through the verifier.

Figure 3 illustrates the followed process. The client generates a local verifier and assigns the security requirements as well as any resources that proxy objects are permitted to use (1). After the proxy object has been downloaded, it is passed to the verifier. The verifier performs similar type of security checks as in proposed solution 1, and additionally provides a controlled environment where proxy objects run. Any client requests and any responses from the proxy object pass through the verifier (2). The same is true for any communication held between the proxy object and its corresponding service.

The advantages of this solution are similar to those of the solution proposed in Section 5.1. The need for service providers to produce additional objects besides the default proxy object is eliminated and so are the assumptions relevant to the locality of classes in the bootstrap proxy and the checks

performed by the service's verifier. Moreover, by restricting execution of the proxy object into a set of finite resources, a protected environment safeguarded by the verifier is created. Verification does not occur only once, but the verifier is monitoring the proxy object continuously. Therefore any potential hazards that might take place during the execution of the proxy are more likely to be identified and get dealt with.

6 RELATED WORK

In (Eronen et al., 2000) certificates are used to establish trust between services and users. Secure interaction is assumed, by allowing users and services to interact only if they carry the appropriate credentials, supplied by a security library. However, these credentials must be assigned to every service of the Jini community before any interaction could be realised. That reduces the spontaneity that Jini provides, and requires prior knowledge of the services' properties to exist, in order for the appropriate permissions to be assigned correctly. Trust establishment is also the purpose of (Hasselmeyer et al., 2000a). Trust establishment is attempted between the Lookup Service, the service provider and the client. The authors propose an extension to the Jini architecture with a certification authority, which provides certifications for the authentication of components. Capability managers are responsible for administering the rights for each user. In that way, different access levels for each client can be easily implemented. Their solution, however, assumes that one central certification authority exists, in order for the appropriate certificates and capabilities to be distributed to every Jini component that exists in the system. Thus, a prior knowledge of every service's characteristics should exist something that is not usually the case in Jini. Moreover, the existence of a centralised authority is opposed to the decentralised nature of the Jini technology. The integration of authorisation and authentication techniques in Jini is also examined in (Schoch et al., 2001). The authors try to achieve that without introducing any additional components, besides the facilities that Jini and Java already provide. They try to prevent man-in-the-middle attacks, by signing the proxy object with a digital signature. This allows the clients to authenticate the source of the provided service, although it still can not be verified how the service users the provided by the service data.

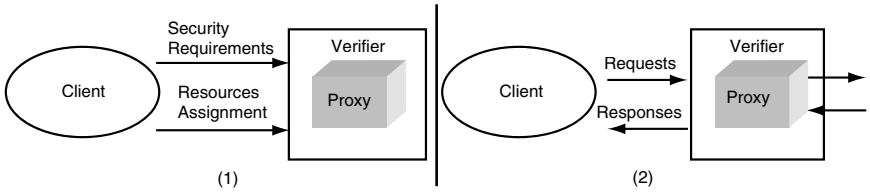

Figure 3: Verifier creation and interaction with the proxy object.

7 CONCLUSION

We presented some security problems related with Jini and how they are countered by the current Jini security model. Our focus is placed in the proxy trust verification algorithm since we believe that an alternative way of verifying proxy object trust might encounter some of the existing problems. We presented our initial ideas in providing an alternative way of ensuring that hostile proxy objects would not impose any risk to clients of the system. We sketched two different approaches in solving the problem. Both involve the concept of a local generated verifier that either verify a downloaded proxy object or impose restrictions to that object's functionality. We also pointed out the advantages of these solutions. Future work includes further rectifying the presented concepts and come up with a viable solution that would integrate with the existing model. Also implement a working prototype and test it in a real world environment.

REFERENCES

Eronen, P., Lehtinen, J., Zitting, J., and Nikander, P., 2000. Extending Jini with Decentralized Trust Management. In Short Paper Proceedings of the 3rd IEEE Conference on Open Architectures and Network Programming (OPENARCH 2000), pages 25-29. Tel Aviv, Israel.

Hasselmeyer, P., Kehr, R., and Voß M. 2000a.Trade-offs in a Secure Jini Service Architecture. In 3rd IFIP/GI International Conference on Trends towards a Universal Service Market (USM 2000), Munich, Germany. Springer Verlag, ISBN 3-540-41024-4, pp. 190-201.

Java Authentication and Authorisation Service (JAAS) http://java.sun.com/products/jaas/ [Accessed 10 Feb. 2004]

Jini specifications archive – v 2.0 http://java.sun.com /products/jini/1_2index.html [Accessed 10 Feb. 2004]

Kagal, L., Finin T. and Peng, Y. 2001. A Delegation Based Model for Distributed Trust. In Proceedings of the IJCAI-01 Workshop on Autonomy, Delegation, and Control: Interacting with Autonomous Agents, pp 73-80, Seattle.

Menezes, A., van Oorschot, P., and Vanstone S. 1996. Handbook of Applied Cryptography. CRC Press. ISBN: 0849385237

Rescorla, E. 2000. HTTP Over TLS, the IETF Network Working Group http://www.ietf.org/rfc/rfc2818.txt [Accessed 09 Feb. 2004]

Rivest, R. 1992. RFC 1321 - The MD5 Message-Digest Algorithm, the IETF Network Working Group, http://www.ietf.org/rfc/rfc1321.txt [Accessed 09 Feb. 2004]

Scheifler, Bob 2002. Comprehensive Network Security for Jini Network Technology Java One Conference Presentation, San Francisco, March 2002 http://servlet.java.sun.com/javaone/sf2002/conf/sessions/display-1171.en.jsp [Accessed 15 Dec. 2003]

Schoch, T., Krone, O., and Federrath, H. 2001. Making Jini Secure. In Proc. 4th International Conference on Electronic Commerce Research, pp. 276-286.

Sun Microsystems Inc. 2003a. Jini architecture specification. http://www..sun.com/software/jini/specs /jini2_0.pdf [Accessed 15 Dec. 2003]

Sun Microsystems Inc. 2003b. Jini architecture specification. http://wwws.sun.com/software/jini/specs /jini2_0.pdf [Accessed 15 Dec. 2003]

http://www.jini.org/ [Accessed 11 Feb. 2004] The Davis project http://davis.jini.org/ [Accessed 11 Feb. 2004

AN EXPERIMENTAL PERFORMANCE ANALYSIS STUDY OF LOSS RATE AND JITTER CHARACTERISTICS IN WIRELESS NETWORKS

M. S. Obaidat[1] and Yulian Wang[2]
[1]*Monmouth University, NJ, USA and* [2]*Tampere University of Technology, Tampere, Finland*
Corresponding Author: Prof. M. S. Obaidat, Department of Computer Science,
Monmouth University, W. Long Brach, NJ07764, USA
E-mail: Obaidat@monmouth.edu

Keywords: Wireless networks, Jitter, loss rate, mobile IP, performance evaluation, bit error rate (BER), Quality of Service (QoS), Diffserv, resource reservation, performance evaluation.

Abstract: Among the challenges in wireless networks is the high bit error rate, which is due mainly to atmospheric noise, physical obstructions found in the signal's path, multipath propagation, interference from other systems and terminal mobility. This high bit error rate makes it difficult to offer guaranteed services over the wireless link. In this paper, we present an experimental analysis study on the loss rate of wireless systems using six different scenarios. We also present an experimental study of jitter for UDP traffic over wireless Mobile IP networks. It is found that in order to provide different Quality of Service, QoS, to downstream traffic flows and control network's loss rate and jitter, it is not enough to have only DiffServ flow control mechanisms. A protocol for wireless link resource reservation and cooperation by senders is also needed. We identify the relationship of jitter, loss rate and class allocation's effect using Class Based Queueing (CBQ) and the packet sending rates in the wireless networks. It is found that loss rate and jitter can be controlled with DiffServ flow control mechanism, but it requires that the total traffic rate should be within the limit of the wireless link capacity. Various tests have been conducted under different settings and operating conditions.

1 INTRODUCTION

1.1 Study of Loss Rate

There are fundamental differences between wireless and wired LANs, which pose difficulties in the design of such systems and protocols (Nicopolitidis, P. et al., 2003), (Nicopolitidis, P. et al., 2002). (The wireless medium is characterized by high bit error rates (BERs) that can be ten times than that for wired LANs. Moreover, errors in wireless LANs occur in bursts, whereas in traditional wired systems errors appear randomly. Among the challenges in wireless networks are: (a) wireless medium unreliability, (b) spectrum use, (c) power management, (d) security, (e) routing, and (f) interfacing with wired networks. The phenomena causing reception errors in wireless systems are: (a) free space path loss, (b) Doppler shift due to station mobility, and (c) multipath propagation due to mechanisms such as reflection, diffraction, and scattering. Such mechanisms cause the signals to travel over many different paths (Nicopolitidis, P. et al., 2003), (Green, D. and Obaidat, M. S., 2003). Mobile IP protocol is an extension of the Internet protocol intended to support mobility in the Internet across all kinds of networks, both fixed (wire-line) and wireless types. When employed with wireless access networks, it can be used to create truly mobile networks. In practice, it enables people to access the Internet continuously with their laptop computers and other portable IP-capable devices while moving around an area covered by wireless LANs.

Mobile IP was developed in response to the increasing use of mobile computers in order to enable them to maintain Internet connection during their movement from one access point to the other. The term mobile here implies that the user is connected to one or more application across the Internet and the access point changes dynamically.

J. Ascenso et al. (eds.), e-Business and Telecommunication Networks, 19–29.