A Synthetic Fraud Data Generation Methodology

Conference Paper · December 2002

DOI: 10.1007/3-540-36159-6_23 · Source: DBLP

Håkan Kvarnström

7 PUBLICATIONS **71** CITATIONS

THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

On the Implementation and Protection of Fraud Detection Systems

HÅKAN KVARNSTRÖM

Department of Computer Engineering
School of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden, 2004

On the Implementation and Protection of Fraud Detection Systems

Håkan Kvarnström
ISBN 91-7291-461-0

Doktorsavhandlingar vid Chalmers Tekniska Högskola
Ny serie: 2143
ISSN 0346-718X

School of Computer Science and Engineering
Chalmers University of Technology
Technical Report No. 27 D
ISSN 1651-4971

Department of Computer Engineering
Chalmers University of Technology
SE-412 96 Göteborg, Sweden
Phone: +46-(0)31-772 1000
http://www.ce.chalmers.se

Author contact info:
http://www.ce.chalmers.se/staff/hkv

Printed in Sweden
Chalmers Reproservice
Göteborg, Sweden, 2004

On the Implementation and Protection of Fraud Detection Systems

HÅKAN KVARNSTRÖM
Department of Computer Engineering
Chalmers University of Technology

Abstract

Society of today is becoming increasingly dependent on the availability and correctness of IT-systems. There are a growing number of industries that base their business almost exclusively on creating growth and economic value using IT-systems. Banks, credit card providers, online gambling brokers and providers of telecommunication services are a few examples where secure and reliable communication and computing environments are of the outmost importance for survival. These new services can create increased customer value and provide new business opportunities. However, history shows that fraudsters constantly seek economic gain by exploiting weaknesses in technical systems or business models of service offerings. Thus, minimizing economic loss due to fraud is an important goal in many enterprises. An important tool in that process is an efficient fraud detection system (FDS) that can give early indications of misuse of products and services offered to customers. This thesis addresses the subject of implementing and protecting an FDS. Specifically, we give arguments for the importance of thorough testing, training and sufficient protection when deploying a successful fraud detection system. We show how emerging computing environments put new demands on the security mechanisms and security extensions that protect the information and resources of their target environment. Fraud detection systems are no exception and we argue that detection mechanisms will play an important role in such environments. We propose and implement a methodology for generating synthetic data sets that can be used for testing and training fraud detection systems prior to real-life deployment. A comparison of fraud and intrusion detection system is made, where we highlight differences and similarities. Our study shows that the similarities clearly outweigh the differences, and research in one area is most often applicable for both types of systems. Finally, we discuss the topic of protecting fraud detection systems from malicious environments. Specifically, we study the importance of such systems from reverse engineering and propose mechanisms for achieving secrecy of the inherent information, even when deployed in hostile environments.

Keywords: computer security, fraud detection, intrusion detection, fraud, security architecture, computer misuse

List of appended papers

This thesis is based on the work presented in the following papers.

Part I: Implementing fraud detection systems

Paper A	Håkan Kvarnström, Ulf Larsson, Erland Jonsson. New security issues in emerging computing environments - A reflection. *Technical Report 04-02*. Department of Computer Engineering, Chalmers University of Technology, SE-412 96, Göteborg, Sweden.
Paper B	Emilie Lundin, Håkan Kvarnström, Erland Jonsson. Synthetic Fraud Data Generation Methodology. In *Proceedings of the Fourth International Conference on Information and Communications Security*, ICICS 2002, Singapore. December 9-12 2002. Springer Verlag.
Paper C	Emilie Lundin, Håkan Kvarnström, Erland Jonsson. Synthesizing test data for fraud detection systems. In *Proceedings of the Nineteenth Annual Computer Security Applications Conference*, ACACS 2003, Las Vegas, Nevada, USA, December 8-12 2003. IEEE Computer Society.
Paper D	Håkan Kvarnström, Emilie Lundin, Erland Jonsson. Combining fraud and intrusion detection systems - Meeting new requirements. In *Proceedings of the Fifth Nordic Workshop on Secure IT Systems*, NordSec 2000, Reykjavik, Iceland, October 12-13 2000.

Part II: Protecting intrusion and fraud detection systems

Paper E	Hans Hedbom, Håkan Kvarnström, Erland Jonsson. Security Implications of Distributed Intrusion Detection Architectures. In *Proceedings of the Fourth Nordic Workshop on Secure IT Systems*, NordSec 1999, Stockholm, Sweden, November 1-2 1999.
Paper F	Håkan Kvarnström, Hans Hedbom, Erland Jonsson. Protecting Security Policies in Ubiquitous Environments Using One-Way Functions. In *Proceedings of the First International Conference on Security in Pervasive Computing*, SPC 2003, Boppard, Germany, March 12-14 2003. Springer Verlag.
Paper G	Håkan Kvarnström, Hans Hedbom, Erland Jonsson. Protecting Stateful Security Policies using One-Way Functions. In *Proceedings of the Asia Pacific Information Technology Security Conference*, AusCERT 2004, Gold Coast, Australia, May 23-27 2004.

Table of Contents

Acknowledgements

I would first like to thank my supervisor, Erland Jonsson, for his support, encouragement and valuable comments throughout this work and for giving me the opportunity to join the computer security group at Chalmers. In times of great despair and agony, a conversation with Erland has proven to be the medicine needed to find new strength and motivation. I would also like to thank my current and former fellow researchers at Chalmers: Emilie Lundin Barse, Dan Andersson, Stefan Axelsson, Ulf Lindqvist, Stefan Lindskog, Hans Hedbom, Ulf Gustafson, Magnus Almgren and Ulf Larson. It has been stimulating to work and have discussions with you. A special thanks to Emilie Lundin Barse and Hans Hedbom for being perfect partners in research projects and paper-writing sessions for many years. I hope I will have the chance to continue to work with you.

Thanks also to TeliaSonera for believing in me and allowing me to pursue my PhD studies even though it took a few more years than I first anticipated. A special thanks goes to Dr. György Endersz, the person who initially supported and encouraged my ambitions to pursue a PhD. Part of my, and my co-authors', work was funded by organisations outside Chalmers University of Technology and TeliaSonera. A major sponsor has been the "The Swedish Emergency Management Agency". I am grateful for all financial support we have received through the years.

A big thanks to all anonymous reviewers that provided all the valuable comments on my work. Your enthusiasm has greatly helped to improve the quality of my publications.

I am also grateful for all the help I have gotten from the administrative staff at the Department. A special thanks to Ewa Cederheim-Wäingelin. She has always been helpful and solved many problems that have saved me many trips to Gothenburg. I would also like to express my gratitude to Janet Vesterlund for proofreading most of the material in this thesis. She always found bloops in my work even in cases where I was absolutely sure I had gotten everything right.

I thank my parents, Tore and Maj, for encouraging me to study. Your support was essential during my first years of academic studies. Finally, I would like to thank Charlotta for supporting and encouraging me during my studies. Many nights and weekends were consumed by endless sessions of reading, writing and hacking.

Håkan Kvarnström, June 2004, Göteborg, Sweden.

Introductory summary

1 Introduction

Conducting research in the field of information and computer security is both challenging and rewarding. Not only does it necessitate a profound knowledge of subtle technical details of protocols and mechanism, it also requires a broad understanding of disciplines such as software development, system architectures and computer human interaction. I believe that the fact that the field of information and computer security must always be studied in the context of other disciplines and research areas is what makes it so interesting and rewarding. I started my career in computer security by looking at security from the perspective of the user. I found that one could do interesting things with computers beyond their intended purpose, design and use. The stimulating part of this exploration was more than just outsmarting a piece of technology. It was a battle of minds: me trying to outsmart designers, implementers and administrators; them constantly trying to improve their designs, keeping their systems secure against adversaries such as myself. Fortunately, I did not use my skills for illegal purposes but directed my energy towards well organized penetration tests and security analyses. It is a widely known fact that many intrusion attempts and attacks result in benefitting the attacker. Statistics tell a sad tale in this case [Com03]. Not because the hacker is smarter, but because he or she[1] need only find a single flaw to exploit a system, whereas the designer and the implementer need to find every possible flaw in order to be secure. This unbalance may seem unfair, but who has said anything about system development being fair? The question is whether it is possible to protect yourself under such conditions. Lecutus of the Borg once said: *"Resistance is, and always has been, futile."* So, is resistance futile? Faced with this irrefutable fact of unbalance, I had two choices. I could let this injustice defeat me and surrender to the threat, or I could do my best to find new ways to bring balance to the equation. I chose the second option and this thesis is my contribution.

1.1 The importance of security

Security is important in today's society. The main reason is that information technology is playing an increasingly important role for organizations and individuals of all types. The public sector relies on information and communication systems to provide everyday services such as health care and e-government to society and its population. Corporate organizations move parts of their business towards online services providing e-commerce, information and communication services to allow their customers to increase their efficiency and competitiveness. They rely on the availability, confidentiality and correctness of information and resources for the success of their business. Large network infrastructures connect subsidiaries around the globe while numerous short-range access networks enable end-users and their terminals to communicate from wherever their physical

[1] For the sake of readability I will henceforth use *he* when in fact I mean *he or she*

location might be. End-users benefit from increased access and availability to information and entertainment services, allowing them to save time, as well as kill time, by having seamless and continuous access to their services and resources. As security evolves from being a mainly corporate issue to a personal dilemma, new problems arise. Information and resources such as bank accounts, payment services and meta data concerning a user and his whereabouts are constantly trafficking global networks waiting to be hijacked by non-scrupulous persons. One can observe that the need for information security (both corporate and personal) has increased dramatically in the last few years due to the penetration of information technology into everyday business and personal lives. Most security experts agree that 100% security can never be achieved. There is always a balance between a sufficient level of security and the cost for its deployment.

1.2 Why does security fail?

An important question is why today's security controls are failing so miserably in protecting our information and resources. It should be noted that experienced security engineers and researchers have not yet found a way to successfully address escalating security problems and their consequences. One reason for the difficulty of providing security is the complexity of its nature. As previously stated, security must be tackled in relation to a context. Security controls by themselves serve little purpose. Deploying security involves information, systems, computer networks, humans and many other complex beings to work together. Bruce Schneier[2]once said; *"The future of digital systems is complexity, and complexity is the worst enemy of security"*. The reason for this complexity is multidisciplinary and far from being a question of technology alone. Below I will briefly discuss some of the issues affecting security and hopefully give an indication of the complexity of the problem.

The inability to learn from mistakes. One reason for the escalating security problem is that the security community tends to have a difficult time learning from mistakes. There is a reluctance to be educated or to educate, that is deeply rooted in the field of security. One can say that the struggle for security backfires on itself. Organizations that experience security problems do not share their experiences. Experiencing break-ins and other security incidents is considered a bad thing and few are willing to openly share their problems and solutions. Hence, the learning process is slowed down. Typically, what is happening today is that, when an attack is found, it is kept secret. The problems are fixed silently and the community learns nothing from the process. This is a serious problem in the field of security today.

The security investment paradox. Companies and organizations invest in security to reduce their technical and business risks. However, a successful imple-

[2] Founder and currently CTO of Counterpane Internet Security, Inc

mentation of security controls, which minimizes fraud and other security incidents, will most likely require a substantial investment in equipment, education, staff etc. The paradox is that the more you invest, the more successful you are likely to be at keeping incidents at bay and the less you are reminded of the need for the investment. It is easy to underestimate the risks if you rarely experience security incidents. The unwillingness among companies to share their incidents with other does not improve the situation.

Lack of metrics for measuring security.

"What is not measurable make measurable - Galileo Galilei (1564-1642)"

A fairly well known problem in the field of security is that it is difficult to measure the level of security provided by a security service. In many respects, security is binary. There is a very fine line between security and insecurity as it only takes a single vulnerability to render the security useless. Building a layered and fault-tolerant security architecture with redundancy helps to improve the security, but it is still hard to make a reliable measurement of the strength of security. Many researchers have worked on the issue of assigning a measure to security. I frequently mention two references that I use as examples in my work as a security professional.
The work of Jonsson [Jon97] views security from a dependability point of view. The advantage of this approach is that well-established measures for reliability can be applied to model the level of security of a system. Jonsson used his models to calculate the level of *degradation* caused by an attack and gave predictions of the Mean time To Degradation (MTTD). In research by Dacier [DDK96], *privilege graphs* were used to estimate a quantitative level of the security of a system. Hence, he could give a quantitative measure of a systems ability to resist an attack.
I would like to stress that this is a major problem today and is in my opinion one of the greatest obstacles to a broad understanding of the needs for security. The lack of formal qualitative and quantitative risk metrics makes it difficult for risk managers and security professionals to measure the effect of investments in security. Quantitative methods have proved useful in other sciences and the security field must find ways to quantify its usefulness. Tom DeMarco writes, *"You cannot control what you cannot measure"* in his book, Controlling Software Projects. In the end, quantitative measures are the only things that count when management evaluates business risks on which they base their business decisions.

An immature field of engineering. Even though the field of computer and software engineering is nearly 60 years old (since the introduction of the ENIAC in 1946), software development can in many ways be considered an immature field of engineering. In most countries you need a degree and a license to practice medicine or law, but you need no formal training whatsoever to be a software developer. Even though the software may be used in life critical applications, you don't even need a formal specification or requirements capture of the software. In addition, even *with* proper training, humans make mistakes. Without training, they can be expected to make even more. Imagine a bridge being built without a formal specification or by people with insufficient training. Of course, many developers do develop these specifications, but the customers seldom show any interest in them. This does not mean that software users are ignorant or stupid. It means that they, for some reason, trust the vendors to provide trustworthy products. Again, compare a piece of software to a bridge. Only the most dedicated bridge geek would ask for its specification before his crosses it. That is probably one of the reasons why certification and evaluation, such as according to FIPS, Orange Book and TC SEC, are not in widespread use except for specific high security applications. Increased cost is another. Dan Geer exemplifies current software engineering: *"We're letting creative artists build bridges... then trying to stabilize them with unlicensed labors while they're collapsing"*. The software community is painfully aware of this situation, which resembles the way technical development took place in the beginning of the industrial revolution. It is scary how many pieces of software come to be, that in fact should not.

Lack of liability. Many people believe that the current situation with vendors selling immature software is jeopardizing the long-term development of IT. One of the reasons that enable them to do this is that they deny liability for their products. For example, a car manufacturer has a far more widespread liability than any software vendor which forces the company to do things right from the beginning, or at least do its very best to avoid flaws and poor design. Of course, this should not only apply to software vendors but also to system integrators and service providers. Most often, perfectly good components are put together in ways that make security suffer. I believe that the development towards greater liability is necessary to increase the quality, and hence the security, of software.

Lack of diversity. Most organizations try to minimize their investments in and maintenance costs for IT. In the process, a homogenization of the IT environment is often done to avoid a situation where they have too many different applications and platforms to manage. However, homogenization often leads to less diversity. Diversity is a double-edged sword. Low diversity often means low complexity, which is good for security. At the same time, low diversity means that all the eggs are put in one basket. If there is only one brand of firewalls, chances are that a discovered flaw will affect all the company's firewalls and thereby create a serious threat. This of course violates the principle of layered security architectures to provide redundancy in case a mechanism fails. Good or bad, but history shows that companies that focus on software from a single vendor are likely to suffer serious consequences in case of a discovered flaw. The overwhelming number of published attacks against products from Microsoft is evidence of that.

Security design principles. Designing software and systems with security in mind is not easy. There are many pitfalls, and history shows that most developers sooner or later fall into the pit. One reason for this situation is that basic and fundamental design principles are not followed. Principles that have been known for years. Saltzer and Schroeder [SS75] specified eight different design principles for building secure computer systems: *economy of mechanism; fail-safe defaults; complete mediation; open design; separation of privilege; least privilege; least common mechanism and psychological acceptability*. This was written in 1975 and many of their principles are still routinely broken during software development. In 2002, Microsoft set an ambitious goal to improve security in their products. They launched the ”Trustworthy Computing” initiative in an effort to reduce the number of security problems in their products [MdVHC03]. The move included special training for programmers and a year later over 20 products have gone through the Trustworthy Computing Release Process and more than 11,000 engineers received training in writing secure code. Although this is an interesting challenge, only time will tell whether the number of bugs will be reduced in their software.

Security stumbling on itself. In my career as a security analyst and designer I have implemented many security functions for telecom services and applications. However, adding security is not always a straightforward process and may lead to new security problems. A security function may itself contain information or features that can be exploited by attackers. A logging function may disclose information to an attacker that could enable him to guess usernames and passwords. A firewall meant to protect a group of hosts could disclose information about hosts, protocols in use and the network topology of the target systems. This was pointed out by Hedbom, Lindskog and Jonsson [HLJ99], who argued that the reliability of added security mechanisms is often dependent on the security of the mechanism whose insecurity they are supposed prevent. And vice versa. They called such security functions *security extensions*. The idea of security extensions is central in this thesis. In the forthcoming sections, intrusion and fraud detection systems will be discussed in great detail. Such systems can clearly be classified as security extensions as they themselves are security mechanisms aiming to prevent other flaws and vulnerabilities from being exploited. Indeed, I will later show that introducing intrusion and fraud detecting capabilities to a system may jeopardize the system’s overall security. Thus, security extensions must be implemented and used with caution.

2 Countering security threats

Criminal activities and computer system attacks are becoming increasingly common. Not only is the number of attacks growing but the level of sophistication has also increased. As a result, the impact on individuals and organizations is greater than previously. A recent study [cU02] by the UK's National High-Tech Crime Unit (NHTCU) indicates that the growth rate of cyber crime is doubling every 18 months, as according to Moore's law. If this alarming growth rate continues, it is not a matter of *if* high-tech cyber crimes will outnumber traditional crimes, but *when*.
Cyber crime and fraud need to be combated at many different levels. At the political and legislative level, laws and regulations must ensure that the use of IT systems for managing resources, information and business transactions can be controlled and regulated. Unfortunately, the development of such legislation has been slow, which has left a window of opportunity for fraudsters and criminals to exploit the lack of applicable laws. To remedy some of the problems, the European Union released two directives, the *electronic signature directive* [fOPotEC99] and the *directive on privacy and electronic communications* [fOPotEC02] which create a legal framework and establish criteria for companies, organizations and individuals to conduct business without jeopardizing security and privacy. In most European countries, the protection of personal data is a constitutional principle. Citizens' right to privacy is part of the European Convention on Human Rights. Laws and regulations are necessary to be able to prosecute cyber criminals. However, the rest of this thesis will limit its scope to technical and administrative issues and their corresponding problems and solutions.

2.1 Protecting IT systems

Finding countermeasures to protect computer systems is not a trivial task. It is not just a matter of adding some encryption, authentication and logging services to take care of security. What kinds of countermeasures should be in focus? What kinds of threats are we seeking to find protection against? To bring some order to this problem, a categorization of security countermeasures can be used. The categorization shown in Figure 1 was inspired by [Sch00].

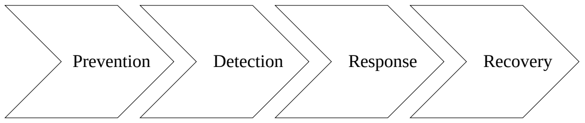

Figure 1: Security Countermeasures

These four categories complement each other and, if we focus our countermeasures on one of the categories, we may give less attention to others. For example, if we put all our efforts into protecting our target environments, using

strong authentication, encryption and access control, we may relax our requirements for detection and response. This can be done because we may feel so confident that our *preventive* countermeasures do their job so well. On the other hand, if we know that our target environment suffers from weak authentication (e.g. passwords in clear text) and the access control is based on those weak presumptions, we may want to increase our countermeasures for detecting intrusions and fraud. When deploying security, countermeasures are normally distributed over these categories to create *a layered security architecture*. Attacks circumventing the preventive measures can then be detected and hopefully stopped before damage occurs.
The four categories can be described as:

Prevention. By deploying preventive countermeasures we aim to prevent security threats by eliminating as many vulnerabilities as possible. Examples of mechanisms are: authentication, encryption and access control schemes. Preventive countermeasures can be compared to the guards, doors and locks in a building. They prevent wrongdoers from doing wrong.

Detection. This class of countermeasure aims to detect intrusions, fraud or attempts thereof. They also serve as a second layer defense when preventive mechanisms fail. The detection may identify misuse by insiders (legitimate users) as well as external adversaries. It detects wrongdoers when they do wrong.

Response. Taking proper action after an attack or security breach is handled by a response countermeasure. An attack may lead to an action to close down sensitive information systems to prevent a breach of confidentiality. Other actions may be to notify a security officer or reconfigure packet filters (e.g. firewalls) to block incoming network traffic from certain IP addresses. A proper response is essential to make detection useful. A detection system (DS) screams in vain if no one is listening and is ready to respond.

Recovery. Recovering from an incident means that the system must be restored to its latest good state. Recovery can affect everything from administration to technology. Administrative routines may have to be changed, new passwords assigned to users and the integrity of applications and data must be reassured. Also, log files may need to be restored from backup to allow forensic analysis. Full recovery may not always be possible, as a breach of confidentiality is irreversible. Once the information is disclosed to an attacker, the secret is out and cannot be reversed unless we apply significant physical force to the attacker's head. The confidentiality of information used by the security extensions themselves is an important and relatively unexplored field. In Paper E, we explore this field and show how it affects intrusion detection systems and firewalls.

2.2 Setting the stage for intrusion and fraud detection.

The fields of security can be viewed upon from many different angles and perspectives. People with different occupations and backgrounds often interpret the field on the basis of their own experiences. For a doctor of medicine, *integrity* means that a patient can rely on the doctor's integrity not to disclose sensitive information, such as the patient's medical condition. For a risk manager, integrity means that information is protected against illicit modification. This situation is typical in many disciplines and it is important to provide clear definitions of terms upon which you build further reasoning. In this thesis, many terms are used that may be interpreted differently among different readers and I will therefore define the most important terms to avoid confusion.
Many sources can be found that give their own definition of security. In my role as an engineer, I usually rely upon definitions given by international standardization bodies. These definitions are widely used by industry when writing requirement specifications and documentation. The definitions provided by the standardization bodies are often the result of endless discussions and arguing among researches and practitioners in the field. Being a researcher, I find myself looking for definitions in research publications, which are sometimes less useful in the eyes of an engineer. The first three definitions are straightforward and are seldom subject to much controversy [fOPotEC91].

Confidentiality: Prevention of unauthorized disclosure of information.

Integrity: Prevention of unauthorized modification of information.

Availability: Prevention of unauthorized withholding of information and resources.

The definitions above make up the popular *CIA* acronym. However, many other important terms are worth defining. The *authenticity* of data and resources gives assurance about the identity of a piece of data or a communicating party. *Non-repudiation* services prevent a party from denying that he performed a certain action, such as creating, deleting, sending or receiving some data.

Besides the general security definitions above, specific definitions related to fraud and intrusion detection are frequently used in this thesis. I have already mentioned the words *intrusion* and *vulnerability* many times, so it is upon time to properly define them and a few other terms that I frequently use, and sometimes misuse, in my papers.

Intrusion: An attack in which a vulnerability is exploited, resulting in a violation of the implicit or explicit security policy.

Fraud: An intentional deception or misrepresentation that an individual knows to be false that results in some unauthorized benefit to himself or another person. It should be noted that persons inside an organization or external to it can commit fraud. Insider fraud may be very difficult to find, especially when conducted by senior management. Such fraud is not a completely rare occurrence in today's corporate environments. Fraud investigators often talk about the *10-80-10* law, which states that 10% of people will never commit fraud, 80% of people would commit fraud under the right circumstances, and 10% actively seek opportunities for fraud.

The definitions of *vulnerability* and *security policy* are taken from the thesis presented by Ulf Lindqvist [Lin99].

Vulnerability: A condition in a system, or in the procedures affecting the operation of the system, that makes it possible to perform an operation that violates the explicit or implicit security policy of the system.

Security policy: A statement about what kinds of events are allowed or not allowed in the system. An explicit policy consists of rules that are documented (but not necessarily correctly enforced), while an implicit policy encompasses the undocumented and assumed rules that exist for many systems.
You cannot say whether a certain action is a crime unless there is a law that defines right and wrong. The security policy is the equivalent of the law in our legal system. It can be used by screening devices (e.g. firewalls) to prevent unauthorized access to resources and information. It can also be used by misuse detection systems to identify malicious and hostile activity.

2.3 An introduction to intrusion and fraud detection systems

Intrusion detection and *fraud detection* are terms that define the category of mechanisms that aim to detect intrusion and fraud according to the definitions above. They raise an alarm once they detect the occurrence of an intrusion or fraud, similar to a smoke detector that triggered at the occurrence of fire and smoke. If traditional preventive countermeasures fail, the detection mechanism should react. In [Lin99], traditional preventive mechanisms are depicted by a wall that protects the target system from the attackers. A detection system should respond to the occurrence of any of the events illustrated in Figure 2.

2.3.1 Intrusion detection systems

Intrusion and fraud detection systems analyze information from the target computer system to try to isolate and identify any misuse [And80]. The difference between intrusion detection systems (IDS) and fraud detection systems (FDS) is not entirely clear. In short, however, they share more similarities than differences, and paper D in this thesis explores this issue in great detail.

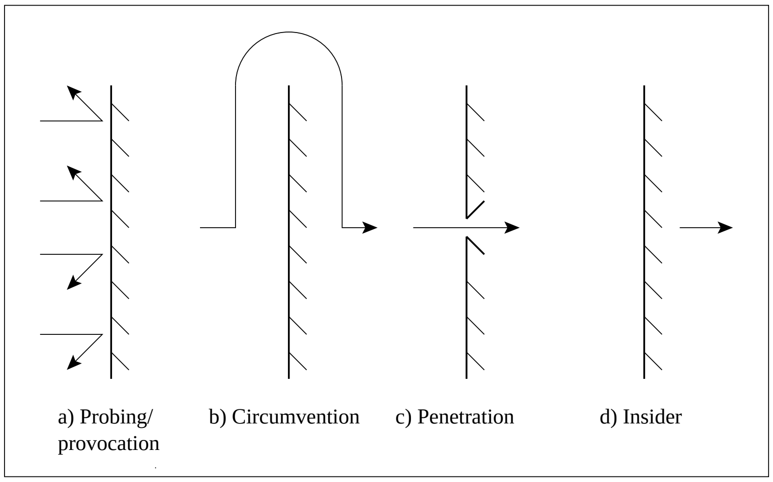

Figure 2: Events that should trigger IDS response

Intrusion detection systems are in no respect a novel invention. Japanese engineers made one of the very first designs of intrusion detection systems, based on technology, over 400 years ago. The Nijo Castle in Kyoto, which was built between 1601 and 1603 and supervised by Itakura Katsushige, an official representative of the Shogun in Kyoto, used an innovative mechanism for dealing with intruders. Despite the challenging political and military climate in Japan at the time, the castle was considered to be particularly unfortified (i.e. it lacked sufficient preventive countermeasures). Fences were built around the castle but it more resembled a defenseless princely estate than a defensive structure. To balance the lack of preventive countermeasures, they installed ”nightingale” floors designed to squeak to warn of the presence of ninja assassins. The inner rooms of the building had tatami matting [3] on the floors, covering huge smooth dark floorboards, which squeak and chirrup as you cross them. They put devices of metal pins and springs under certain boards so that when they were walked uponit would sound like a nightingale and the owner of the house would hear an intruder. In short, they followed the same countermeasure model as described in Figure 1. They had fences around the house to ”prevent” intrusions, nightingale floors to ”detect” them and swords to ”respond” to them. A proper ”response” (e.g. killing the ninja assassin) would require very little ”recovery” beyond neatly disposing of the remaining pieces. Security problems are similar, but our mechanisms to prevent, detect and respond to them have been slightly refined.

[3] Tatami mats are thick, woven mats used in Japanese houses as a floor covering and are usually made of a rice straw backing with a facing of woven rushes.

A great number of commercial Intrusion Detection Systems have appeared in the past few years [Kva99]. Although there has been research in this area for at least two decades, only very simple techniques have been adapted in commercial systems. In research, a wide range of mechanisms and techniques have been tested and analyzed. An overview of the research in the field of intrusion detection is given by Axelsson [Axe98]. Most of the techniques used for fraud detection have also been used for intrusion detection. It is common practice that commercial intrusion detection systems use network packets and host-based log files, often from the operating system, as input. In other systems, application log files such as firewall logs or logs from a web server can provide input for analysis.
The most common detection strategy in IDSs is to create rules for known intrusive behavior. This strategy is called *misuse detection* and is often implemented as rule-based expert systems or as simple pattern matching. Events that these systems detect are e.g. network denial of service attacks, such as SYN-flooding and buffer-overflow attacks that are used to gain greater privileges. The other detection strategy is *anomaly detection*. In anomaly detection, behavior profiles are created and deviations from these profiles are detected. It can be implemented using e.g. statistical methods or neural networks. This strategy can be used to detect masqueraders, password guessing etc. A discussion of detection strategies and implementation methods is given in [HB95].

2.3.2 Fraud detection systems

Fraud detection systems share many features with intrusion detection systems and to some extent solve an easier problem. Typically, an FDS has fewer capabilities than an IDS. A fraud detection system deals with the event categories of *Circumvention*, *Penetration* and *Insider* (as described in Figure 2). Fraudsters seek to *circumvent* billing and charging systems to gain some economic benefit. They *penetrate* access control mechanisms in an attempt to manipulate payment streams. Of course, an *insider* can do all of the above and try to cover up his actions by deleting logs and audit trails. *Probing* and *provocation* of systems are usually not detected by an FDS unless they eventually result in some economic loss or benefit. I usually describe a fraud detection system as a system that detects the *consequence* of an intrusion or attack. The FDS detects that, for some reason, someone is violating the implicit or explicit security policy of the system. It cannot necessarily say *why* or *how*, but someone or something shows unusual behavior. This is very similar to anomaly based intrusion detection systems, which may also have a difficult time figuring out what really triggered an alarm. FDS uses a variety of data analysis techniques to find a fraudster among a large number of honest users. This is like looking for a few needles in a haystack, but harder. In the needle and the haystack problem, all needles look the same and the hay can easily be recognized as being different from the needles. All you have to do is to put all straws of hay to the left and all the needles to the right and after some time, depending on the size of the haystack, you have one small pile of needles and a large stack of hay. This classification of needles and straws requires only two categories. In the field of fraud detection, there can be many different types of fraudsters and many will act and behave very similar to non-fraudulent users. The classification of events as fraudulent or non-fraudulent is the key issue for every fraud detection system. It aims at correctly assigning events into a limited set of classes with as few incorrect assignments as possible. In certain applications, two classes of fraud / non-fraud may be sufficient whereas, in other applications, it may be necessary to use several classes, as depicted in Figure 3. For certain detection mechanisms, a probability measure can be assigned to each class, giving indications of the likelihood of a certain event being fraudulent.

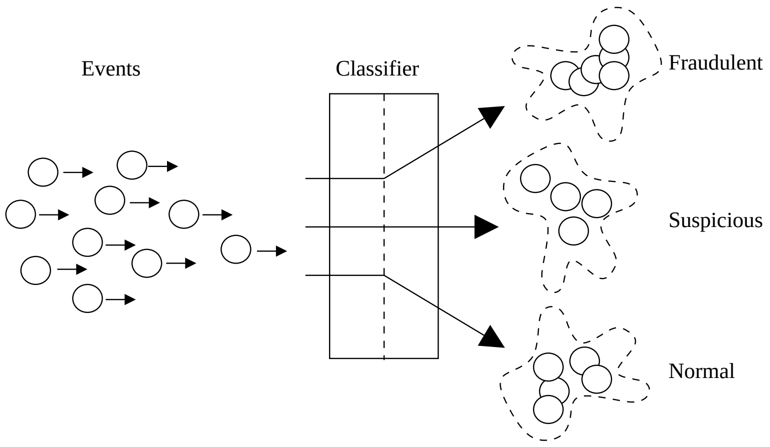

Figure 3: Detecting fraud by classification of events

Similar to intrusion detection, a policy must be in place for the FDS to be able to distinguish between the righteous and the evil. This classification of events into different classes has been used in many different fields. Banking and credit card companies apply FDS to find customers that try to use other people's credit cards. The telecom industry tries to identify telephony service subscribers, that use excessive amounts of services with no intent of paying[4]. Common to both domains are the tools and methods used to detect fraud. Data mining and various intelligent techniques have successfully been used to detect fraud and many different intelligent techniques have been proposed, developed and evaluated [wg00b]. Other intelligent techniques include rule induction, decision tree induction, instance-based learning, ANN, k-NN, Bayesian nets, and SVM, among others [FPSSE96].

4 More information on this topic can be found in paper A.

2.3.3 Dealing with false alarms

Similar to intrusion detection systems, FDS are subject to the problem of false alarms. A *false positive* is an alarm generated when the detection system erroneously classifies a normal event as intrusive. When the detection system instead classifies an intrusive event as normal, it is called a *false negative*. Under ideal conditions, false alarms should not exist. A correctly classified event is either a *true positive* or a *true negative* depending whether it is intrusive or not. A high number of false alarms have the effect that the security manager loses his confidence in the system, just as your neighbor would if your car alarm went off every night for no reason. Eventually, the alarms are ignored or the detection system is disconnected and it no longer serves a purpose.
There are many different methods for reducing false alarms. A popular method is to define a threshold that must be reached before an alarm is raised. Thresholds can be defined in many ways. For example, intrusions considered severe could trigger an alarm, whereas less significant intrusions such as a port scan, could be discarded. Of course it is very hard to define a threshold and it will most likely vary from implementation to implementation. In fraud detection systems, the threshold is often based on some kind of economic loss. We can safely ignore fraud that costs more to investigate than would be lost. A single occurrence where a phone boot has been manipulated and shows irregular behavior may not be worth investigating, whereas a large organized *call selling operation* (see paper A) may raise serious concerns. In short, we could focus on significant fraud and ignore the others. This is different from intrusion detection where it is difficult to define an insignificant intrusion. Another popular method for reducing false alarms is *correlation*. Event and alert correlation allows the detection system to make better judgments of the situation. Events from different sources can be correlated to get a better picture of the nature of the system's use or misuse. Alarms from different detection systems can be correlated to refine the building of the intrusion scenario, which helps the detection system tp discriminate between false positives and false negatives. A recent thesis and excellent study on correlation issues is given by Gorton [Gor03].

2.3.4 Components of a detection system

IDS and FDS share many characteristics and functional components. A framework for describing intrusion detection architectures and their components is CIDF (Common Intrusion Detection Framework) [Tun99]. The architectural components of CIDF are shown in Figure 4 and consist of four different types of functional blocks. CIDF is maintained by the CIDF working group, which was originally formed as a collaboration between DARPA (Defense Advanced Research Projects Agency) funded intrusion detection and response (IDR) projects. The design goals of CIDF are to develop a set of specifications that allows different IDR components to interoperate and share information and allows different IDR subsystems to be re-used in contexts different from those for which they were designed.

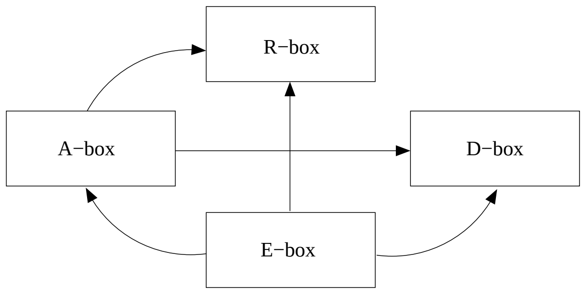

Figure 4: Components of CIDF (Common Intrusion Detection Framework)

The modularization of functions into boxes allows intrusion detection components to be reused in other systems. Each box is specialized to perform its respective task.

Event Generators (E-boxes). The E-boxes generate events by collecting information from their environment. They produce messages that describe their observations and that are forwarded for further processing or storage.

Event analyzers (A-boxes). A-boxes analyze messages in order to draw conclusions. The conclusions are sent in messages to R-boxes for a response or to be forwarded to other A-boxes for further analysis.

Event databases (D-boxes). Persistent storage of messages is provided by D-boxes. Stored messages can later be retrieved and forwarded to their intended destinations.

Response units (R-boxes). R-boxes are response units carrying out actions on behalf of other CIDF components. They receive orders from messages provided to them by other boxes and carry them out.

CIDF is an excellent architecture for providing interoperability among components in a detection system. However, in my research, I have found a more detailed architecture useful in certain situations. In Paper D, we used an extended architectural model to discuss similarities and differences between IDS and FDS (as shown in Figure 5). I will also use this architectural model in this introduction to explain, in more detail, important functions and components of a detection system.

Target. The targets that are monitored are usually some kind computer system, application or service that can be subject to fraud or intrusion. In a telecom environment, a typical target is the cell phone service that can be subject to fraud.

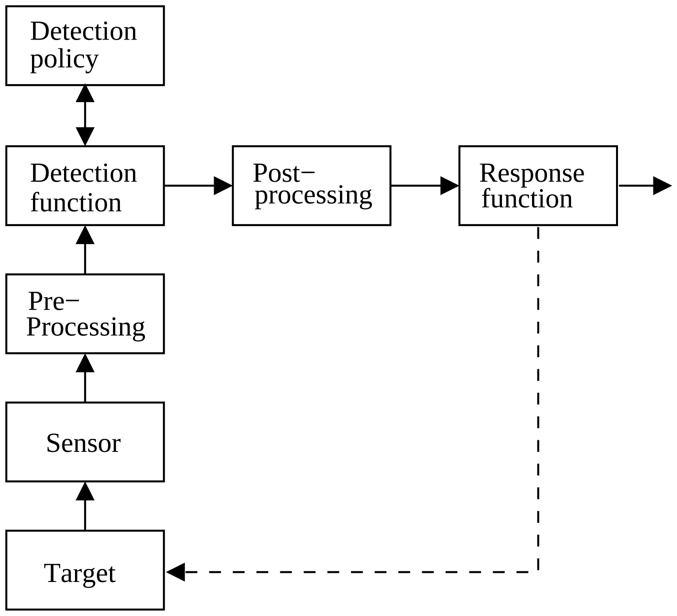

Figure 5: Components of an IDS / FDS

Sensor. The sensor is a device or function that collects data from the target systems to detect fraud and intrusion. A sensor may be a network element providing network packets or an application delivering usage behavior.

Preprocessing. Preprocessing involves filtering and formatting audit data such that they can be used in different analysis (detection) mechanisms. Depending on the sensor and the target, it may be a manual or an entirely automated process.

Detection function. Analysis of collected data is made by the detection function. This uses various techniques such as misuse detection based on simple rules (e.g. pattern matching) or anomaly-based detection (e.g. statistical deviations) in an attempt to find evidence of fraud or intrusion.

Detection policy. The detection policy is defined as the set of rules that together states what events or occurrences are considered authorized versus unauthorized. Examples of policies are rules for access control to targets, misuse signatures and statistical user or system normal behavior.

Post processing. Post processing includes activities such as correlation, classification, prioritization and storage of alarms. For example, correlating alarms is important for identifying related alarms and limiting the number of alarms generated by the detection system.

Response function. The response function is responsible for notifying the proper authority once an alarm is generated. Responses can be passive or active. Passive response functions include logging to a log server or displaying the alarm on an administrative console. Active response functions can be used to terminate the attack or try to limit the damage caused by the attack.

2.3.5 Classification of detection systems

Implementing fraud and intrusion detection systems requires many decisions to be made, decisions about the architecture, the detection methods used, the type of data to be collected and many others. In a survey I made on commercial intrusion detection systems [Kva99] a classification was developed that help comparison between different systems. This aims at identifying properties that lend themselves to comparison. Many classifications of intrusion detection systems have been presented by the research community to help identify, classify and evaluate important properties. The classification presented here is an extension of the work by Axelsson [Axe98] and is summarized in Table 1.

Aspect	*Criteria*
Functional aspects	Granularity of data processing Source of audit data Detection method Response to detected misuse Degree of interoperability Adaptivity Detection capability
Security aspects	Security
Architectural aspects	System organization System and network infrastructure requirements
Operational aspects	Performance
Management aspects	Manageability

Table 1: Classification of comparison criteria

The evaluation criteria are grouped into classes dealing with similar aspects of the detection system.

Functional aspects. The group of functional aspects contains the most extensive collection of criteria. The *granularity of data processing* decides whether input events are processed continuously or in batches, at some regular interval. This affects the response time of the DS and can range from real-time with hard deadlines to batch processing at regular intervals (e.g. once a day, once a week). *Source* *of audit data* refers to the targets that generate the input events. The source of data is usually either network or host based. Network-based data are typically read directly off some multicast network (Ethernet). Host-based data (security or application logs) are collected from hosts distributed throughout the network and can include operating system kernel logs, application program logs and network equipment logs or other host-based security logs. The *detection method* refers to the mechanism or method used to analyze the audit data searching for unauthorized events or behavior. Two different approaches for detecting misuse are commonly used: *rule based* and *anomaly based*. *Response to detected misuse* can be either *passive* or *active*. Passive systems respond by notifying the proper authority. They do not take any measures to prevent or limit the damages caused by an attack. Active systems may not only notify the proper authority but also initiate the necessary countermeasures. These countermeasures often seek to limit the damage inflicted by the attack. The *degree of interoperability* measures the detection system's ability to cooperate with other similar systems. Interoperability can be of interest at various levels in the architecture serving many different purposes, such as exchange of audit data records, detection policies, misuse patterns, user statistics, alarm reports and event notifications. The *adaptivity* of the detection system decides whether it can be adapted to site specific needs with relative ease. The *detection capability* denotes the detection system's capability to recognize a certain type of attack. For example, a detection system analyzing application logs may not be able to detect attacks exploiting vulnerabilities at the network level. At the application level, the network is an unknown abstraction.

Security aspects. The ability of the detection system to withstand attacks against itself is called *security*. This includes its ability to maintain its confidentiality, integrity and availability at all times, even under an active attack against any of its components. We discuss this property in more detail in Paper E.

Architectural aspects. The architecture *(system organization)* of a detection system can be either centralized or distributed. In practice, it may be difficult to categorize a system as strictly centralized or fully distributed, as some subsystems may be centralized while other subsystems are distributed. In many cases, data collection is distributed while data analysis is centralized.

Operational aspects. *Performance*, in this context, means the ability of the DS to meet its deadlines as a real-time system. Primarily two parameters affect the performance of the DS. (1) The **communication overhead** is caused by the distribution of audit data and the communication between the various subsystems of the DS. (2) **Computational overhead** applies mainly to host-based IDS. While network-based ID systems usually run on a dedicated system, host-based IDS execute and collect audit data on the target they monitor. The performance penalty depends greatly on such parameters as granularity of data processing, size and growth rate of system logs, size and complexity of the detection rule base etc.

Management aspects. The *manageability* indicates the system's ability to be managed or to send alarms to dedicated management systems and applications. This is usually a very important criterion in larger companies and organizations.

3 The objective and scope of the thesis

The research presented in this thesis is divided into two parts with different scopes: Part I *(Implementing fraud detection systems)* and Part II *(Protecting Intrusion and fraud detection systems)*. The overall objective of my research is to find security mechanisms and countermeasures that meet the needs of future telecom services. In general, there is not a very great difference between the security needs and requirements of a telecom operator and those of other branches and organizations. Today, the business of a telecom service provider is more or less focused on the Internet and the corresponding security threats and countermeasures. Four of the most important subproblems are:

i) How can we reduce the complexity of our security solutions?

ii) How can we design and build a redundant and fault-tolerant security architecture?

iii) How can we better protect resources and information belonging to our customers?

iv) How can we deploy security to reduce the cost inflicted by intrusion and fraud?

This thesis addresses parts of these research problems. Part I focuses on *iii* and *iv*. Part II focuses on *ii*. Problem *i* is not addressed in my research, but is an important problem and is discussed briefly in the introductory parts.

The overall objective of the research at the Department of Computer Engineering is to model security from a *dependability* perspective in an attempt to find quantitative measures of security, measures that can be used for predictive purposes. Over the years, many problems have been addressed in our research. Below, I mention a few that best fit the framework of my research.

v) What methods can be used to counter threats and thereby improve security? How should these methods be designed, used and implemented in order to be efficient and generic?

vi) How can we design reliable and dependable security services?

vii) How can we make better use of log data from systems and services to find intrusions and fraud?

Problem *v* is rather general but still important. In the thesis, the focus is on detection systems and their role as a countermeasure. In fact, I believe that countermeasures built on *detection* and *response* complement and to some extent replace the need for *preventive* countermeasures. Both Part I and Part II aim to help solve the problems of *v*. Part I addresses the problem of *vii*. Results are provided that show the importance of having a sufficient amount of data for the training and testing of fraud detection systems. The research aims at generating high quality data in high volumes that can be used for this purpose. Problem *vi* is addressed in Part II. Implementing intrusion and fraud detection systems may introduce new risks that must be targeted. The goal of the research was to design a detection system that does not depend on a secure environment for its operation. Specifically, we provide mechanisms to protect confidential policy information in the detection system.

4 Research methodology

4.1 The paradigm of security

A paradigm is a set of ideas and concepts combined with basic assumptions and basic rules (examples) on how to use these ideas and assumptions. Thomas Kuhn, a philosopher of science and author of *The Structure of Scientific Revolutions* [Kuh62], took the idea of paradigms and applied it to the field of science. He uses paradigm to mean the model that scientists hold about a particular area of knowledge. Within a scientific community, defined by the paradigm in practice, the people are "brain-washed" into that paradigm. This is brought about by the continuous education of people by schools, textbooks or whatever means are necessary to convince them to believe in the paradigm. In many ways, this resembles religious groups in society.
So, what is the paradigm of security? I am not sure there is a single answer to this question. Depending upon how we look at security, different paradigms may be in practice at the same time. The paradigm is more or less a framework for the foundations and processes we use to create knowledge within the field. It is also about the means we use to decide what is true and false. In the following section, I will discuss the problem of proving the validity of research in the field of security and put my own research in the light of this discussion.

4.2 Proving the validity or research results

A constant source of debate is how research should be conducted in order to give validity to the results. Let us assume that we are about to prove that a system is *secure*. Formally, we can phrase that problem: *How do we know that the hypothesis defining the component is true?*
Consider the following alternatives. Given a component and its hypothesis *P [P is true]* is indeed true if:

i) P has been carefully tested without being falsified

ii) P is known to be true based on mathematical principles and deductive reasoning

Based on experience and a reasonable "feeling for" the paradigm of security, I would say that these two are the ones used in practice to prove the security of a component. Let us explore these alternatives and see what consequences they will have on security and what conclusions can be drawn from the discussion.

4.2.1 Falsification as a means to prove security

According to falsificationism, theories are shown to be false by observation and experiment. Theories are constructed by guesses or speculation to overcome problems encountered by previous theories. The falsificationist believes that a hypothesis can be verified and theories established by the evidence of observations. Hypotheses that fail to stand up to observational tests are said to be falsified and must be eliminated and replaced by other more innovative conjectures. Only the most fit hypothesis survives these tests and can be regarded as a viable theory. In order to verify a hypothesis it must be falsifiable. That is, it must be possible to test the hypothesis and there must be methods to draw conclusions as to whether the hypothesis survived the tests. A hypothesis must be extensively exposed to falsification. A theory that has survived falsification is said to be *corroborated*. It can never be said of a theory that it is absolutely true, no matter how well it has stood up to the tests.

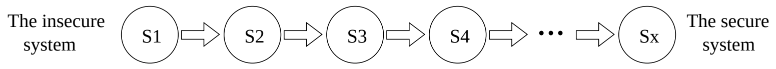

Figure 6: The evolution of a secure system according to the falsificationist

However, it can be said that the current theory is better than its predecessor in the sense that it withstands tests that falsified its predecessors. In Figure 6, a security system, S, consisting of a *set of theories* is recursively tested and falsified. As existing assumptions and theories are falsified, new bold conjectures are proposed. A new theory will be accepted as a worthy candidate if it is more falsifiable than the previous one, especially if it predicts new improved behaviour. The stronger the theory, the more eager one should be to test it further. $Sx[x \Longrightarrow \infty]$ represents the perfectly secure system, or in the general case, the absolute truth (*The* Truth).
It is clear to me that a falsificationist view can never be used to prove the security of a system. Nevertheless, this is in many ways the ruling paradigm of security today. These ideas, first presented by the German philosopher Karl Popper [Pop68], has a number of limitations. For example, the falsification of a theory can be deflected by an ad-hoc modification of the theory. In fact, this is basically what is happening in the security community of today. In the area of security there have been cases when the hypothesis has been falsified, whereby an ad-hoc statement was added to the hypothesis limiting the variety of circumstances under which it holds true. The discovery of weak keys in DES is a typical example [Dav83].
Another problem concerns the validity of the observations. Popper believed that observations are "mini-hypotheses" of their own. Observations must therefore be tested for validity. This gives a circular phenomenon. An observation can therefore never be fully verified. Popper's solution to the problem of circularity was to consider a hypothesis true when the science community decides it to be true. In my thinking mind, this is what the security community is doing today. Many assumptions about security mechanisms in use today have not been verified as true (e.g. RSA and AES). However, they have not been completely falsified either and are accepted by the research community as *probably* secure.

4.2.2 Mathematical principles and deductive reasoning as a means to prove security

Another way of proving the validity of research is to model it mathematically. Proving the security of system may start by proving the security of the atomic components on which the system is built. Usually, in today's paradigm of security, these atomic components are cryptographic mechanisms or mathematical theories. It must be shown that the only way the adversary can break the security is by breaking a ”hard problem” (proof by reduction). By a ”hard problem” we mean a mathematical problem which is known to be, or can be proven to be, hard or impossible to solve[5]. Following the proofs of the atomic components, one can define a formal model using a special logic. Using that logic, we can derive new statements from others using deduction rules. Thereby, the security of composite components can be formally proven on the basis of the assumption that the atomic components are secure. Even though formal proofs are used to some extent in today's research, they are often based on assumptions and hypotheses believed to be true. These beliefs, formed by the paradigm of computer security, are widely accepted as some kind of ”truth”. It is clear that much of the technology available today cannot be formally verified, and systems rather evolve as a result of recursive testing and falsification. Eventually, the system *appears* to be secure as no one has been able to falsify it. In the land of the falsificationist, security by obscurity is king.

4.3 Thesis research methodology

The research programs at the Department of Computer Engineering at Chalmers University often contain elements of empirical studies in an attempt to verify theoretical results. This thesis is no exception. It is trapped in the ruling paradigm of security and makes numerous assumptions that are not necessarily true. The arguments for the validity of my thesis are both theoretical and practical. Empirical studies and tests were made to show the validity of our theoretical arguments[6]. To a large extent, the research was conducted as part of a larger research program, which influenced the results and methodologies I used.

4.3.1 Identifying the research problems

Many of my research problems stem from observations of security problems in telecom environments. Paper A highlights many of these problems and contains examples of existing and future security problems that telecommunication companies are facing. Statistics show that fraud and intrusions are serious problems for telecom operators, which is something that inspired my research. As previously stated, these research problems more or less coincide with those of my de-

[5] A common category of such ”hard-problems” are the problems known to be NP-complete, such as the discrete knapsack problem. The discrete knapsack problem is about finding a set of numbers that add to a specific sum. NP-complete problems are those which are believed not to have a polynomial time solution. [6] Tests that do not falsify my thesis.

partment at Chalmers University of Technology. Both Part I and Part II identify important problems and raise serious questions. All of my publications identify their own sub-problems for which they aim to find solutions. For a complete discussion of the research problems addressed in this thesis, see section 3.

4.3.2 Theoretical studies

The research problems that I have worked on have been studied theoretically in an attempt to find answers and solutions to the problems. In Papers B and D, IP-based services such as video-on-demand applications were studied in order to find means and mechanisms for detecting fraud and intrusions. The studies were purely theoretical as log data from the video-on-demand application were not yet available. This in turn led us to the development of a methodology for generating synthetic log data.

In paper E, distributed intrusion detection architectures were studied to show possible vulnerabilities posing a threat to the target system. Although this was also a pure theoretical study, we applied our hypothesis to existing intrusion detection systems to validate our results.

Papers F and G proposed solutions to deal with problems identified in Paper E. Mechanisms were suggested to protect the confidentiality of the detection policy in a detection system and theoretical arguments were given for its applicability to real systems.

4.3.3 Practical tests and verification of the results

Empirical studies and practical tests were done primarily in Part I. In Paper B, a methodology was developed to generate synthetic log data containing normal and fraudulent behavior. In Paper C, the methodology was used to simulate thousands of users, both normal and fraudulent, which we used for training and testing a neural network detector. Using our synthetic data, we showed that a detector could be trained and tested even when only small amounts of data are available[7].

[7] This may be the case if the service generating data is under development or still has too few users to provide sufficient amounts of data.

5 Summary of appended papers

5.1 Part I: Implementing fraud detection systems

5.1.1 Paper A: New security issues in emerging computing environments - A reflection

Society is becoming increasingly dependent on information and communication systems to provide distributed, networked and instantly available services regardless of location. Users are offered a seamless experience to their services, not only with respect to authentication and access-control but also with respect to services and network access available in their vicinity. The services are ubiquitous and pervasive [Wei91], adaptive and tailored to suit the needs of the user and his current context (See also Paper A).
Unfortunately, this evolution is accompanied by an increased vulnerability to threats. Existing threat models need to be revised to meet new requirements imposed by this scenario. In this position paper we argue that concepts and mechanisms, such as detection and prevention mechanisms (e.g. intrusion detection systems and intrusion prevention system), are essential and will be increasingly important to counter these threats in future IT environments for e-business and e-society. They serve as a powerful complement to, and to some extent, a replacement for conventional preventive security mechanisms. We give real-world examples showing selected security problems and requirements from emerging computing technologies.

5.1.2 Paper B: A Synthetic Fraud Data Generation Methodology

In many cases synthetic data are more suitable than authentic data for testing and training of fraud detection systems. At the same time synthetic data suffer from some drawbacks because they are indeed synthetic and may not have the realism of authentic data. In order to counter this disadvantage, we have developed a method for generating synthetic data that is derived from authentic data. We identify the important characteristics of authentic data and the frauds we want to detect and then generate synthetic data with these properties.

5.1.3 Paper C: Synthesizing test data for fraud detection systems

This paper reports on an experiment aimed at generating synthetic test data for fraud detection in an IP based video-on-demand service. The data generation verifies a methodology previously developed (Paper B) that ensures that important statistical properties of the authentic data are preserved by using authentic normal data and fraud as a seed for generating synthetic data. This enables us to create realistic behavior profiles for both users and attackers. We used the data to train a fraud detection system based on a neural network detector to show the usability and applicability of the synthetic data. The trained system is then exposed to a set of authentic data to measure parameters such as detection capability and false alarm rate and to a corresponding set of synthetic data, and the results are compared.

5.1.4 Paper D: Combining fraud and intrusion detection systems - meeting new requirements -

This paper studies the area of fraud detection in the light of existing intrusion detection research. Fraud detection and intrusion detection have traditionally been two almost completely separate research areas. Fraud detection has long been used by such businesses as telecom companies, banks and insurance companies. Intrusion detection has recently become a popular means to protect computer systems and computer based services. Many of the services offered by businesses using fraud detection are now computer based, thus opening new ways of committing fraud not covered by traditional fraud detection systems. Merging fraud detection with intrusion detection may be a solution for protecting new computer based services. An IP-based telecom service is used as an example to illustrate these new problems and the use of a fraud model is suggested.

5.2 Part II: Protecting intrusion and fraud detection systems

5.2.1 Paper E: Security implications of distributed intrusion detection architectures

This paper addresses the problem of protecting an Intrusion Detection System (IDS) against intrusions as a function of the degree of distribution, i.e. it deals with the problem of the security of the IDS itself. We define three basic Intrusion Detection Architectures (IDAs) denoted *strictly centralized*, *distributed* and *fully distributed* architectures. In the paper we argue that the evolution is proceeding towards more and more distributed systems. One problem is that a higher degree of distribution increasingly exposes systems to tampering and leakage of information. To cope with this problem we suggest a number of fundamental security requirements and discuss the extent to which they may be applied to the architectures and to what benefit. Finally, a few existing intrusion detection systems are analyzed in view of these requirements, showing that there are no systems that completely solve the problem of self-protection.

5.2.2 Paper F: Protecting Security Policies in Ubiquitous Enviroments Using One-Way Functions

This paper addresses the problem of protecting the security-related information in security extensions, e.g. the detection policy in an intrusion detection system or the filtering policy in a firewall. Knowledge of the security policy would potentially facilitate penetration of the target system by an intruder, short-circuit the firewall or circumvent the intrusion detection system's detection mechanisms. In order to avoid this risk we suggest that the policy should be protected using one-way functions and the paper suggests a scheme for protecting stateless policies.

A stateless policy is a policy that takes only the current event into consideration when decisions are made and not the preceding chain of events. Thus, the process of comparing events towards the policy, i.e. making decisions, could be done in much the same way that passwords are hashed and compared in UNIX systems. However, one important distinction is that security policies contain a certain variability that must be taken care of, and a method for this is discussed.

5.2.3 Paper G: Protecting Stateful Security Policies Using One-Way Functions

This paper extends the research in Paper F and addresses the problem of protecting security-related information, such as the detection policy of an intrusion detection system. We suggest a protection scheme to protect stateful policies, which can deal with temporal sequences of events. It can be described by a finite state machine corresponding to regular languages. The protection scheme uses one-way functions and can be seen as a way to provide obfuscation and thus prevent reverse engineering. We provide a complexity analysis of its ability to resist attacks. An example is given to show its applicability.

6 Research contributions

This thesis provides results that are applicable in the fields of both fraud and intrusion detection. Part I contributes to the art of generating synthetic data for testing and training fraud detection systems. Part II contributes to the field of protecting fraud and intrusion detection systems against hostile environments.

6.1 Contributions to the field of synthetic data generation

Having access to realistic data for testing intrusion and fraud detection systems is important and unfortunately few available data sets exist[8]. One of the reasons for this is that live data may contain sensitive information about users in the systems. Information that may be under restriction by laws and regulations or simply embarrassing to the organization from which the data originate. Another reason is the unwillingness of organizations to disclose information about fraud and attacks they are subject to. As a result, few companies or organizations are willing to share live data.
Training and testing fraud detection systems require data with certain properties. Access to authentic data from real services is often preferred although this approach may suffer from the tight coupling to the target system, leading to non-representative data. Our research allows small amounts of live data to be used as seeds for generating large amounts of synthetic data with specific properties. One advantage of such data is that they can be designed to demonstrate properties not available in the authentic data, or even include attacks expected in the future. This makes special experiments possible, e.g. stress tests. Further, one may want to train and test a detection system prototype adapted to a certain target environment before it is put into operation. In that case, the use of artificial data is the only possible approach. Finally, synthetic data may be generated to cover extensive periods of time, which could have taken months or years to collect from the real target system. A large number of users can also be simulated, even if very few users currently use the service. This makes it possible to make scalability tests.

6.2 Contributions to the field of protecting intrusion and fraud detection systems

Deploying intrusion and fraud detection systems may introduce new threats to the target environments they aim to protect. We show that the architecture of the detection system has a great impact on its security. The security of the detection system itself is important for several reasons. First, a compromised detection system must be considered corrupt and can no longer produce reliable output. Second, and more important, we show that a compromised detection system may impose additonal threats towards the target system, as leakage of sensitive information may result. The information may be misused by an intruder to gain

[8] See the *Related work* section for references to existing data sets.

knowledge about the target systems (e.g. weaknesses, protocols used, security policies etc.). Such knowledge can be used by intruders to bypass the detection system in an attempt to gain unauthorized access to a service or information system. In addition, we present mechanisms based on one-way functions that can be used to remedy parts of these problems.

7 Related work

The papers included in this thesis contain references to previous work related to the topic and research problem of each paper. In this introduction, I will give a brief summary of related work and other work that I believe is seminal, or to work not mentioned in my papers. Further, I will give references to research that is relevant to my thesis but did not fit any of the topics addressed in my papers. Some of my papers were published many years ago and additional research related to my thesis may have been published to which I like to show my respect. Finally, I will briefly mention relevant PhD dissertations in the field of fraud and intrusion detection.

7.1 Fraud detection

First I would like to mention that published research in the area of fraud detection is difficult to find. Some good work has indeed been done, but the feeling one gets from reading papers in the field is that much research is confidential and conducted within companies. However, some interesting research are available for public scrutiny. Fraud detection has been applied to many different fields and branches. Davis and Goyal [DG93] used knowledge based methods to find fraud in cell phone systems. They observed that using an average user as a basis for threshold detection was problematic. Instead, they modeled every user individually and used a profile that could adapt over time. Burge et al. [BSTM$^+$97] used unsupervised learning to detect fraud in sequences of call records. Moreau et al [MB97] used supervised feed-forward neural networks to detect fraud in mobile telecommunication systems. This is similar to the detection schemes we used in Paper C. They argue that detection of excessive usage is problematic due to the fact that the most profitable customers are indeed high-volume users. Fawcett and Provost [FP96][FP97] used rule-based methods to detect fraud. They used call records from the cell phone system and identified indicators of fraud using adaptive rule sets. Research was done on intelligent techniques for the detection of telecom fraud in the Eurescom project P1007. The project evaluated various ”intelligent techniques”, such as data mining and neural networks for the purpose of detecting fraud in cell phone and IP-based services [wg00a]. For the reader interested in telecom fraud detection, a good start is Collin’s series of articles on telecom fraud, which can be found in [Col99a][Col99b][Col00]. A noteworthy journal article on telecom fraud was written by Hoath [Hoa98].

Detection of credit card fraud was treated by Leonard [Leo93]. He deploys expert systems to find fraudulent uses of credit cards. Stolfo et al. [SFL$^+$97] uses meta-learning techniques to combine results from multiple classifiers to find indications of credit card fraud. Chan and Stolfo explore non-uniformity of class distributions in credit card fraud detection [CS98]. Neural network models for credit card fraud detection were presented by Dorronsoro et al.[DGSC97], Ghosh and Reilly [GR94], Hanagandi et al. [HDB96] and Brause et al. [BLH99].

Fraud detection shares many features with intrusion detection systems. Most research in anomaly-based intrusion detection is applicable to fraud detection as well. However, there seems to be little cross-referencing between the two fields. Intrusion detection has its origin in computer systems and networks whereas fraud detection stems from fields such as banking, telecom and insurance. In [KLJ00] we discuss several similarities between fraud and intrusion detection systems and list important work applicable to both fields. Especially useful is research on various detection methods such as AI, data mining, neural networks, case based reasoning etc. There are also statistical methods for user profiling and combinations of several methods [GC94]. A study of methods applicable for both fields of research was conducted by Debar et al. [DDW99]. Fawcett and Provost [FP99] applied their fraud detection model to the problem of intrusion detection. They used their anomaly detection engine in an attempt to analyze UNIX commands taken from 8000 login sessions from 77 users. They reflect that, at a superficial level, detecting intrusions is very similar to detecting cell phone fraud. However, they experienced problems, as there was no distinguishable intruder behavior. They conclude that, in the fraud detection domain, fraudulent behavior is often quite different from legitimate behavior, which was not the case in the UNIX command data they used. For that purpose, profiling users should be more important.

7.2 Synthetic data generation

Research on synthetic data for fraud detection applications is not frequently published. However, there are some interesting research results. Aleskerov et al. [AFR97] used synthetically generated data to detect credit card fraud. They used a *transaction generator* to generate purchase transactions. Three inputs were used to control the characteristics of the synthetic data: the category of purchase, the typical amount of money spent and the time since the last purchase of the same category. Statistical distributions were applied to provide the information about the amount and time during generation. The tools they developed could utilize any possible distribution, although they limited their tests to Gaussian distributions. They use a three-layer neural network model for training and detection. They showed a fraud detection rate of 85% and no false positives, which was satisfactory for their application. The data generation tests in this thesis use a more complex user model as the ”transactions” for video-on-demand applications contain more parameters that must be modeled. Synthetic data are not commonly used in the fraud detection area, although the use of manipulated authentic data is discussed in some papers, e.g. [CFPS99] and [BSTM$^+$97]. In the intrusion detection area, more work has been done using synthetic test data. Huge amounts of synthetic test data were generated in the 1998 and 1999 DARPA intrusion detection evaluations [LFG$^+$00][HLF$^+$01]. The test data they generated contained network traffic and system call log files (SUN BSM log files) from a simulated large computer network. Both attacks and background data were generated synthetically, but the background data were said to be similar to sampling data from a number of Air Force bases. Mainly, the background data was generated using software automata that simulated the usage of different services. Data simulating attacks were generated using attack scripts based on the work of Kendall et al. [Ken99]. Some more complicated background data and attacks were injected live into the system. Unfortunately, no extensive analysis of the quality of the generated data has been made. Some efforts in further automating data generation are described in [HRLC01]. McHugh [McH00] criticizes the lack of validation of test data in the 1998 DARPA evaluation. The process of generating data is vaguely described, and it is difficult to determine the quality of the data. Some improvements were made but many of these issues are still not fully addressed. Debar et al. [DDWL98] developed a generic intrusion detection test bed and suggested the use of a finite state automata to simulate user behavior. While this methodology is rather close to our work in papers B and C, they did not use this method in their test bed. They declared it practical only if the set of user commands is limited. Instead, they used recorded live data from user sessions. Puketza et al. [PZC$^+$96] describe a software platform for testing intrusion detection systems where they simulate user sessions using the UNIX package *expect*. Some interesting work has been done on measuring characteristics in data and how they affect the detection systems, e.g. Lee and Xiang [LX01] and Tan and Maxion [TM03]. In [MT00], Maxion and Tan generate ”random” synthetic data with different degrees of regularity and show that it affects the false alarm rate drastically.

7.3 Security of intrusion detection systems

The reliability of intrusion and fraud detection systems is important for their trustworthiness. In this thesis, we argue that the architecture has an impact on the security and, hence, the trustworthiness of a detection system. The majority of the research on security and detection systems has focused on the security of the target systems. However, there are a few published articles that address the security of the detection system itself. In the work of Debar et al. [DDW99], *fault tolerance* is introduced as a property that addresses an intrusion detection system's ability to resist attacks. Axelsson [Axe98] also identifies the lack of research in this field and introduces *security* as the *”the ability of the system to withstand hostile attacks against the system itself”*. In the work of Ptcaek et al. [PN98], more practical results are demonstrated to show how the security of the target systems can be jeopardized if the confidentiality of the policy database is lost. They showed how knowledge about the detection function can be used to circumvent detection of an attack. Handley et al. [HPK01] demonstrated similar results. Later, Wagner and Soto [WS02] showed how anomaly based intrusion detection systems could be subject to attack. They introduced the notion of a mimicry attack, which allowed an attacker to cloak his intrusion to avoid detection by the IDS. A mimicry attack mimic the behavior of the systems but with a ”malicious twist” that prevents the detection systems from triggering an alarm. Forrest et al. [FPAC94] use

an innovative approach inspired by human immunology to protect computer systems. They describe the problem of protecting computer systems as the problem of learning to distinguish self from other. The ideas are based on the generation of T cells in the human immune system.
Neumann [Neu95] identified an extensive set of requirements for tamper proofing NIDES [Lab95]. He suggests that tamper proofing can be achieved by fulfilling a series of goals related to the authenticity, integrity and confidentiality of the analysis system (NIDES) and its components. Subsystem encapsulation is suggested for protecting the audit data and the analysis subsystem's rulebase. To prevent reverse engineering of the detection policy (rule base), Neumann proposes the use of encryption.

The problem of protecting policies (papers F and G) is related to the problem of protecting software / code from the influence of malicious environments where they may be subject to reverse engineering. The issue of protecting security policies against unauthorized disclosure is closely related to the reverse engineering problem. Several different techniques have been proposed to tackle the reverse engineering problem, and an overview is given by Collberg et al. [CT00]. Several researchers have suggested using encryption to hide the execution of a program [HP87]. However, many such systems require hardware for encryption / decryption process, which limits the applicability. Encrypted evaluation of functions was demonstrated by Sander and Tschudin [ST98a][ST98b]. They demonstrated how polynomial functions could be computed using a software solution only in a hostile environment. The limitation of polynomial functions was later extended to all functions computable by circuits of logarithmic depth [SYY99]. Obscuring the code through obfuscation to prevent attack is another means of protection. Program obfuscation is a semantics-preserving transformation technique that aims to make the target application ”unreadable”. Extensive research has been conducted in this area and an overview is given in [CT00]. Later, Barak et al. proved the impossibility of achieving program obfuscation under certain conditions. They showed that a software virtual black box generator that can prevent reverse engineering of program code, except from that of its input and output, does not exist [BGI$^+$01]. In practice, this is of less concern, as applications for practical obfuscation schemes would still be useful in many situations and under less strict conditions as pointed out by Oorschot [vO03].

Fault-tolerant techniques were used in the MAFTIA project to build dependable systems. These authors showed how an intrusion-tolerant intrusion detection system could be designed to provide a secure service despite the presence of malicious faults [De02]. However, their work focused on the system and architectural level rather than on mechanisms and protocols.

Research in the field of secure multi-party computation has addressed related problems. Canetti and Halevi [CH98] showed how to maintain authenticated communication even in the presence of break-ins. Their method allows a certain degree of resilience against attackers to protect the *integrity* of authenticated communication. Goldreich et al. [GMW87] demonstrated how ”mental games” could be played between parties without leaking any information about the actual computation. Neither of these techniques addresses the same problems brought up in this paper, although they all share the same goal of keeping computations and information private.

7.4 Recent PhD dissertations in fraud and intrusion detection

This section lists PhD dissertations that I have found in the field of fraud and intrusion detection. I have limited the scope to cover work published during the last few years.

7.4.1 Fraud detection

Lundin Barse. The work by Lundin Barse [Bar04] deals with problem of having good quality input data for training, testing and operating intrusion and fraud detection systems. In part one of her thesis, a comparison is made between intrusion and fraud detection systems. A survey on research in intrusion detection systems is presented in an attempt to identify new areas where progress is needed. Part two is devoted to improving log data used for detecting intrusions and fraud. The third part deals with privacy issues in intrusion detection systems and suggests methods for anonymizing log data.

Ohta. In a thesis by Ohta [Oht03], a study of how the forensic type phase in auditing affects audit risk when using a stylized game theoretic analysis. The forensic type phase is the phase in which the auditors try to detect fraud. The main result of the thesis is that introducing a forensic type phase does not always reduce audit risk as the auditors themselves can be fraudulent.

Hollmén. A recent doctoral thesis on fraud detection, Hollmén [Hol00] summarized his work on fraud in mobile communication systems. He presented methods to detect fraud using partially labeled data. Partially labeled data may exist when it is known that a certain subscriber is fraudulent, but it is not known exactly when in time. He detected fraud using probabilistic models and neural networks and showed how uncertainties in the fraud domain could be addressed.

7.4.2 Intrusion detection

Uppuluri. In the dissertation of Uppuluri [Upp03], detection and response mechanisms are explored based on a specification-based approach. Attacks are detected as deviations from a specified behavior by an enforcement algorithm based on REE (Regular Expressions over Events) and EFA (Extended Finite Automata). A prototype for the UNIX operating system was implemented were sequences of system calls were analyzed.

Pikoulas. Pikoulas [Pik03] argues that it is impossible to fully protect a computer network from intrusions, especially in target environments where the behavior of the users change over time. He uses Bayesian statistics to evaluate user behavior and tries to predict future behavior of users. The thesis proposes an agent-based intrusion detection architecture. Communication agents are suggested as a tool to make the security agents more efficient in gathering information from other agents.

Mahoney. In the thesis by Mahoney [Mah03], contributions are made to the field of network anomaly detection. He models network protocols from the data link layer to the application layer in order to detect attacks on vulnerabilities in the implementation of these protocols. A machine learning approach is used for that purpose. The tools and methods developed in his research were tested on the 1999 DARPA / Lincoln Laboratory intrusion detection evaluation set.

Bivens. Bivens proposes a distributed framework for network management middleware. In his thesis [II03] distributed functional agents are used to carry actions on behalf of the management application. The framework he developed was used to deploy network-based and host-based intrusion detection in distributed environments. In his work, learning techniques such as perceptron-based neural networks, self-organizing maps and genetic algorithms were used by the detection mechanisms. A comparison was made with traditional architectures of network management.

Bhargavan. This dissertation [Bha03] demonstrates and evaluates passive run-time monitoring techniques to test black-box implementations of network protocols and their conformance to published specifications. His results are of interest for network intrusion detection systems where systems are suceptible to false positives / negatives due to incorrect or incomplete protocol modeling. A new passive protocol monitoring framework is introduced that provides analysis tools to combat such deficiencies.

Tan. Tan [Tan02] addresses the issue of evaluating anomaly detection systems. A framework for anomalies and fault-injection into categorical data was developed. She demonstrates that the ability to detect attacks differs significantly amongst anomaly detectors. The results refute the hypothesis that all anomaly detectors are equally capable of detecting attacks or intrusions.

Toth. Toth [Tot03] deals with three diffrent problems related to intrusion detection: (1) using clustering techniques to parallelize event analysis and thereby increase the performance of the detection engine; (2) using abstract signatures to detect entire classes of attacks allowing variations of known attacks; (3) estimating the affect of response actions on the usability of the network.

Kruegel. Kruegel's dissertation [Kru02] introduces the concept of *Network Alertness*. The detection process is realized through a collaboration of nodes that correlate and assemble pieces of evidence in the victim's network. His approach allows nodes to collaborate in a point-to-point fashion, providing improved scaleability and fault tolerance compared to existing solutions. Decentralized algorithms for pattern detection were implemented to allow collaboration in distributed environements.

Kim. Kim [Kim02] uses ideas from the human immune systems and applies them to intrusion detection. He investigates three different evolutionary algorithms and demonstrates that it is possible to design an intrusion detection system that can adapt to continuously changing environments, dynamically learing the pattern of "self", and detecting new patterns of "non-self".

Seleznyov. In this thesis by Seleznyov [Sel02], an anomaly intrusion detection system was considered in terms of modeling computer users. He used machine learning techniques to discover temporal and sequential regularities in user behavior. His results were empirically verified on real computer users.

Zamboni. The work of Zamboni [Zam01] introduces the concept of using internal sensors to perform intrusion detection. An architecture called ESP is developed that allows the implementation of embedded detectors for the purpose of achieving localized data reduction. He shows that both *specific* and *generic* detectors can be implemented. The former are specialized for certain types of intrusions whereas the latter are able to detect different types of intrusions. Performance testing of the ESP implementation showed the feasibility of the approach.

Julisch. A thesis by Julisch [Jul01] addresses the problem of dealing with false alarms in intrusion detection systems. The basic idea is that every false alarm occurs for a reason, *the root cause*. He shows that most false alarms can be related to only a few root causes. On the basis of this observation, a new paradigm for alarm handling is proposed using alarm clustering techniques. Experiments indicate that an average of 70% of the false alarms can be eliminated.

Ning. Ning [Nin01] uses abstractions to design generic intrusion detection models. He presents a hierarchial model to support attack specification and event abstraction in distributed intrusion detection. The model allows generic signatures that can accommodate variations of knowns attacks. An experimental system *CARDS* was implemented to test the approach.

Lane. Lane [Lan00] investigated in his thesis machine learning issues in the domain of anomaly detection. Instance-based models and hidden-Markov models are used to analyze command line data from users. He show that users can roughly be divided into behavioral classes related to their experience level.

Lindqvist. Lindqvist [Lin99] investigated the fundamentals of analysis and detection of computer misuse. He analysed and categorized attacks and suggested an intrusion detection library format for sharing intrusion detetction information among IDS developers.

8 Conclusions and directions for future research

The thesis identifies and discusses some important issues that must be targeted when implementing intrusion and fraud detection systems. We showed the importance of test data for training, testing and evaluating detection systems. We developed and demonstrated a methodology for generating synthetic data that provides designers and implementers with a new tool in their struggle to build better detection systems. Our work focused on fraud detection applications. However, future work will aim to bring these results to the field of intrusion detection.

Our work also brings the fields of intrusion detection and fraud detection closer together. We have conducted research in the border between the two fields in an attempt to identify synergies, similarities and differences. We showed the advantages of combining intrusion and fraud detection to increase the efficiency and accuracy of detection. We genuinely believe that cross-border research would help develop both fields, and our work provides a starting point for future work in this direction.

The thesis identifies and discusses some important security issues that must be addressed when deploying a misuse detection capability. Although we suggested methods and techniques to identify and remedy some of these vulnerabilities, many security issues still remain unsolved. Future research in the field of software protection and prevention of reverse engineering will hopefully solve many of the remaining problems.

In conclusion, security vulnerabilities have existed since the beginning of computer programming and we must accept the idea that they probably always will. Existing and commonly used security techniques such as authentication, access control and encryption are prime targets for attackers, and history shows that they may fail for reasons undreamed of by their creators. Intrusion and fraud detection systems provide a second layer of defense and play an important role in securing today's and future computer systems.

References

[AFR97] E. Aleskerov, B. Freisleben, and B. Rao. CARDWATCH: A Neural Network based Database Mining System for Credit Card Fraud Detection. In *Proceedings of the 1997 IEEE/IAFE Conference on Computational Intelligence for Financial Engineering*. IEEE Press, 1997.

[And80] James P. Anderson. Computer security threat monitoring and surveillance. Technical report, James P Anderson Co., Box 42, Fort Washington, PA 19034, USA, April 15 1980.

[Axe98] Stefan Axelsson. Research in intrusion detection systems: A survey. Technical Report 98-17, Department of Computer Engineering, Chalmers University of Technology, SE-412 96 Göteborg, Sweden, December 15 1998. Revised August 19, 1999.

[Bar04] E. Lundin Barse. *Logging for intrusion and fraud detection*. PhD thesis, Chalmers University of Technology, Göteborg, Sweden, 2004.

[BGI$^+$01] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (Im)possibility of Obfuscating Programs. *Lecture Notes in Computer Science*, 2139, 2001.

[Bha03] K. Bhargavan. *Network event recognition*. PhD thesis, University of Pennsylvania, 2003.

[BLH99] R. Brause, T. Langsdorf, and M. Hepp. Neural Data Mining for Credit Card Fraud Detection. In *Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence*, pages 103–106, 1999.

[BSTM$^+$97] Peter Burge, John Shawe-Taylor, Yves Moreau, Bart Preneel, Christof Stoermann, and Chris Cooke. Fraud Detection and Management in Mobile Telecommunications Networks. In *Proceedings of the European Conference on Security and Detection ECOS 97*, pages 91–96, London, April 28-30 1997. ESAT-SISTA TR97-41.

[CFPS99] Philip K. Chan, Wei Fan, Andreas L. Prodromidis, and Salvatore J. Stolfo. Distributed Data Mining in Credit Card Fraud Detection. *IEEE Intelligent Systems*, 14(6), Nov / Dec 1999.

[CH98] R. Canetti and S. Halevi. Maintaining Authenticated Communication in the Presence of Break-ins. *Journal of Cryptology: the journal of the international Association for Cryptologic Research*, 1998.

[Col99a] Michael Collins. Telecommunications Crime - Part 1. *Computers & Security*, 18:577–586, 1999.

[Col99b] Michael Collins. Telecommunications Crime - Part 2. *Computers & Security*, 18:683–692, 1999.

[Col00] Michael Collins. Telecommunications Crime - Part 3. *Computers & Security*, 19:141–148, 2000.

[Com03] Computer Security Institute and Federal Bureau Of Investigation. 2003 CSI/FBI Computer Crime and Security Survey. 2003. Abridged version at http://www.gocsi.com.

[CS98] Philip K. Chan and Salvatore J. Stolfo. Toward scalable learning with nonuniform class and cost distributions: A case study in credit card fraud detection. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 164–168, 1998.

[CT00] Christian Collberg and Clack Thomborson. Watermarking, Tamper-Proofing, and Obfuscation - Tools for Software Protection. 2000.

[cU02] National High Tech crime Unit. Hi-Tech Crime - the impact on UK business. http://www.nhtcu.co.uk, December 2002.

[Dav83] D. W. Davies. Some Regular Properties of the DES. *Advances in Cryptology: Proceedings of Crypto 82, Plenum Press*, pages 89–96, 1983.

[DDK96] Marc Dacier, Yves Deswarte, and Mohamed Kaâniche. Quantitative Assessment of Operational Security: Models and Tools. Technical Report, LAAS Report 96493, May 1996, 1996.

[DDW99] H. Debar, M. Dacier, and A. Wespi. Towards a Taxonomy of Intrusion Detection Systems. *Computer Networks*, 31(8):805–822, April 1999.

[DDWL98] H. Debar, M. Dacier, A. Wespi, and S. Lampart. An Experimentation Workbench for Intrusion Detection Systems. Technical Report RZ2998, IBM Research Division, Zurich Research Laboratory, Zurich, Switzerland, mar 1998.

[De02] M. Dacier (editor). Design of an Intrusion-Tolerant Intrusion Detection System, Deliverable D10. MAFTIA European Project IST-1999-11583, IBM Zurich Research Laboratory, 2002.

[DG93] A. Davies and S. Goyal. Management of cellular fraud: Knowledge-based detection, classification and prevention. *Proceedings of Thirteenth International Conference on Artificial Intelligence, Expert Systems and Natural Language*, 1993.

[DGSC97] J. R Dorronsoro, F. Ginel, C. Sánchez, and C. S. Cruz. Neural fraud detection in credit card operations. *IEEE Transactions on Neural Networks*, 8(4):827–834, 1997.

[fOPotEC91] Office for Official Publications of the European Communities. Information Technology Security Evaluation Criteria. Technical Report Version 1.2, June 1991.

[fOPotEC99] Office for Official Publications of the European Communities. Directive 1999 / 93 / EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures, 1999.

[fOPotEC02] Office for Official Publications of the European Communities. Directive 2002 / 58 / EC of the European Parliament and of the Council of 12 July 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector (Directive on privacy and electronic communications), 2002.

[FP96] Tom Fawcett and Foster Provost. Combining Data Mining and Machine Learning for Effective User Profiling. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 8–13, 1996.

[FP97] Tom Fawcett and Foster Provost. Adaptive Fraud Detection. In *Journal of Data Mining and Knowledge Discovery 1(3)*, pages 291–316, 1997.

[FP99] Tom Fawcett and Foster Provost. Activity monitoring: Noticing interesting changes in behavior. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 53–62, 1999.

[FPAC94] Stephanie Forrest, Alan S. Perelson, Lawrence Allen, and Rajesh Cherukuri. Self-Nonself Discrimination in a Computer. In *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy*, pages 202–212, Oakland, CA, 1994. IEEE Computer Society Press.

[FPSSE96] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy (Eds.). *Advances in Knowledge Discovery and Data Mining*. AAAI Press / The MIT Press, 1996.

[GC94] B. Garner and F. Chen. Hypothesis generation paradigm for fraud detection. In *Proceedings of TENCON '94, IEEE Region 10's Ninth Annual International Conference.*, 1994. Theme: Frontiers of Computer Technology.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play ANY mental game. In *Proceedings of the 19th Annual ACM conference on Theory of computing*, pages 218–229. ACM Press, 1987.

[Gor03] Dan Gorton. Extending Intrusion Detection with Alert Correlation and Intrusion Tolerance. Technical Report 27L, Göteborg, Sweden, 2003. Licentiate thesis.

[GR94] S. Ghosh and D. L. Riley. Credit card fraud detection with a neural network. In *Proceedings of Twenty-Seventh Hawaii International Conference on System Sciences*, pages 621–630. IEEE Computer Society Press, 1994.

[HB95] Lawrence R. Halme and Kenneth R. Bauer. AINT misbehaving - a taxanomy of anti-intrusion techniques. *Proceedings of the 18th National Information Systems Security Conference*, pages 163–172, October 1995.

[HDB96] V. Hanagandi, A. Dahr, and K. Buescher. Density-based clustering and radial basis function modeling to generate credit card fraud scores. In *Proceedings of the IEEE/IAFE 1996 Conference o Computational Intelligence for Financial Engineering (CIFEr)*, pages 247–251. IEEE Press, 1996.

[HLF$^+$01] Joshua W. Haines, Richard P. Lippmann, David J. Fried, Eushiuan Tran, Steve Boswell, and Marc A. Zissman. 1999 DARPA Intrusion Detection System Evaluation: Design and Procedures. Technical Report Technical Report 1062, MIT Lincoln Laboratory, February 2001.

[HLJ99] Hans Hedbom, Stefan Lindskog, and Erland Jonsson. Risks and Dangers of Security Extensions. In *Proceedings of the Security and Control of IT in society-II (IFIP SCITS-II)*, pages 231–248, Bratislava, Slovakia, June 15-16 1999.

[Hoa98] Peter Hoath. Telecoms fraud, the gory details. Computer Fraud & Security. *Computers & Security*, 20(1):10–14, 1998.

[Hol00] Jaakko Hollmén. *User profiling and classification for fraud detection in mobile communications networks*. PhD thesis, Helsinki University of Technology, Espo, Finland, 2000.

[HP87] Amir Herzberg and Shlomit S. Pinter. Public protection of software. *ACM Transactions on Computer Systems, 5(4)*, pages 371–393, 1987.

[HPK01] Mark Handley, Vern Paxson, and Christian Kreibich. Network intrusion detection: evasion, traffic normalization, and end-to-end protocol semantics. *USENIX Security Symposium (Washington, DC, 13–17)*, August 2001.

[HRLC01] Joshua Haines, Lee Rossey, Rich Lippmann, and Robert Cunnigham. Extending the 1999 Evaluation. In *Proceedings of DISCEX 2001*, Anaheim, CA, June 11-12 2001.

[II03] J. A. Bivens II. *Distributed framework for deploying machine learning in network management and security*. PhD thesis, Rensselaer Polytechnic Institute, Troy, New York, 2003.

[Jon97] Erland Jonsson. A Quantitative Approach to Computer Security from a Dependability Perspective. 1997.

[Jul01] K. Julisch. *Using Root Cause Analysis to Handle Intrusion Detection Alarms*. PhD thesis, University of Dortmund, West Lafayette, Indiana, USA, 2001.

[Ken99] Kristopher Kendall. A database of computer attacks for the evaluation of intrusion detection systems. Master's thesis, MIT, 1999.

[Kim02] J. W. Kim. *Integrating Artificial Immune Algorithms for Intrusion Detection*. PhD thesis, University College London, 2002.

[KLJ00] Håkan Kvarnström, Emilie Lundin, and Erland Jonsson. Combining fraud and intrusion detection - meeting new requirements. In *Proceedings of the fifth Nordic Workshop on Secure IT systems (NordSec2000)*, Reykjavik, Iceland, October 2000.

[Kru02] C. Kruegel. *Network Alertness - Towards an adaptive, collaborating Intrusion Detection System*. PhD thesis, Technical University of Vienna, 2002.

[Kuh62] Thomas Kuhn. *The Structure of Scientific Revolutions*. University of Chicago Press, 1962.

[Kva99] Håkan Kvarnström. A survey of commercial tools for intrusion detection. Technical Report 99-8, Department of Computer Engineering, Chalmers University of Technology, SE-412 96 Göteborg, Sweden, 1999.

[Lab95] SRI Computer Science Laboratory. Next-generation Intrusion Detection Expert System (NIDES) - A Summary. Technical report, 1995.

[Lan00] T. Lane. *Machine Learning Techniques for the Computer Security Domain of Anomaly Detection*. PhD thesis, Purdue University, West Lafayette, IN, USA, 2000.

[Leo93] K. J. Leonard. Detecting credit card fraud using expert systems. *Computers and Industrial Engineering*, 25(1-4):103–106, 1993.

[LFG$^+$00] Richard P. Lippman, David J. Fried, Isaac Graaf, Joshua W. Haines, Kristopher R. Kendall, David McClung, Dan Weber, Seth E. Webster, Dan Wyschogrod, Robert K. Cunningham, and Marc A. Zissman. Evaluating Intrusion Detection Systems: The 1998 DARPA

Off-line Intrusion Detection Evaluation. In *DISCEX 2000*. IEEE Computer Society Press, January 2000.

[Lin99] Ulf Lindqvist. *On the fundamentals of analysis and detection of computer misuse*. PhD thesis, Chalmers University of Technology, Göteborg, Sweden, 1999.

[LX01] Wenke Lee and Dong Xiang. Information-Theoretic Measures for Anomaly Detection. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, May 2001.

[Mah03] M. V. Mahoney. *A machine learning approach to detecting attacks by identifying anomalies in network traffic*. PhD thesis, Florida Institute of Technology, Melbourne, Florida, 2003.

[MB97] Yves Moreau and Peter Burge. Fraud Detection in mobile communications networks using supervised neural networks. In *Proceedings of SNN'97, Europe's Best Neural Networks Practice*, 1997.

[McH00] John McHugh. The 1998 Lincoln Laboratory IDS Evaluation: A Critique. In *Recent Advances in Intrusion Detection, Third International Workshop, RAID 2000, Toulouse, France, 02-04 October 2000, pages*, pages 145–161, Toulouse, France, October 2-4 2000. Lecture Notes in Computer Science #1907, Springer-Verlag, Berlin.

[MdVHC03] Craig Mundie, Pierre de Vries, Peter Haynes, and Matt Corwine. Trustworthy Computing. Technical report, Microsoft Corporation, 2003.

[MT00] Roy A. Maxion and Kymie M.C. Tan. Benchmarking Anomaly-Based Detection Systems. In *International Conference on Dependable Systems and Networks*, New York, New York, June 2000. IEEE Computer Society Press.

[Neu95] P. G. Neumann. Architectures and formal representations for secure systems. Technical report, Final Report; SRI Project 6401; Deliverable A002, 1995.

[Nin01] P. Ning. *Abstraction-based Intrusion Detection in Distributed Environments*. PhD thesis, George Mason University, Fairfax, VA, USA, 2001.

[Oht03] Y. Ohta. *The forensic-typ phase: A game-theoretic analysis of fraud detection in auditing*. PhD thesis, State University of New York at Buffalo, 2003.

[Pik03] J. Pikoulas. *An Agent-based Bayesian Method for Network Intrusion Detection*. PhD thesis, Napier University, Edinburgh, England, 2003.

[PN98] Thomas H. Ptacek and Timothy N. Newsham. Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection. Technical report, Suite 330, 1201 5th Street S.W, Calgary, Alberta, Canada, T2R-0Y6, 1998.

[Pop68] Karl R. Popper. The Logic of Scientific Discovery, 1968. London: Hutchinson.

[PZC$^+$96] Nicholas J. Puketza, Kui Zhang, Mandy Chung, Biswanath Mukherjee, and Ronald A. Olsson. A methodology for testing intrusion detection systems. *Software Engineering*, 22(10), 1996.

[Sch00] Bruce Schneier. *Secret & Lies - Digital Security in a Networked World*. John Wiley & Sons, Inc, 2000.

[Sel02] A. Seleznyov. *An Anomaly Intrusion Detection System Based on Intelligent User Recognition*. PhD thesis, University of Jyväskylä, Finland, 2002.

[SFL$^+$97] Salvatore Stolfo, Wei Fan, Wenke Lee, Andreas Prodromidis, and Philip Chan. Credit Card Fraud Detection Using Meta-Learning: Issues and Initial Results. In *AAAI Workshop on AI Approaches to Fraud Detection and Risk Management*, Providence, Rhode Island, July 1997.

[SS75] Jerome H. Saltzer and Michael D. Schroeder. The protection of Information Systems. *Proceedings of the IEEE, 63(9),* pages 1278–1308, September 1975.

[ST98a] Tomas Sander and Christian F. Tschudin. On Software Protection via Function Hiding. *Lecture Notes in Computer Science*, 1525:111–123, 1998.

[ST98b] Tomas Sander and Christian F. Tschudin. Protecting Mobile Agents Against Malicious Hosts. *Lecture Notes in Computer Science*, 1419:44–??, 1998.

[SYY99] Tomas Sander, Adam Young, and Moti Yung. Non-Interactive CryptoComputing For NC 1. In *IEEE Symposium on Foundations of Computer Science*, pages 554–567, 1999.

[Tan02] K. M. C. Tan. *Defining the Operational Limits of Sequence-based Anomaly Detectors*. PhD thesis, University of Melbourne, Victoria, Australia, 2002.

[TM03] Kymie M. C. Tan and Roy A. Maxion. Determining the Operational Limits of an Anomaly-Based Intrusion Detector. *IEEE Journal on Selected Areas in Communication*, 21(1), January 2003.

[Tot03] T. Toth. *Improving Intrusion Detection Systems*. PhD thesis, Technical University of Vienna, 2003.

[Tun99] Brian Tung. The Common Intrusion Detection Framework (CIDF), Juni 22 1999. http://gost.isi.edu/cidf.

[Upp03] P. Uppuluri. *Intrusion detection/prevention using behavior specifications*. PhD thesis, State University of New York at Stony Brook, 2003.

[vO03] P. C. van Oorschot. Revisiting Software Protection. *6th International Conference on Information Security*, 2003.

[Wei91] Marc Weiser. The computer for the 21st Century. *Scientific American 265(3)*, pages 66–75, 1991.

[wg00a] Eurescom P1007 working group. Application of Intelligent Techniques to Telecommunications Fraud Detection - Findings and Recommendations on Intelligent Techniques to Telecommunications Fraud Detection and Analysis. Technical report, Eurescom P1007, 2000. Distribution limited to Eurescom members.

[wg00b] Eurescom P1007 working group. Application of Intelligent Techniques to Telecomuncations Fraud Detection - New Techniques (Deliverable 1, Annex C). Technical report, Eurescom P1007, 2000. Distribution limited to Eurescom members.

[WS02] D. Wagner and P. Soto. Mimicry attacks on host based intrusion detection systems. In *Proceedings of the Ninth ACM Conference on Computer and Communications Security*, 2002.

[Zam01] D. Zamboni. *Using Internal Sensors for Computer Intrusion Detection*. PhD thesis, Purdue University, West Lafayette, IN, USA, 2001.

Part I: Implementing fraud detection systems

Paper A

New security issues in emerging computing environments - A reflection

Reprinted from

Technical Report 04-02, May 2004
Department of Computer Engineering
Chalmers University of Technology
SE-412 96, Göteborg, Sweden

New security issues in emerging computing environments - A reflection

Håkan Kvarnström [*] Ulf Larson
{*hkv, ulfla*}*@ce.chalmers.se*
hakan.kvarnstrom@teliasonera.com
Department of Computer Engineering
Chalmers University of Technology
412 96 Göteborg, Sweden

Abstract

Society is becoming increasingly dependent on information and communication systems to provide distributed, networked and instantly available services regardless of location. Users are offered a seamless experience in their services, not only with respect to authentication and access-control but also to services and network access available in their vicinity. The services are ubiquitous and pervasive, adaptive and tailored to suit the needs of the user and his current context.
Unfortunately, this evolution is accompanied by greater vulnerability to threats. Existing threat models must be revised to meet new requirements imposed by this scenaro. In this position paper we argue that concepts and mechanisms, such as detection and prevention mechanisms (e.g. intrusion detection systems and intrusion prevention systems), are essential and will be increasingly important to counter these threats in emerging IT environments for e-business and e-society. They serve as a powerful complement to and, to some extent, a replacement for conventional preventive security mechanisms. We give real world examples showing selected security problems and requirements of emerging computing technologies.

Keywords: computer security, fraud detection, intrusion detection, pervasive computing, ubiquitous computing, security countermeasures

1 Introduction

Information technology plays an increasingly important role for organizations and individuals of all kinds. The *public sector* relies on information and communication systems to provide everyday services such as health care and e-government

[*]The author is also with TeliaSonera AB, SE-123 86 Farsta, Sweden

to society. Medical records that are stored electronically allow information about a patient's medical history to be accessed far from his local medical center in the event of a medical emergency away from home. E-government services such as requests for birth certificates, submissions of tax declarations and applications for financial aid are provided as a value-added service to the population. This not only improves the level of service but also decreases authorities' work loads due to the higher degree of automation.

Corporate organizations move parts of their business towards online services providing e-commerce, information and communication services to allow their customers to increase their efficiency and competitiveness. Large network infrastructures connect subsidiaries across the globe while numerous short-range access networks enable end-users and their terminals to communicate from wherever their physical location might be. *End-users* benefit from increased access and availability to information and entertainment services allowing them to save time as well as kill time by having seamless and continuous access to their services and resources. Mark Weiser wrote in his seminal article, "The Computer for the 21st Century" [Wei91]:

> *"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it".*

Although this evolution adds much value to individuals and organisations, new risks emerge. As an increasing number of public and corporate services move online, the threat model by which we assess personal, business and even national security risks needs to be revised along with the countermeasures we use.

2 A case study of existing IT environments

For the sake of discussion we will study an IT dependent sector, namely *telecom*. We study it from an IT perspective and give examples of how deployment of IT can benefit a telcom company's customers.

2.1 Telecommunication services providers

Telecommunication services providers offer services to their customers that allow them to fulfill their communication needs. These communication services range from the physical layers (e.g. FDDI, SDH, Ethernet) to the application layer (e.g. web and messaging services). Traditionally, telecom companies have focused on person-to-person communications services (e.g. fixed and cellular / mobile phone services), although this has dramatically changed due to the rapid deployment of the Internet and its wide range of services. Recently, the introduction of mobile Internet services (e.g. GPRS and UMTS) has blurred the boundaries between mobile person-to-person services and the Internet, allowing new types of services such as mobile mail boxes and chat. The evolution of mobile phone services going from analog terminals dedicated to voice to digital smart phones with full Internet access has mainly been driven by the miniaturization of electronics, increased bandwidth over radio links and global standardization efforts. Simultaneously, an increase in customer demands for mobile services has allowed a rapid development and deployment of these technologies, enabling users to use mobile voice and Internet services in most parts of the world.

To give a better idea of how mobile terminals and networks can add value to users, we will give an example that illustrates a typical use case, *the mobile office*. A mobile office environment allows users to access services and resources wherever they may be. This *nomadic* behavior is becoming more and more popular as IT pervades our lifes and physical surroundings. Figure 1 shows a user connected via the GPRS network to the Internet and his home office.

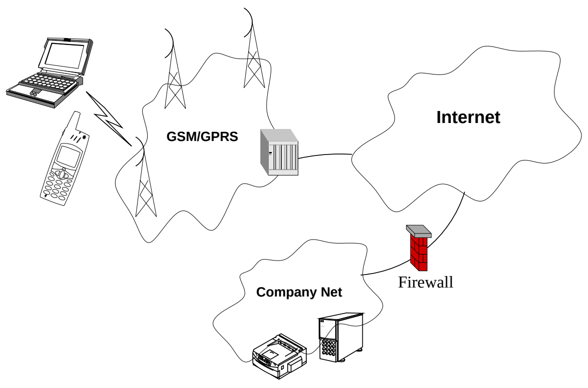

Figure 1: A mobile user accessing his home office

E-mail and messaging. The extension of e-mail and messaging (e.g. ICQ, MMS) to mobile terminals is a valuable complement to the office workstation. Full featured e-mail clients are available for most types of terminals, which gives the user a degree of freedom not previously available. Not only can he receive short messages via SMS, he can also exchange documents, pictures and audio clips just as he would on any regular computer.

VPN (Virtual Private Network) access to the home network. VPN services allow the user to securely access information and resources in his home network. The VPN creates an encrypted tunnel between the user's terminal and a termination point inside the company's security domain. It allows him to communicate over insecure networks without the risk of compromising the security (integrity or confidentiality) of the communication and the information carried.

Positioning services. Positioning services can be offered by triangulating direction vectors to the terminal from several base stations. A wide range of services can be offered using position data as an enabler. Digital maps and travel guidance can be offered to help the user find a certain address. Context aware services such as weather forecasts, speed radar warnings and traffic avoidance information can utilize the user's position to send up-to-date and accurate information based on his current position. Among the category of less desirable services (from the user's perspective) are position dependent advertisements, which can send special product offerings and sales information when the user is close to a store or a place of purchase.

Premium rate services (PRS). Similar to premium rate phone numbers, where a subscriber can call a number to get technical support, phone directory information and access to explicit content etc, premium rate messaging services (PRMS) can be used to enable charging of services (e.g. micro payments). Premium rate SMS services can be used to allow companies to charge for online services without the trouble of deploying a separate billing system beyond the existing phone service subscription. In the context of the mobile office, PRMS can be used to find names and phone numbers in online directories and to order documents and other data from information services available online.

WAP services. WAP is the mobile environment's equivalent to the world wide web. The lightweight WAP protocol (WML) allows mobile terminals to access information databases and have them presented in a form suitable to his terminal's capabilities (e.g. small screen, limited input capabilities etc). In combination with GPRS, which provides an ”always on” experience, the user can reach the Internet as well as his company's Intranet.

The above techniques, which are in use today, create new opportunities for both professionals and non-professional users. However, new enabling technologies introduce complexity and security vulnerabilities that we need to address. Section 3, describes fraud and security vulnerabilities imposed by these techniques.

2.2 Other environments and sectors

These scenarios from the telecom environment contain many challenges that must be addressed with respect to security. However, there are many other examples where IT pervades our lives. To a large extent, the security concerns we face are similar in nature and they depend more on their pervasiveness in society than on technical differences among different fields of applications. We will briefly mention a few that we believe have now and will continue to have a great impact on IT in society.

Electronic banking. One area of traditional business in which information technology has had a great impact is banking. The integration of traditional banking and IT, e-banking, has grown rapidly in the past few years. According to a survey by Datamonitor[Dat03], the forecast for Europe in 2003 was 60 million online banking users, which is a threefold increase since 2000. The e-banking concept has evolved from a situation where the customer needed to be physically present in the bank (or at an ATM) to a situation where he has total mobility using his cell phone or WAP browser. E-banking today adds much value to customers, such as location independency, time savings and 24-hour access to his services. Examples of services are:

- Access to personal bank accounts
- Payment of bills
- Money transfers
- Stock trading

Electronic payment. The use of electronic payment has evolved from traditional credit cards to more sophisticated and secure mechanisms. The security of credit cards is basically based on an insurance model, i.e. credit card companies accept the risks and costs caused by fraud. For many applications, credit cards do not meet new requirements such as low overhead costs, increased security in transactions and scaleability. Improved payment schemes have recently been successfully deployed to add functionality and security. Schemes currently used include:

- Online Payment using PayPal [pay04], which is widely used on eBay [eba04]
- Premium rate phone calls and SMS
- Pre-paid micropayment cards
- Pre-paid calling cards

E-government and E-society. A rapidly growing area that has just begun to impact everyday life is e-services that enhance existing societal and governmental services. This evolution will increase the service level and availability of many previously manual services. Even though the evolution is slow and has just started, several services have been tested and evaluated in many countries for large-scale deployment:

- Electronic tax declarations
- Online education
- Electronic and online voting
- Remote and online health care services
- Remote monitoring and surveillance (e.g. monitoring children at a day-care center)

The services named above share many similarities and characteristics. Many of the security problems are similar and can be addressed with the same countermesures. It is clear that *trust* is a key issue: (1) The user must trust the services to be accurate and indeed provide the service they claim to provide. (2) The services must be able to establish trust in the entities that utilize them. Both of these are escalating problems that will be explored in the following chapters. They have in common that they depend heavily on communication infrastructures connecting users and services. As with many new technologies, people tend to grow dependent upon them. In these cases, there is no reason to believe otherwise, as these services are slowly gaining an influence on our lives.

3 Fraud and security concerns in existing telecom environments

The scenarios we have discussed raise concerns about the security required to meet the needs of the users and the service providers. In the light of the mobile office scenario, we will limit our discussion to issues related to mobile access technologies. However, most of our ideas are applicable to fixed access technologies. Section 2.1, described a *mobile office* scenario. As useful as this might be to professional users, there are threats and risks that must be addressed. Fraud in cellular phone service has existed since the deployment of the first analog systems, which comprised analog speech transmissions. AMPS (Advance Mobile Phone Systems), TACS (Total Access Communication Systems) and NMT (Nordic Mobile Telephony) systems are examples of systems that have been widely used. The analog systems were not designed with security as a top priority, which lead to extensive fraud. Due to the high amount of fraud, most European countries have now switched to digital systems which more or less solve these security problems. For historical reasons, we will briefly discuss typical fraud cases for analog cellular systems, such as tumbling, eavesdropping and cloning.

Tumbling. Tumbling attacks exploited weaknesses in the subscriber verification mechanism when a mobile phone was outside its home area (roaming). The fraudster kept switching between (tumbling) electronic serial numbers before each new call, which effectively circumvented network security checks. The result was that a fraudster obtained cellular phone services free of charge. This attack could also lead to economic loss for other subscribers, as the attacker could have used an existing serial number.

Eavesdropping. The analog cellular phone systems typically did not have encryption of speech transmission. They were therefore vulnerable to eavesdropping. This was regularly exploited and made it impossible to transmit classified information over the cellular phone system.

The switch to digital cellular services has not completely solved all security problems. Many of the problems have been solved, such as these described above, but new security problems have emerged. These problems are not caused only by technical flaws but also result from the way in which the service is packaged and sold to subscribers. Below we give a short introduction to security problems and fraud known to exist in current digital cellular systems deployed around the globe.

Cloning. It has been shown that it is possible to clone SIM-cards in the GSM system and to duplicate terminals so that they have identical serial numbers. This means that the legitimate subscriber is billed for all voice and data traffic originating from the cloned terminal.

Roaming fraud. A feature of modern cellular phone systems is that a subscriber may use his phone outside his operator's coverage area (roaming). This is especially useful during visits to other countries where the subscriber can use the foreign operator's network to gain access to the phone system using his own cell phone. Billing and charging during roaming are still the responsibility of the home operator, which has a roaming agreement with the foreign operator to cover all charges generated by the subscriber. There may be significant delays in information exchange between the operators, however so the subscriber's credit limit may have been greatly exceeded. This delay may be used by fraudsters to use a stolen or cloned phone for free until the real customer finally receives his bill and the phone is blocked for further use. According to the roaming agreement between operators, the fraudster's operator is liable for the roaming fees, which may be a considerable loss. Note that this is not a purely technical problem but also an issue of contractual agreements and regulations.

Subscription fraud. Subscription fraud is conducted by signing up for a subscription under a false name and address. Several months may elapse from the period between the start of the subscription and until the time it becomes clear that a payee does not exist. The name and the subscriber may not even exist and thus there is no valid address to which the bill can be sent. During this period, the phone is usually used extensively and generates huge bills and a considerable loss to the operator.

Call selling operations. Call selling operations are a clever way of making money on the cellular phone system. It is basically a variation of subscription fraud, where false subscriptions (often in large amounts) are used to resell calls at a lower price. The fraudster uses features such as call-forwarding and three-way conferencing to connect multiple sites. This service is sold at a low and competitive rate and is often used for long distance calls, which in turn generates huge bills. Since the subscription is obtained under a false name, the operator cannot send a bill. After the operator has locked out the phone in its network, which can take weeks or months, the fraudster can reprogram the phone and sell it on the black market for an additional profit.

Premium Rate Service (PRS) fraud. Premium rate services allow a service provider to charge for his services using a dedicated premium rate phone number (e.g. a 900 number in Sweden). Typically, a premium rate service is set up such that the operator collects a fee for the service and uses a revenue sharing scheme to divide the income between itself and the owner of the PRS. PRS fraud comes in many forms. An especially ingenious fraud is a combination of subscription fraud and PRS fraud. A fraudster starts a company, possibly abroad or using a dummy owner of the company. The company sets up a number of ”premium rate services” pretending to provide some service to the public. Other phone subscriptions (fixed or mobile) are then used to call the PRS number, which quickly generate large incomes. The phones used may have been acquired under false names (subscription fraud). It is not unusual that these lines are kept busy 24 hours a day to maximize the income. The operator pays the fraudulent company for his ”income” but will eventually (weeks or months later) fail to bill the subscribers using the service as their subscriptions were acquired under false names or using cloned phones. Once the operator starts to investigate the PRS provider, the company is closed and the contact persons cannot be found.

4 Emerging computing environments

This section discusses techniques that may affect the way we use information technology in the near future. In the same manner as powerful computing and storage techniques have woven themselves into people’s everyday lives, transparent access to resources will soon be a reality. The concept of pervasive and ubiquitous computing constitutes the foundation of data transparancy and access. It allow us to always be able to access personal data resources through smart transmission methods and to have base stations and receiving devices covering areas such as university campus areas and corporate buildings.
Beginning with Mark Weisers seminal article [Wei91], the concept of ubiquitous and pervasive computing has grown into a reality. The ideas can now be realized as hardware devices shrink in size and weight and as new technologies for transmitting and receiving data appear on the market.

Networking techniques such as Ad-hoc networking and peer-to-peer [Ora01] networking are gaining in popularity and grid computing architectures are applied to solve complex, distributed and time consuming problems.

4.1 Ad-hoc networks

An ad-hoc network is a wireless network without a permanent infrastructure. Ad-hoc connections are spontaneously created when nodes come close to one another. These connections often only last for a limited time, e.g. in relation to a data transfer. After a successful session, the nodes disconnect. An ad-hoc network works well in situations where events are spontaneous and short, e.g. when someone attends a meeting at work or downloads a timetable at a bus stop. The advantage of an ad-hoc network is that it allows a user to use localized services simply by moving close enough to the service source, e.g. when the user comes close to a bus stop, a timetable is automatically downloaded into his mobile device. Popular technologies for ad-hoc networks include Bluetooth and WiFi (IEEE 802.11b).

4.2 Peer-to-peer networks

A peer-to-peer network is an architectural structure where each communicating pair of nodes is both server and client at the same time. Unlike the traditional client-server model, where one computer is designed to provide services for connecting clients, the peer computers can take on both roles. Communication between peers allows both peers to initiate connections and transfer files at the same time. A combination of ad-hoc networking and peer-to-peer networks is a file sharing network such as Kazaa[Kaz04]. Here, the users of the network can both act as server, i.e. other peers download files from users, and client, i.e. files are downloaded from other peers.

4.3 Ubiquitous and pervasive computing

Ubiquitous and pervasive computing [Wei91] are ideas that aim at creating a transparent, integrated computing environment. Computers and computing units will be woven into the general infrastructure of society and will be found in such everyday items as coffee cups, clothes and cars. As technology evolves and computing devices become smaller and more powerful these ideas become implementable. Techniques such as voice recognition, Internet connectivity and wireless communication are used to create the transparent computing environment, an environment where the resources and individuals are always reachable, regardless of location or time.

5 Facing new security risks

As new computing paradigms gain acceptance, new security risks emerge. This section gives insight into some of the problems we will face once these new techniques are deployed.

5.1 Difficulties in defining a security domain

Future e-business scenarios are likely to be more complex than today's. The new services outlined in previous sections are expected to bring about new business models and, as a consequence, new actors seeking different roles. Several different types of actors will interact with each other to realize a service. A *service provider* may have a relationship with the customer. He would be responsible for the service, its provisioning and support to end-users. The service provider depends on other actors to be able to fully deliver his service, such as a *network provider*. The network provider ensures that the user's terminal has network connectivity to the service. The service provider may also need to work with a *content provider* who provides the contents needed to realize a service. Examples of contents are digital maps, digital audio and video, user position data etc. The user may pay a *payment provider* for his service. The payment provider is responsible for charging for the use of a service. As many other functions, charging requires security services such as authentication to be reliable. The service may also need other security functions, such as encryption and non-repudiation services. Security functions may be provided by a *security service provider* that offers services such as PKI, authentication servers, logging and audit functions, and non-repudiation services.

The brief example above shows that many different actors may be involved in realizing a service. A problem here is that it is difficult to clearly define a security domain. A security domain may be defined as having a common set of security policies governed by some authority. Here, a service may be provided by several different companies, possibly located in different countries following different laws and regulations. Making a risk analysis for this scenario is a complex task, even though the service itself may appear to be simple.

5.2 An increase in fraud

Fraud is driven by the urge of an individual or organization to gain some benefit. As the value of the services produced and delivered by IT systems increases, so does the incitement for fraud. A clear indicator of this can be seen in the area of digital media, where individuals download and redistribute music and video files for their own use and for profit. The media industry is now trying to fight these fraudsters by every possible means. The human mechanisms behind the behavior is easy to understand: Cheap is good, but free is better. Fraud investigators often talk about the *10-80-10* law, which states that 10% of the people will never commit fraud; 80% would commit fraud under the right circumstances and 10% actively seek opportunities for fraud. Unfortunately, as more and more services move online, new opportunities emerge and fraudsters coming into being. It is highly likely that we have only seen the beginning of these problems. Fraud is not likely to be combated by preventive means only. Fraudsters often attack the off-line processes of a service. Weaknesses in the billing/invoicing system, flawed contractual agreements or a lack of ability to establish the subscriber's true identity at the time of purchase are examples of weaknesses they exploit, exploits that already causes substantial loss to operators around the world.

5.3 Vulnerable interfaces and terminals

The mobile terminal of today is becoming increasingly complex. The computational and storage capabilities of a modern terminal resemble those that desktop computers had just a couple of years ago. Their network capabilities are often extensive. It is not rare that they have the best of several worlds in terms of communication: first, GPRS for radio-based and always-on communication over long distances; second, WLAN for short-range radio access using Internet hotspot; and third, IR and Bluetooth for personal area network (PAN) applications allowing resource sharing and service provisioning in the user's immediate surroundings (e.g. the same room or in a car, train or airplane). All these techniques have vulnerabilities, and several have been discovered and published [New01]. Recently, a flaw in Bluetooth-enabled phones was discovered that allowed intruders to access information in a nearby phone. The intruder could also use the phone as an access point to the Internet over GPRS [Whi03] without the authorized user's consent.
That an attacker can access information in mobile terminals may have serious consequences. The increasing use of keys for authentication and encryption in combination with weaknesses in terminal security pave the way for new identity theft attacks. This can be prevented by storing keys in tamper-resistant hardware (e.g. smart cards), but limitations in terminal capabilities and the desire for user friendliness often result in keys that are stored locally on the terminal or device. Moreover, the consequences of an identity theft attack is likely to become worse as the value of the online services increases.

VPN services have recently become an important technology in mobile personal computing and are often used to access a company's Intranet from a remote location. In section 2.1, VPN services were mentioned as an enabling technology for the mobile office. In the light of the security domain discussion above, this technology adds another strong argument for storing keys securely. A security domain is not stronger than its weakest member and each VPN connection is a possible weak link. It is difficult or almost impossible for a firewall to distinguish a compromized client from an intact if client authentication succeeded. However, the authentication may have been conducted using compromized authentication keys, which leaves the company firewall in doubt. Suddenly, the company has a situation in which it has as many possible entrances through the firewall as it has employees. The notion of a securiy domain with boundary protection has then collapsed and rendered the firewall useless.

5.4 Jeopardizing user privacy

The increasing use of mobile terminals also jeopardizes users' privacy. All communication by radio is by definition observable from a distance. In GSM / GPRS, the position of a user can easily be established by triangulating the user's position in relation to the radio base stations. GSM and GPRS have protective measures to protect the privacy of subscribers by using temporary subscriber identities (TIMSI) during communication. However, location-based services are becoming popular, which forces telecommunication companies to provide location data to service providers in a refined format. The attacker is not forced to listen to the radio interfaces but can attack the management systems available to manage subscribers and their location. Other techniques such as WLAN are also subject to privacy attacks. In WLAN, the identity of the user[1] is broadcasted in clear text. The same problem exists in Bluetooth, where the hardware address can be intercepted in the clear.

6 New security requirements

It is quite clear from the above discussion that it will be necessary to enforce new security requirements to meet the challenges in providing new types of computing environments.

6.1 New trust models, establishment of identity and authorization.

Establishing the identity of users and entities becomes increasingly important as more and more services are offered online. E-government and health care services are perhaps the most notable examples of this, where the identity of the user is vital to the trust and reliability of the services. Existing trust anchors and mechanisms, such as public key cryptography and PKI, are used for this purpose.

However, existing solutions do not always solve new problems. Scaleability issues will limit the applicability of such existing security services as authentication and authorization. The entities consist not only of human users, but also a large number of *connected things*. A classic and often mentioned scenario is the intelligent and automated home, which is filled with small, connected, communicating gadgets. The possible explosion of gadgets makes it difficult to use current means for security that is based on identity claiming, authentication and authorization. This problem can be illustrated with an example from a futuristic home environment. Consider a home with several refrigerators. How would you assign names to these refrigerators and how would you tell the grocery store what items the refrigerators are allowed to order? Who would be responsible if expensive groceries are ordered without your consent? It is clear that a home environment with hundreds, or possibly thousands, of connected things cannot utilize authentication and authorization schemes as we traditionally use them (e.g. access control lists). This would create a security management problem of huge proportions. This is especially true in environments where a massive number of entities perform frequent actions on a large number of objects. It should also be pointed out that entity authentication builds upon the trust in the entity.

[1] The MAC address of his network interface card, which is globally unique

This means that, if the entity cannot be trusted, entity authentication will fail to provide any assurance of the true identity [CGRZ03].

The problem of establishing the identity of users and entities in pervasive computing environments forces us to develop *new trust models* that can be used to establish trust (e.g. in a business transaction). The identity problem affects many other security services. For example, it may be hard to enforce repudiation, as the identity of a communicating party may not have been established. Trust in the identity also forms a basis for sound key management. Even in key exchange schemes such as Diffie-Hellman [DH76], entity authentication is important to be able to twart certain attacks.

From the identity problem follows the problem of authenticity. How can we be certain that e.g. a piece of software, information or an instruction sent from a claimed identity is indeed authentic? In general, if we are not certain about who sent us the information, we are less likely to trust its authenticity.

How can we possibly trust a software agent acting on behalf of its user? One way to achieve trust and enable business transactions would be to build the trust upon the behavior of the agent and not entirely on its claimed identity. This would require a behavior based detection scheme to identify anomalous behavior. It is also likely that multiple security mechanisms must cooperate to establish a reasonable level of trust.

6.2 Maintaining accountability and respecting privacy.

Lack of authentication and trust raises concerns about the possibility to provide accountability. Accountability means that users and entities can be held accountable for their actions. To be able to achieve that, it is necessary to be certain about the identity of the entity performing a certain action. This may indeed be a complex task, even in a traditional client-server architecture where all users are properly authenticated and a considerably more difficult task in more complex computing architectures. An entity can collect data for the purpose of accountability, but he may fail to identify the fraudster. A naive accountability model would be to allow entities to do anything they like and then try to detect misuse. This could be done by every entity in the system and hence would work even in ad-hoc network applications where the identities of the nodes are unknown. However, though the identity may be unknown to a local entity, this may not be the case in a broader perspective. User-initiated actions such as the creation of ad-hoc networks, e-commerce transactions, file sharing, group communication etc. will inevitably leave footprints in the digital domain. These footprints can easily be analyzed in attempts to tie a certain event to a user or entity and thereby identify a fraudster who tries to remain anonymous.
Increasing surveillance to achieve accountability comes with a price. As more and more users move online and the number of available services provided increases, privacy also becomes an issue.

In a scenario discussed by the Gartner Group, they state that [Gro]:

"By 2010, driven by the improving capabilities of data analysis ... privacy will become a meaningless concept in Western societies."

This frightening prediction may very well be a reality if we fail to take actions to prevent it. Deploying privacy enhancing techniques such as pseudonymization of data helps to some extent. However, pseudonymization creates other problems. It can be harder to achieve accountability if the user wishes to remain anonymous. There is a fine balance between anonymity and accountability such that *reidentification can be achieved for liability purposes* while at the same time *sufficient privacy* is offered to users. This was demonstrated in an article by Biskup and Fleger [BF00] in a typical intrusion detection system. Recent advances in data correlation techniques make it even more difficult to prevent reidentification as information from many different sources can be aggregated to narrow down the number of possible identities that are using a service. In addition, the adoptation of cheap wireless gadgets such as Bluetooth and RFIDs will further increase the possibility to track people without their knowledge and beyond their control.

6.3 Impact on telecommunication companies and their services

Telecommunication companies will face new challenges in the years to come. Increased use of pervasive and wireless technology, new business models and regulative requirements put new demands on current infrastructures and architectures. Design and development of reliable and trustworthy security services are perhaps the most important efforts when deploying these new services. New business requirements mandate that structural and organizational flexibility be provided. Structural flexibility means that products and services can be realized independent of the structural composition of the underlying network. Security services should be able to adapt to structural changes as the infrastructure evolves over time. Organizational flexibility means that mergers and acquisitions can be performed without jeopardizing the IT infrastructure. Compatibility of security functions, separability of users, security functions and controls are examples of important characteristics needed to fulfill the flexibility requirements.

Increasing use of network encryption will affect the infrastructure in many ways. First, audit and detection systems such as IDS will change from being mainly centralized to a highly distributed function monitoring the tunnel termination points. Second, the notion of security domains will slowly change. In today's networks, firewalls and network level filters are the predominant control to define a security policy. In the years to come, a more flexible definition of security domains is needed to provide increased flexibility and the ability to meet expected business requirements [Bla96].

As pervasive and ubiquitous computing concepts are adopted, new problems arise. Privacy issues (e.g. location privacy) are an important area where we need to find solutions in order to gain user acceptance of future services. Also, new legislation and regulations create new demands for privacy as well as other security mechanisms (eg. lawful interception).

The increasing complexity of network infrastructure requires vendors and suppliers to access customers infrastructure components for the purpose of remote management. Often these components are highly distributed and are part of the telecom company's security and administrative domains. In all, the telecom company scenario is complex and the complexity is rapidly increasing. Moreover, the problems described are not well-addressed using existing means of protection. New protection schemes must therefore be applied to explore possible solutions to the problems they are facing.

7 Countering the new threats

It is no longer sufficient to *prevent* attacks in an otherwise friendly environment. When creating new systems they must be designed to work in a *hostile environment* where new attacks threaten the system on a daily basis.

7.1 Rethinking current defences

Finding countermeasures to protect computer systems is not a trivial task. It is not just a matter of adding some encryption, authentication and logging services and believing that security has been taken care of. What kinds of countermeasures should you focus on? What kinds of threats are we seeking to find protection against? To bring some order to this problem, a categorization of security countermeasures can be used. The categorization shown in Figure 2 was inspired by [Sch00].

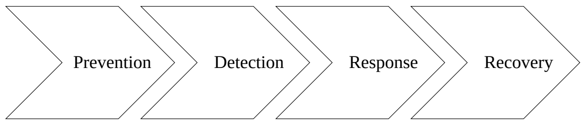

Figure 2: Security Countermeasures

These four categories complement each other and, if we focus our countermeasures on one of the categories, we may give less attention to others. For example, if we put all efforts into protecting our target environments using strong authentication, encryption and access control, we may relax our requirements for detection and response. This can be done because we may feel so confident that our *preventive* countermeasures do their job so well. On the other hand, if we know that our target environment suffers from weak authentication (e.g. passwords in clear text) and the access control is based on those weak presumptions, we may want to increase our countermeasures for detecting intrusions and fraud.

When deploying security, countermeasures are normally distributed over these categories to create *a layered security architecture*. Attacks circumventing the preventive measures can then be detected and hopefully stopped before damage occurs. The next sections will discuss each countermeasure and give indications of how these must change to meet future needs.

7.2 Prevention

The goal of the preventive countermeasures is to prevent security threats by eliminating as many vulnerabilities as possible. As previously stated, preventive mechanisms cannot always be trusted (e.g. entity authentication). In general, we believe that increased complexity will make it hard to design preventive controls that can fulfil the security requirements of emerging computing enviroments. Prevention works best under strict and controlled conditions in which a high assurance of trust can be established. Tradionally, these countermeasures have been provided by a company or organisation under the supervision of professionals. As the computing paradigm shifts from static networking models to pervasive and ubiquitous computing, so does the security architecture. The countermeasures must be heavily distributed and rely upon the responsibility of many parties. As the responsibility of the individual for the overall security increases, we need new measures to augment the preventive countermeasures. Dan Geer, former CTO of @Stake, states that authentication cannot possibly keep up with the number of people who need it and the number of transactions we try to control with it. He argues that authentication does not scale, but that surveillance does.

> *”The costs to observe are virtually zero, so it’s not a question of will it exist, but what will we do with it?”*

This leads us to the next countermeasure, which aims to detect misuse.

7.3 Detection

Detection of intrusions, fraud or attempts thereof are usually considered a second layer defence. When the preventive mechanisms fail, detection comes to the rescue and raises alarms. The problems described above concerning the diffculties of providing sufficent preventive countermeasures may lead to a shift towards increasing the use of fraud and intrusion detection systems. A relaxation of the preventive means may not be as serious as one may first believe. Once again, a quote by Dan Geer explains this elegantly:

> *”You can do anything you like, but if you screw up badly enough, someone will find you.”*

In some services, misuse and intrusions may be irrelevant, as long as they do not affect business. In others, this may not be the case. Detection may also identify misuse by insiders (legitimate users) in addition to external adversaries. As the value of services increases along with increases in the number of available services, insiders may become tempted to misuse services for their benefit.

We believe that detection could prove to be a valuable tool in establishing trust. Assurance of an identity may be less relevant than assurance that a claimed identity is trustworthy. Of course, the notion of trustworthy depends on the application and the actualt context. Establishing trust using detection is similar to how humans establish trust. A person may trust a local locksmith to make a copy of his car keys just because he, or his trusted friends, have previously had good experience in using his services. After some time, without breaching a customer's trust, the locksmith gains a reputation of being trustworthy.

In the light of the research problems discussed in this paper, future detection countermeasures must be designed to meet future needs and requirements:

1. Lightweight and flexible to allow integration in mobile terminals and gadgets
2. A shift towards detection of misue of business logic and entity behavioral analysis rather that looking at bits and bytes.
3. Tighter integration of detection and prevention. Today, manual reconfiguration is used chiefly to thwart attacks after a detected break-in
4. Detection schemes that can operate in highly distributed enviroments without a centralized infrastructure
5. Detection schemes and mechanisms that can operate in hostile and unfriendly environments without jeopardizing the security of the target systems or the trustworthiness of the detection system itself.

7.4 Response

Taking proper action after an attack or security breach has been detected is a response countermeasure. An attack may lead to an action to close down sensitive information systems to prevent breach of confidentiality. Other actions may be to notify a security officer or reconfigure packet filters (e.g. firewalls) to block incoming network traffic from certain IP addresses. A proper response is essential to make detection useful. The deployment of context-aware and pervasive computing environments will need a more active security management than previously. Ad-hoc security associations between communicating parties may be created on-the-fly in untrusted environments. Under such conditions, self-protection using intrusion/fraud detection will be essential. With the same ease as connections and security associations are created, they need to be torn down in the case of a detected misuse. Active response based on misuse detection will play an important role in these environements.

7.5 Recovery

Recovering from an incident means that the system must be restored to its latest good state. Recovery can affect everything from administration to technology. Administrative routines may have to be changed, new passwords must be assigned to users and the integrity of applications and data must be assured. Also, log files may need to be restored from backup to allow forensic analysis. Full recovery may not always be possible, as a breach of confidentiality is irreversible. Once the information is disclosed to an attacker, the secret is out and cannot be reversed. At a deeper level, recovery may be provided using techniques from the field of dependability and survivability. In-depth security mechanisms providing a multi-layered defense strategy may prove useful to counter attacks before serious damage occurs. These defenses can be guided by detection and response measures to optimize the defense strategy and limit the consequence of an attack.

8 Conclusions

Classic preventive countermeasures such as authentication, encryption and access-control will not be adequate to achieve sufficient security in emerging computing enviroments. We argue that detection and response will play an important role in realizing these environments. We believe that a tighter integration between different countermeasures must be established to allow preventive countermeaures to be actively managed and allow them to respond to threats and intrusions. We state that it is important that detection systems take the business logic into account to detect and prevent attacks causing economic loss. Techniques from many fields, such as dependability, survivability, intrusion and fraud detection, must be combined to achieve better coverage in the overall protection. We believe that existing trust models upon which we establish security will not meet future requirements. We must develop new trust models based on threat assessment guided by intrusion and fraud detection to better respond to intrusions and misuse.

References

[BF00] J. Biskup and U. Flegel. On Pseudonymization of Audit Data for Intrusion Detection. In *Workshop on Design Issues in Anonymity and Unobservability*, pages 161–180, 2000.

[Bla96] B. Blakley. The Emperor's old armor. In *Proceedings of the 1996 Workshop on New Security Paradigms*, pages 2–16, Lake Arrowhead, California, USA, September 17-20 1996.

[CGRZ03] S. Creese, M. Goldsmith, B. Roscoe, and I. Zakiuddin. Authentication for Pervasive Computing. In *Proceedings of the 1st International Con-*

ference in Pervasive Computing, LNCS 2802, pages 116–129, Boppard, Germany, March 2003. Springer Verlag.

[Dat03] Datamonitor. Europe's online banking population to rise. http://www.nua.ie/surveys/?f=VS&art_id=905358751&rel=true, 2003.

[DH76] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory 22*, pages 644–654, 1976.

[eba04] ebay. http://www.ebay.com, 2004.

[Gro] Gartner Group. Insight for the connected world. Report.

[Kaz04] Kazaa. http://www.kazaa.com, 2004.

[New01] T. Newsham. Cracking WEP keys. *Black Hat Briefings*, 2001.

[Ora01] A. Oram. *Peer-to-Peer : Harnessing the Power of Disruptive Technologies*. O'Reilly & Associates, 2001.

[pay04] Paypal. http://www.paypal.com, 2004.

[Sch00] Bruce Schneier. *Secret & Lies - Digital Security in a Networked World*. John Wiley & Sons, Inc, 2000.

[Wei91] Marc Weiser. The computer for the 21st Century. *Scientific American 265(3)*, pages 66–75, 1991.

[Whi03] O. Whithouse. War Nibbling: Bluetooth Insecurity. Technical report, @Stake, 2003.

Paper B

A synthetic fraud data generation methodology

Reprinted from

Proceedings of the Fourth International Conference on Information and Communications Security
Singapore
December 9-12, 2002

A synthetic fraud data generation methodology

Emilie Lundin Håkan Kvarnström* Erland Jonsson

{*emilie, erland.jonsson*}*@ce.chalmers.se*
hakan.kvarnstrom@teliasonera.com
Department of Computer Engineering
Chalmers University of Technology
412 96 Göteborg, Sweden

Abstract

In many cases synthetic data is more suitable than authentic data for the testing and training of fraud detection systems. At the same time synthetic data suffers from some drawbacks originating from the fact that it is indeed synthetic and may not have the realism of authentic data. In order to counter this disadvantage, we have developed a method for generating synthetic data that is derived from authentic data. We identify the important characteristics of authentic data and the frauds we want to detect and generate synthetic data with these properties.

Keywords: fraud detection, synthetic test data, data generation methodology, user simulation, system simulation

1 Introduction

Fraud detection is the process of automated analysis of data from a service with the goal of revealing attempts to use the service without paying or in some other way illicitly benefit from the service. When designing a fraud detection system, it is essential to have suitable data for evaluation and testing. This data must be representative of normal and attack behavior in the target system since detection systems can be very sensitive to variations in input data.

Using synthetic data for evaluation, training and testing gives several advantages compared to using authentic data. Data properties of synthetic data can be tailored to meet various conditions not available in authentic data sets. On the other hand purely synthetic data suffers from the fact that they may be quite

*The author is also with TeliaSonera AB, SE-123 86 Farsta, Sweden unrealistic and will consequently not properly reflect the properties of authentic data. Thus, the aim of this work has been to develop a methodology for generating synthetic data with realistic properties and this is achieved by using authentic data as a property ”seed”. Thus, the method uses statistical properties from smaller amounts of authentic data and generates large amounts of synthetic data preserving parameters important for fraud detection and the training of fraud detection systems. Examples of such parameters are user and service behavior.

The rest of this paper is organized as follows. Related work is summarized and discussed in section 2. Advantages and motivation for using synthetic data can be found in section 3 and our method for generation of synthetic data is presented in section 4. Conclusions can be found in section 5.

2 Related work

There are very few papers which focus on properties of test data. Most papers describing training and test data focus on testing a specific detection prototype, or is about benchmarking and comparing detection systems. Often, data is only mentioned briefly. In most papers, (manipulated) authentic data is used.

The absence of research on synthetic data for fraud detection applications has led us to study work in the intrusion detection area. In [KLJ00] we discuss several similarities between fraud and intrusion detection systems. However, there are some differences that need to be considered. Intrusion detection is performed on log data from operating systems, applications, and networks to detect malicious actions in a computer system. Fraud detection is normally performed on log data from a specific service to find people trying to gain unauthorized benefits from the service. Fraud detection is often more specialized and adapted to a service as opposed to intrusion detection which is more general. Roughly, fraud detection can be seen as application-specific intrusion detection. We believe that the way test and training data is retrieved and used is similar and the same methods for generating data can be used.

In the following two sections, we review some papers that present test methods using authentic and synthetic test data. We describe how the data is retrieved, used, and manipulated, and how it compares to our method.

2.1 Authentic data

In the JAM project [Jam01], some papers have been written that actually focus on test and training data properties. For example, [CFPS99], and [SFP$^+$98] describe experiments on varying the fraud rate in training data and also introduces a cost model where each transaction is associated with a cost. In [SFP$^+$98] which is an unpublished technical report, the data and the manipulation of data is described more thoroughly, but this information is shortened in the published papers based on the same experiments. For the experiments two large sets (500000 records each) of authentic credit card transactions are used. The transactions are labeled as fraudulent or legitimate and the fraud rate is around 20%. A cooperation with two banks provided the data. The labeling seems to have been performed by bank employees and the authors mention that some of the unclear cases were indeed labeled as fraudulent. The authors concluded that the optimal rate of fraudulent vs. normal events in training data depends on the cost model and the learning algorithm used. For their applications, the desired rate is shown to be close to 50:50. The fraud categories presented in their work are likely to be representative of future fraud. However, the process for labeling data is not described.

In [BSTM+97], data is described but not analyzed in much detail. The data used for the experiments in this paper is GSM Toll Tickets from Vodafone. It consists of calls made by 300 normal users during two months. This data is "sanitized", but it is not specified how this is done. It also consists of calls made by 300 fraudulent users but it is not explained how these fraudulent users are selected. The paper describe what information the Toll Tickets contain although the ratio between normal and fraudulent events is probably not realistic.

2.2 Synthetic data

Probably the biggest effort in testing and comparing intrusion detection systems is made by DARPA in 1998 and 1999 [LHF+00]. A great deal of work has been spent on generating large amounts of test and training data for this project. The generation of data is best described in [HLF+01]. The test data contains network traffic and system call log files (SUN BSM log files) from a simulated large computer network. Both attacks and background data has been generated synthetically, but the background data is said to be similar to sampling data from a number of Air Force bases. Background data is generated mainly by using software automata simulating the usage of different services. Attack data is generated by running attack scripts. The database of attacks used is described in [Ken99]. Some more complicated background data and attacks are injected live into the system. No extensive analysis of the quality of the generated data has been made. Some efforts in automating the data generation further is described in [HRLC01].

McHugh [McH00] criticizes the lack of validation of test data in the 1998 DARPA evaluation. The process of generating data is vaguely described, and it is difficult to determine the quality of the data. There are no guarantees that the attacks used are really representative and realistic. Also, he questions whether the background data is really representative of the user behavior in the computer systems used and if the behavior in these systems represent user behavior in other computer systems. Another question he raises is whether attacks are realistically distributed in the background data. Some improvements were made in the 1999 experiments but many of these issues are still not fully addressed.

Debar et al. [DDWL98] developed a generic intrusion detection testbed. They show how to simulate user behavior with a finite state automata to obtain normal traffic, but state that this is only a practical approach if the set of user commands is limited. Instead, they use recorded live data from user sessions, which has been replayed inside the workbench. They also describe how they create attack scripts to use in the testing process.

Maxion and Tan [MT00] discuss the effects of more or less irregular background data. They generate "random" background data with different degrees of regularity and show that it affects the false alarm rate drastically. The paper also measures regularity in real-world log data.

Chung et al. [CPOM95] claim that concurrent and distributed intrusions are likely to occur in a computer system, and therefore, should be included in test data for intrusion detection systems. They have developed a tool that parallelizes attack scripts to simulate concurrent intrusions.

Puketza et al. [PZC$^+$96] describe a software platform for testing intrusion detection systems. They have used the UNIX package *expect* to generate synthetic user sessions. However, they do not analyze the quality of this data to any great extent.

From the discussion above we conclude that most projects seem to use synthetic data due to lack of authentic data or lack of data having desired properties. In the DARPA project [dar01], it was necessary to generate synthetic data since it was not possible to retrieve the desired amount of data. Also, most projects seem to suffer from a shortage of real attacks, which makes it necessary to inject fraud / intrusions synthetically.

3 Using synthetic data for evaluation

Training and testing of fraud detection systems require data which has certain properties. Access to authentic data from real services is often preferred although it may suffer from lack of control of what fraud cases it contains and the amount of data may not be sufficient. Another possibility is to use synthetic data.

Synthetic data can be defined as data that is generated by simulated users in a simulated system, performing simulated actions. This definition can be relaxed to include humans performing simulated actions on a system. Simulated actions means that people (or a program) perform actions according to a specification created by the experiment organizers, and not act according to their normal behavior. This specification should reflect the desired behavior of the system. In this paper, we use the wider definition of synthetic data. It depends on the situation whether it is better to simulate user behavior using a software automata or to hire people to generate background data and attacks. There is also a choice between using a fully simulated system, a real system or a mix of real and simulated system components.

3.1 Rationale

Use of authentic data is not always a viable solution for evaluation of detection systems. For future services (e.g. services that are planned or under development) authentic data may not exist or may only be available in small quantities. In these situations, synthetic data is the only possible solution for conducting tests. Moreover, expected fraud cases for services under development are not present in the authentic data as the service has no real users. Under these circumstances, synthetic data containing expected fraudulent user behavior must be generated for testing of the detection system.

An advantage of synthetic data is that it can be designed to demonstrate properties, or include attacks, not available in the authentic data. This gives a high degree of freedom during testing and training. In addition, synthetic data may be generated to cover extensive periods of time which could have taken months or years to collect from the real target system. Also, a large number of users can be efficiently simulated, even though the real system only has a few of them. This makes scalability tests possible.

3.2 Data for training fraud detection systems

The process of detecting fraud involves analyzing large amounts of data, looking for signs of fraudulent behavior. Intelligent techniques such as neural networks and other AI (Artificial Intelligence) techniques can be used to decide whether an event or a group of events indicate fraudulent behavior. Common to most intelligent techniques is that extensive training needs to be performed. During training, data similar to that of the target system must be available. In addition, the data must be labeled (i.e. all events must be categorized as normal or fraudulent) so that the detection algorithm can learn to distinguish normal usage from fraudulent. We have identified a number of properties that characterize good training data.

- Data is labeled, i.e. we have exact knowledge of which attacks are included in the data.
- The attacks in the input data represent the attacks we expect to occur in the target system. (Not necessarily the same attacks that currently occur in the system.)
- The number and distribution of attacks in the background data (fraud / normal data ratio) are adapted to the detection mechanism. Some detection methods perform better if they are trained on data where attacks are overrepresented. In [CFPS99], it is shown that varying the amount of attacks in the data will greatly affect the training process of the detection algorithms.
- The amount of data is large enough. In particular, certain AI algorithms need huge amounts of training data.

These properties indicate that synthetic data can be a better choice for training fraud detection systems.

3.3 Data for testing fraud detection systems

The detection algorithms need to be tested on data sets containing expected fraud in the system. Testing the detection capability of the algorithm may require a different data set than used during training, even though most of the properties for training data is valid also for test data. For example, it is easier to test the system if data is labeled. Some additional important properties for test data are stated here.

- The number and distribution of attacks in the background data (fraud / normal data ratio) should be realistic.
- The attacks in the input data are realistically integrated in the background data. For example, time stamp of an attack, time between attacks, and time between parts of an attack may affect detection results.
- Normal (background) data should have similar statistical properties as authentic data from the target system. Different behavior in the system may affect detection performance drastically. In [MT00], it is shown that the false alarm rate rises sharply when background data becomes more irregular. This would indicate that test data is often system specific.

Testing a fraud detection system involves many diverse activities. Scalability tests need to be performed to see whether the systems can handle current and future data volumes. Again, large authentic data sets may not be available to realistically conduct such tests. Various stress tests can be conducted to test the system's ability to withstand sudden changes in data volume or the characteristics of the data. For example, the system can be tested to determine its susceptibility to parameter changes. These changes could be triggered by sudden changes of the environment in which the detection system operates or by faults and errors in the collected data. By using synthetic data, faults and other unexpected events can be injected to study the susceptibility of the detection system and its ability operate under harsh conditions.

4 Data generation methodology

There is a great deal of complexity in synthetic data generation and it is time-consuming to create the necessary components. Therefore, a methodology is needed to structure the work and to point out the choices that have to be made.

The main components needed to automate the data generation process are specifications of desired user behavior in the system, a user / attacker simulator, and a system simulator. The goal of the methodology is to guide the production of these components. The starting-point of the methodology is the collection of information about the anticipated behavior in the target system. The methodology includes both background and attack data generation. Therefore, we need both information about possible attacks as well as normal usage. This data serves as basis for user and system modeling.

4.1 Methodology overview

In figure 1, the methodology for generating the data generation components is pictured. The first step is the *collection of data* that should be representative of the anticipated behavior in the target system. The data may consist of authentic background data, background data from similar systems, authentic attacks, and other collections of possible attacks. The second step is to *analyze the collected data* and identify important properties such as user classes, statistics of usage, attack characteristics, and statistics of system behavior. In step 3, the information from the previous step is used to identify parameters that need to be preserved to be able to detect the anticipated attacks, and to *create user and attacker profiles* that conforms to the parameter statistics. A *user model* is created in step 4. This model must be sophisticated enough to preserve the selected profile parameters. Also, the attacks are modeled in this step. The user and attacks simulators implements the models. In step 5, the system is modeled. The model must be accurate enough to produce log data of the same type as the target system for the input user actions. The system simulator is implemented according to this model. It is important that it is possible to configure the user and system models using variable input parameters in order to change the properties of the generated data during operation.

In the following sections, each of the steps in the data methodology are discussed and methods for implementing them are suggested.

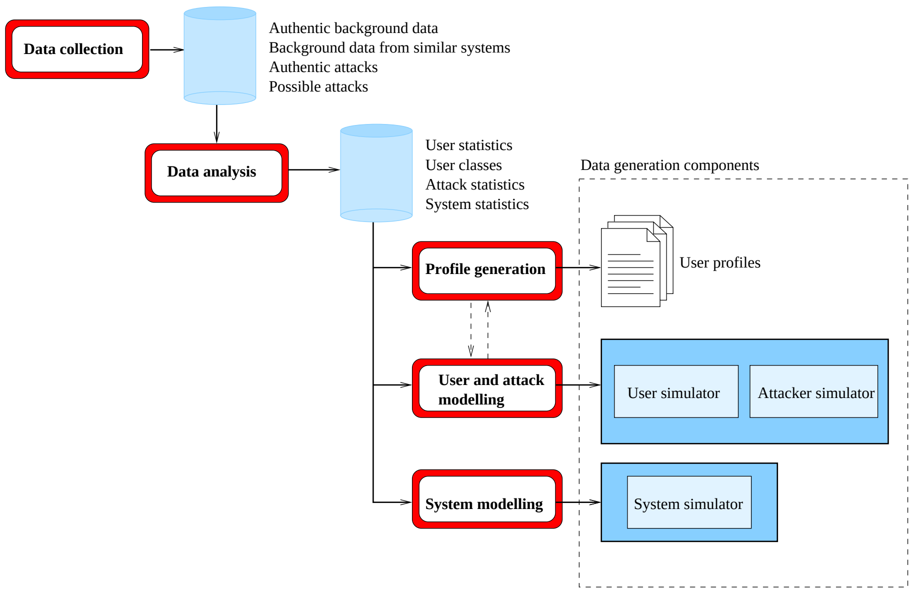

Figure 1: Synthetic log data generation method

The division into steps with well defined interfaces reduces the complexity of the task and makes it possible for different groups of people to work on different tasks. It is possible to run some of the work in parallel. For example, as soon as some authentic background data is collected and analyzed, it is possible to start working on both the user model and the system model. In parallel to the user and system modeling, attack data may be collected and analyzed to create the attack model.

It is possible to use people instead of a user simulator to create the user actions. It is also possible to use the whole, or parts of the real system instead of a system simulator. In some situations, this may be preferable, especially if the system or user behavior is very complex, and needs to be modeled in great detail. However, there are some disadvantages in using people and real hardware and software in the generation process.

If only simulation programs are used, it is possible to make the data generation process fully automatic. In the DARPA evaluation [HLF$^+$01], they use humans to inject attacks, and a great deal of system hardware and software. They admit that their data generation procedure required much manual administration and attention during the simulation runs.

An advantage of using fully simulated components, is that it has the possibility to become very scalable, both concerning the number of users and the simulation time period. If humans are used to generate background data, each person can only perform the tasks of one or a few simulated users. If we want to use humans to inject attacks, the simulation time can not be much faster than real-time. Also, if real software and hardware is used, these components may limit the simulation speed considerably.

The rest of this section presents and discusses the different steps of the methodology in some detail and describes the final creation of the synthetic log data.

4.2 Data collection

The data we need as a starting point in the data generation method are samples of background data and attacks representative of the anticipated behavior on the target system. The output log data from the target system is the input to the detection system. This is the type of data that should be produced in the synthetic data generation process. Therefore, it is convenient to have samples available of authentic data from the target system. Hopefully, this data also contains information about user and system behavior which is representative. This may be the most valuable source of data for the generation. If too small amounts of authentic data is available, it is also possible to collect data from similar services. In each case, we need to determine if this data is applicable to our service. Even if we have authentic data, these may not be representative, e.g. because the number of users are going to increase or functions in the system are changed before the detection system will be in operation.

Authentic attacks are often not available. Therefore, we need to collect them in other ways. In the intrusion detection area, databases of known attacks are more or less publicly available. In the fraud area, they are often more service specific, and we may need to “invent” possible attack scenarios or adapt known frauds from other types of services to our situation. Collected attacks may be injected into the target system to get corresponding log data. It is important that the log data can be labeled to know exactly which entries corresponds to a specific attack.

It is difficult to know that properties of the collected data corresponds to the environment that the detection system will operate in. On the other hand, this is not a problem specific for synthetic data. If the detection system is trained and tested on data with the “wrong” properties, the detection will not be correct irrespective of the type of input data. We must predict the future behavior of users and hope that the prediction is close enough to reality.

The output from this step may have many different formats. Some of it may be labeled log data on the same format as will be used as input to the FDS. It may also be databases of attacks with only some information about each attack.

4.3 Data analysis

The next step is to analyze the collected data. The goal of this work is to get a picture of how the system is used and how user actions and attacks show up in log data.

One task is to identify classes of users with similar behavior. Exploratory Data Analysis (EDA) [Tuk77] is an existing set of ideas on how to study data sets to uncover the underlying structure, find important variables, detect anomalies etc. We believe that techniques based on those ideas could be used for this purpose. This may include use of visualization tools and / or clustering methods. It is important to use several classes of user behavior to get diversity in the generated data.

Another task is to identify important attack features, i.e. parameters that are useful for detection of the anticipated attacks. For example, to detect an attacker guessing passwords on a system, the number of failed logins within a certain time interval is a useful parameter. Also, system statistics must be examined. For example, response time and amount of network traffic generated by different events may be useful for the system modeling.

The log data generated by the target system should be examined to determine if it is adequate for effective detection. If it is not adequate, it may be necessary to implement additional logging mechanisms and collect new data.

The output from this step is user classes and a number of statistics for different aspects of user, attacker, and system characteristics.

4.4 Profile generation

The next step is to identify important parameters for user behavior in the statistics. One way to identify these parameters is to study the features needed to detect expected frauds. These features must have correct statistical properties in the generated data to be useful for detection. One example of a fraud indicator would be users logging in at night that usually only log in during daytime. In this case, we must preserve the statistical distribution of logins during the day for the simulated users. Also, correlation between parameters may be fraud indicators and must be preserved.

When we know what parameters we have and what properties we want to preserve, collected data is used to find statistical values for them. There are more or less automatic tools available for calculating values for parameters and fitting data to statistical distributions.

Output from this step is files for different user classes containing values for all parameters that are required for the user simulation. A cooperation between this step and the user modeling step is needed to adapt the format of the profiles to fit the user model. For example, there are different ways to model the frequency of a specific event. Either, it can be defined as the expected number of events per time interval, or it can be defined as the expected time until the next event.

4.5 Modeling the user

The user and attacker behavior can be very complex to model. To limit the complexity, we should bear in mind that it is only of interest to simulate actions that will affect log data. It is in general easier to model the behavior of users in a service which is less complex. Behavior profiles are appropriate to use as configuration data to the user simulator. This way, we can easily adapt the simulator to changes in behavior and add new user classes.

There are several ways to model user behavior. One way is to use a finite state machine. This is suggested by Debar et al. [DDWL98] and also used in the DARPA evaluation [LHF+00]. This is a suitable method if user behavior is not too complex or if only a limited number of parameters need to be preserved in the background data. Also, it is possible to generate great amounts of data for many simulated users. However, it may be very time consuming to model users and attacks if the behavior needs to be modeled in great detail.

Another way is to "record" real user sessions and attack sessions and replay them in the system simulator. This method is suggested by Puketza et al. [PZC+96]. If the behavior is complex, this may be a better method than modeling the users with a state machine. The problems are that the system simulator must be advanced enough to handle this type of input and it may be difficult to generate large enough amounts of user sessions.

A third way is to use test suites developed for testing functionality in operating systems. This is suggested by Debar et al. [DDWL98]. The test suites are not representative of real user behavior but generate all sorts of events that may be very infrequent in a real-life system. The advantage of this method is that it does not require much work to implement, and it may be useful for evaluating the false alarm rate when behavior is very irregular. However, it is questionable if this type of data is useful for testing fraud detection systems.

The output of this step is "lists" of user actions in a format suitable to use as input to the system simulator.

4.6 Modeling the system

The simulated system must be able to produce output data similar to that of the target system, where the similarity can be restricted to those features that are necessary for the fraud detection. The simplest way to model the service realistically would be to set up a replica of all system components. This would give us perfectly accurate log data as long as the "users" are behaving realistically. However, it may require a great deal of resources if we want to simulate a large system, and there are other disadvantages as mentioned before.

A more realistic way to model the system would be to make a hardware replica of the central system components, but implement the clients in software. One single machine can simulate numerous clients. This is the method used by DARPA [LHF^{+}00]. There are two major problems with this solution. Firstly, it is not trivial to simulate traffic from many different IP addresses from one machine, but it can be done. Secondly, it is not possible to speed up the simulation time as the client software must send out the traffic in real-time if the rest of the components are to respond realistically.

A third solution would be to implement a model of the service in software. Some of the real service software can be used, but we can experience problems if we want to run the simulation at non real-time speed. The advantages of this solution are that we have full control of the simulation process and can speed up the simulation as fast as the computing platform allows. Furthermore, simulation processes can be quickly and easily implemented for low complexity services. The disadvantage is that it may take some time and effort to study the behavior of the service to be able to implement the required functionality. There is also a risk of oversimplification resulting in significant differences between synthetic and authentic data.

The output of this step is data on the same format as the data produced by the target system.

4.7 Creation of synthetic log data

The basic data generation methodology was presented in figure 1. Figure 2 below shows in more detail how the data generation components interact in the generation process.

The *user profiles* are used as input to the *user and attacker simulator*. The *user and attacker simulator* generates user actions that are fed to the *system simulator*. The *system simulator* creates the final synthetic test data and the *configuration data for the user and attacker simulator* contains information that controls the generation of normal user and attack actions. For example, it may be the start time and stop time for simulation, the number of normal users from different user classes, the number of attacks of each type, and the start time and stop time for different attacks. The *configuration data for the system simulator* contains for example information about the number of clients that should be simulated, statistics about reply times and traffic amounts for different events, and behavior in the presence

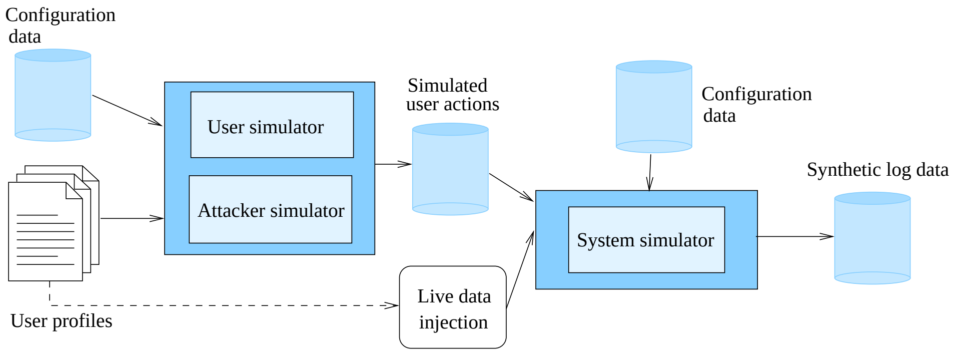

Figure 2: The synthetic log data generation process

of certain attack events. Parameters that we want to vary between different simulation runs should be included in the configuration data. This means that we can generate batches of synthetic data for different purposes without reprogramming the simulators.

It is possible to complement the user simulation with *live data injection*, e.g. to generate more complex series of actions or to generate deviating behavior for special stress tests.

5 Conclusions

The results presented in this paper form a foundation for generation of synthetic test data based on authentic data for fraud detection systems. By starting out from small sets of authentic log data, appropriate synthetic log data can be created in large amounts. Properties of parameters in the initial data are preserved or can be tailored to meet the needs of testing and training of fraud detection systems. In the near future we aim to use our method in practice with authentic log data collected from a pilot-test video-on-demand application.

References

[BSTM+97] Peter Burge, John Shawe-Taylor, Yves Moreau, Bart Preneel, Christof Stoermann, and Chris Cooke. Fraud Detection and Management in Mobile Telecommunications Networks. In *Proceedings of the European Conference on Security and Detection ECOS 97*, pages 91–96, London, April 28-30 1997. ESAT-SISTA TR97-41.

[CFPS99] Philip K. Chan, Wei Fan, Andreas L. Prodromidis, and Salvatore J. Stolfo. Distributed Data Mining in Credit Card Fraud Detection. *IEEE Intelligent Systems*, 14(6), Nov / Dec 1999.

[CPOM95] Mandy Chung, Nicholas J. Puketza, Ronald A. Olsson, and Biswanath Mukherjee. Simulating Concurrent Intrusions for Testing Intrusion Detection Systems: Parallelizing Intrusions. In *Proceedings of the 1995 National Information Systems Security Conference*, pages 173–183, Baltimore, Maryland, October 10-13 1995.

[dar01] DARPA Intrusion Detection Evaluation. http://www.ll.mit.edu/IST/ideval/, July 2001. The main web page for the DARPA evaluation experiments.

[DDWL98] H. Debar, M. Dacier, A. Wespi, and S. Lampart. An Experimentation Workbench for Intrusion Detection Systems. Technical Report RZ2998, IBM Research Division, Zurich Research Laboratory, Zurich, Switzerland, mar 1998.

[HLF$^+$01] Joshua W. Haines, Richard P. Lippmann, David J. Fried, Eushiuan Tran, Steve Boswell, and Marc A. Zissman. 1999 DARPA Intrusion Detection System Evaluation: Design and Procedures. Technical Report Technical Report 1062, MIT Lincoln Laboratory, February 2001.

[HRLC01] Joshua Haines, Lee Rossey, Rich Lippmann, and Robert Cunnigham. Extending the 1999 Evaluation. In *Proceedings of DISCEX 2001*, Anaheim, CA, June 11-12 2001.

[Jam01] The JAM project homepage. http://www.cs.columbia.edu/ sal/JAM/PROJECT/, July 2001.

[Ken99] Kristopher Kendall. A database of computer attacks for the evaluation of intrusion detection systems. Master's thesis, MIT, 1999.

[KLJ00] Håkan Kvarnström, Emilie Lundin, and Erland Jonsson. Combining fraud and intrusion detection - meeting new requirements. In *Proceedings of the fifth Nordic Workshop on Secure IT systems (NordSec2000)*, Reykjavik, Iceland, October 2000.

[LHF$^+$00] Richard Lippmann, Joshua W. Haines, David J. Fried, Jonathan Korba, and Kumar Das. The 1999 DARPA off-line intrusion detection evaluation. *Computer Networks*, Volume 34(Issue 4):579–595, October 2000. Elsevier Science B.V.

[McH00] John McHugh. The 1998 Lincoln Laboratory IDS Evaluation: A Critique. In *Recent Advances in Intrusion Detection, Third International Workshop, RAID 2000, Toulouse, France, 02-04 October 2000, pages*, pages 145–161, Toulouse, France, October 2-4 2000. Lecture Notes in Computer Science #1907, Springer-Verlag, Berlin.

[MT00] Roy A. Maxion and Kymie M.C. Tan. Benchmarking Anomaly-Based Detection Systems. In *International Conference on Dependable Systems* *and Networks*, New York, New York, June 2000. IEEE Computer Society Press.

[PZC$^+$96] Nicholas J. Puketza, Kui Zhang, Mandy Chung, Biswanath Mukherjee, and Ronald A. Olsson. A methodology for testing intrusion detection systems. *Software Engineering*, 22(10), 1996.

[SFP$^+$98] Salvatore Stolfo, Wei Fan, Andreas Prodromidis, Wenke Lee, Shelly Tselepis, and Philip K. Chan. Agent-based Fraud and Intrusion Detection in Financial Systems. Technical report, 1998. Available at: http://www.cs.columbia.edu/ wfan/research.html.

[Tuk77] John W. Tukey. *Exploratory Data Analysis*. Addison Wesley College, 1977.

Paper C

Synthesizing Test Data for Fraud Detection Systems

Reprinted from

Proceedings of Nineteenth Annual Computer Security Applications Conference
Las Vegas, Nevada, USA
December 8-12, 2003

Synthesizing Test Data for Fraud Detection Systems

Emilie Lundin Barse Håkan Kvarnström[*] Erland Jonsson

{*emilie,hkv,erland.jonsson*}*@ce.chalmers.se*
hakan.kvarnstrom@teliasonera.com
Department of Computer Engineering
Chalmers University of Technology
412 96 Göteborg, Sweden

Abstract

This paper reports an experiment aimed at generating synthetic test data for fraud detection in an IP based video-on-demand service. The data generation verifies a methodology previously developed by the present authors [LKJ02] that ensures that important statistical properties of the authentic data are preserved by using authentic normal data and fraud as a seed for generating synthetic data. This enables us to create realistic behavior profiles for users and attackers. The data can also be used to train the fraud detection system itself, thus creating the necessary adaptation of the system to a specific environment. Here we aim to verify the usability and applicability of the synthetic data, by using them to train a fraud detection system. The system is then exposed to a set of authentic data to measure parameters such as detection capability and false alarm rate as well as to a corresponding set of synthetic data, and the results are compared.

Keywords: fraud detection, synthetic test data, data generation methodology, user simulation, system simulation, neural networks

1 Introduction

Fraud detection is becoming increasingly important in revealing and limiting revenue loss due to fraud. Fraudsters aim to use services without paying or illicitly benefit from the service in other ways, causing service providers financial damage. To reduce losses due to fraud, one can deploy a fraud detection system. However, without tuning and thorough testing, the detection system may cost

[*]Håkan Kvarnström is also with TeliaSonera AB, SE-123 86 Farsta, Sweden

more in terms of human investigation of all the false alarms than the gain from reduction of fraud. Test data suitable for evaluating detection schemes, mechanisms and systems are essential to meet these requirements. The data must be representative of normal and attack behavior in the target system since detection systems can, and should, be very sensitive to variations in input data.

Using synthetic data for evaluation, training and testing offers several advantages over using authentic data. Properties of synthetic data can be tailored to meet various conditions not available in authentic data sets. There are at least three application areas for synthetic data. The first is to train and adapt a fraud detection system (FDS) to a specific environment. Some FDSs require large amounts of data for training, including large amounts of fraud examples, which are normally not available in the authentic data from the service. The second application area is to test the properties of a FDS by injecting variations of known frauds or new frauds into synthetic data to study how this affects performance parameters, such as the detection rate. The false alarm rate may also be tested by varying background data, where background data is defined as normal usage with no attacks. The third application area is to compare FDSs in a benchmarking situation.

The aim of this work is to test the feasibility of generating and using synthetic data for training and testing a fraud detection system. Our synthetic data generation is based on the method proposed in [LKJ02], where we use small amounts of authentic log data to generate a large amount of synthetic data. The method identifies important statistical properties and aims to preserve parameters important for training and detection, such as user and service behavior. We apply the data generation method on an IP based video-on-demand (VoD) service running in a pilot test environment with real customers. The authentic data collected from the service are used to generate synthetic log files containing both normal and fraudulent user behavior. The properties of the synthetic log files are verified by visualizing them and using them to train and test a fraud detection prototype.

Synthetic data are not commonly used in the fraud detection area, although, the use of manipulated authentic data is discussed in some papers, e.g. [CFPS99] and [BSTM+97]. In the intrusion detection area, more work has been done using synthetic test data. ([KLJ00] discusses similarities between fraud and intrusion detection systems.) Huge amounts of synthetic test data were generated in the 1998 and 1999 DARPA intrusion detection evaluations [HLF+01]. Debar et al. [DDWL98] developed a generic intrusion detection testbed, and suggest the use of a finite state automata to simulate user behavior. While this methodology is rather close to our approach, they did not use this method in their testbed. They declared it practical only if the set of user commands is limited. Instead, they used recorded live data from user sessions. Puketza et al. [PZC+96] describe a software platform for testing intrusion detection systems where they simulate user sessions using the UNIX package *expect*. However, none of these methods give sufficient control of the data properties of the synthetic data. We believe that our method can provide better data properties that are needed for training and testing in many situations. Furthermore, our method provides scalability of log data in both amount of users and time period.

Some interesting work has been done on measuring characteristics in data and how they affect the detection systems, e.g. Lee and Xiang [LX01], and Tan and Maxion [TM03]. In [MT00], Maxion and Tan generate “random” synthetic data with different degrees of regularity and show that it affects the false alarm rate drastically. The methods proposed in these papers may be useful for synthetic data validation, but have not been used in this paper.

Below, Section 2 and 3 give a a summary of the method proposed in [LKJ02]. The rest of the paper describes how we apply the method on the VoD service and the verification of the generated synthetic data.

2 Motivation for using synthetic data

This section gives the benefits of using synthetic data for various types of testing and explains why authentic data are not a solution in some cases.

2.1 Why not use authentic data?

Authentic data cannot be used in some cases for a number of reasons. The target service may still be under development and thus produce irregular or only small amounts of authentic data. We also have no control over what fraud cases the data contain. Furthermore, it may be impossible or at least very difficult to acquire the amount of or type of data needed for tests. This in turn may be due to the fact that only a limited number of users are available or that we do not know whether the data set contains any frauds.

2.2 Benefits of synthetic data

Synthetic data can be defined as data that are generated by simulated users in a simulated system, performing simulated actions. The simulation may involve human actions to some extent or be an entirely automated process.

Synthetic data can be designed to demonstrate certain key properties or to include attacks not available in the authentic data, giving a high degree of freedom during testing and training. Synthetic data can cover extensive periods of time or represent large numbers of users, a necessary property to train some of the more ”intelligent” detection schemes.

In [LKJ02], we provide an elaborate discussion of data properties that are important for training and testing fraud detection systems. These are summarized in the following bullets:

- Data need to be labeled, i.e. we need to have exact knowledge of the attacks included in the data.
- The attacks found in the input data are representative of the attacks we expect to find in the target system. (Not necessarily the same attacks that currently occur in the system.)

- The number and distribution of attacks in the background data (fraud / normal data ratio) must be adapted to the detection mechanism. Some detection methods perform better if they are trained with data in which attacks are overrepresented.
- The amount of data must be of a sufficient size. In particular, certain AI algorithms need huge amounts of training data to perform well.
- For testing, the number and distribution of attacks in the background data (fraud / normal data ratio) should be realistic.
- For testing, it is important that attacks in the input data are realistically integrated in the background data. For example, the time stamp of an attack, time between attacks and the time between parts of an attack may affect detection results.
- Normal (background) data should have similar statistical properties as authentic data from the target system. Different behavior in the system may drastically affect detection performance.

3 Data generation methodology

Synthetic data were generated by the methodology described in [LKJ02]. For completeness, we briefly introduce the method and refer to the original work for details.

The main components necessary for automating the data generation process are specifications of desired user behavior in the system, a user / attacker simulator and a system simulator. The goal of the methodology is to guide the production of these components. The starting point is the collection of information about the anticipated user behavior in the target system. The methodology includes generating both background and attack data and thus it is necessary to have information about possible attacks and normal usage. These data serve as the basis for user and system modeling.

Figure 1 illustrates the methodology. The first step is the *collection of data* that should be representative of the anticipated behavior of the target system. Data may consist of authentic background data from the target system, background data from similar systems, authentic attacks and other collections of possible attacks. The second step is to *analyze the collected data* and identify important properties, such as user classes, statistics of usage, attack characteristics and statistics of system behavior. In step 3, the information from the previous step is used to identify parameters that must be preserved to be able to detect the anticipated attacks and to *create user and attacker profiles* that conform to the parameter statistics. A *user model* is created in step 4. This must be sophisticated enough to preserve the selected profile parameters. Attackers are also modeled in this step. The user and attacker simulators implement the models. The system is modeled in step 5, and this model must be accurate enough to produce equivalent log data as the target system for the same type of input user actions. The system simulator is then implemented according to this model.

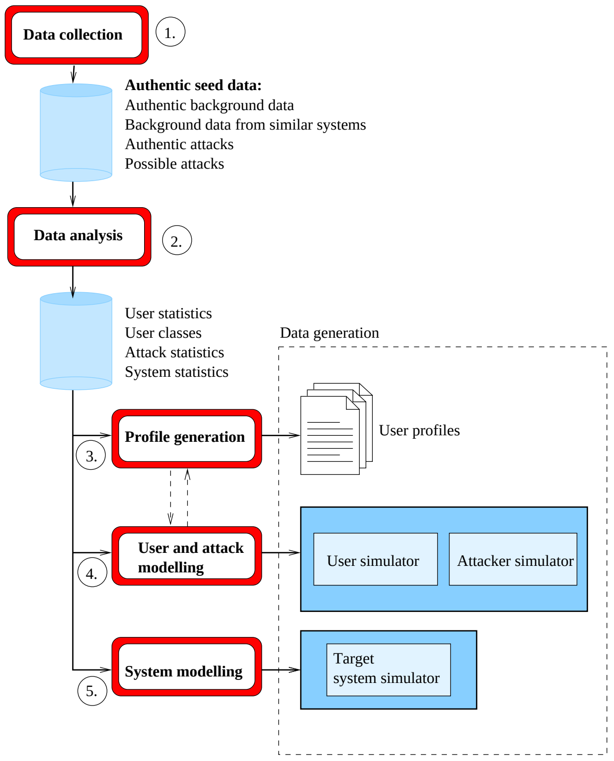

Figure 1: Synthetic data generation method

It is possible to use people instead of a user simulator to create user actions and to use the whole or parts of the real system instead of a system simulator. This may be preferable in some situations, e.g. if the system or user behavior is very complex and needs to be modeled in great detail. In our experiments, we used humans to mimic fraudulent behavior and automata to generate normal background data.

4 Authentic data

This section describes how we implemented the collection of authentic data for the VoD service. The next section (Section 5) presents the work of designing, implementing and using the data analysis, user and system model.

4.1 The target Video-on-Demand system

The VoD service was in pilot operation and had a limited number of users. It could thus provide only small amounts of log data. This system consisted of a number of components, shown in Figure 2. Each user had a set top box (stb) at home, which was connected to the Internet via a fast xDSL connection. When the set top box was turned on, it automatically contacted the service providers DHCP server (Dynamic Host Configuration Protocol server) to get a dynamic IP address. Then, the user could contact the application server, login to it, browse the video database and order a movie. The application server generated an authentication ticket for the user. The VoD server then started delivering the chosen movie after verification of the ticket.

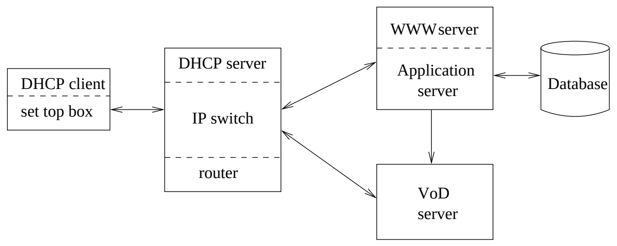

Figure 2: VoD system components

In the VoD-system, a total of 12 ”test users” were active during the service development. Thus, the amount of data generated was not sufficient for training the fraud detection modules. We used these twelve users as an approximation of ”normal” users. They had no knowledge about the implementation of the service or of the fact that their behavior would be used for synthetic data generation.

4.2 Collection of the authentic seed data

We had the opportunity to work together with the development team for the VoD service and could therefore specify the information collected in the log files. To be able to decide what information to collect, we created a database with expected frauds and the indicators or features needed to detect them. Each indicator was analyzed to find out what type of log data was needed to catch the indicator.

The test users of the VoD system were considered ”friendly users” and had been selected as test pilots based on their physical location. All users were known to the development team and thus no frauds were expected to occur. These users generated the background data we collected.

We used employees that acted according to descriptions of the expected fraud cases we had created. These fraud cases were “injected” into the system by using two dedicated set top boxes in the same way as we expect a fraudster to use it.

Data were collected over a period of about three months and the total amount of data was 65 MBytes.

4.3 Fraud data

Four fraud cases were "injected" into the authentic data:

Break-in fraud: The fraudster has "taken over" the identity of a legal user by hacking the user's set top box. The real user may use the service without knowing that he had a break-in. An indication of a break-in could be a sudden excessive usage of the service.

Billing fraud: To avoid paying for the movies ordered, the fraudster has hacked the billing server. In our experiments this was done by removing billing records in the application server log file.

Illegal redistribution fraud: This means that a customer receiving a movie in the VoD service transmits it to other people not paying for the service. Our case of illegal distribution was performed by uploading large files to a computer on the Internet some time after the completion of a movie download. Illegal redistribution of content could also be an indication of an external break-in in the set top box where the attacker in turn uses (and downloads) videos at the expense of a legitimate paying customer.

Failed logins: Several failed login attempts were added with "dummy" user ID's to imitate the behavior of people trying to guess passwords.

4.3.1 Example of a fraud case

This subsection gives an example of the process we used for each fraud case to decide what the fraud indicators were and thus what information we needed in the log files.

Fraud indicators for the fraud case *illegal (re)distribution of services* may be:

- The ratio between transmitted and received data is suspiciously high.
- A great deal of data is transmitted (some period of time) after data has been received.
- A great number of downloads are done.

For the first indicator we need user ID, IP address, bytes transmitted and bytes received in the log data. The second indicator requires IP address, bytes received, bytes transmitted and time stamp. For the third indicator we need user ID, IP address, time stamp, session ID, billing data and bytes received.

Thus, we found that the following information needed to be in the log data: (1) Data from the DHCP server containing information about when a user has his set top box switched on and off; (2) Router statistics in which the number of bytes to and from a user is registered; (3) Data from the application server containing time stamp, user ID, IP address, session ID etc. of user logins; (4) Detailed information about movie orders from the application server; (5) Billing information generated by the billing system; (6) VoD server information about what content was actually delivered to a certain user. These raw data records from the service components were converted to a common format.

5 Generation of synthetic data

The goal of the synthetic data generation was to obtain enough data of sufficient quality to be able to develop and train a fraud detection prototype for a VoD service.

5.1 Data analysis and profile generation

The seed data were analyzed and we identified a number of parameters we considered important, which we then included in the user profiles. The importance of the parameters was evaluated by studying the features needed to detect the identified template fraud examples. We also selected the parameters necessary to determine statistical values for the transitions in the state machine used to model user behavior.

The idea was to sort users into a number of classes based on their behavior. Each class of users has one set of parameter values. Because of the limited number of authentic users, it was difficult to find natural user classes without letting each user form its own user class. This, and the project time limits, resulted in the grouping of all normal users into a single class, which is a coarse simplification of normal user behavior. The frauds affecting user behavior were assigned to their own classes. Normal user behavior and fraudulent user behavior can both be scaled up during the generation process. This allows us to vary the ratio between fraudulent and normal data for our test cases.

A *perl* program parsed the log files and collected data for each transition in separate files. Small *matlab* programs plotted these data and fitted different probability distribution functions to the data. In our first data generation attempt we used the normal probability distribution function to model the time intervals for the transitions. It was obvious when the data were plotted that most of our data sets had the shape of an exponential or gamma distribution function. A chi-square test was used to check which distribution had the best fit. An example of a plot of transition time and the corresponding probability distribution function is shown in Figure 3. This histogram shows the authentic transition times from login to movie order, and the dashed line shows the corresponding probability distribution, from which we get the simulated transition times.

The statistical profile contains the probability for each transition and a distribution function with fitted parameters for the transition time.

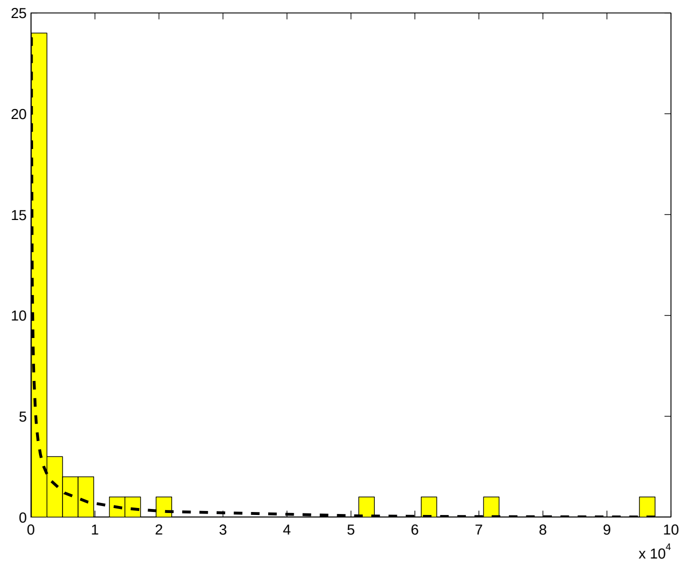

Figure 3: Histogram of time from login to order (in seconds)

Statistics needed for the simulation of the system were also analyzed. For example, traffic for different events, the behavior of the DHCP server and delays were studied. These statistics were collected in a configuration file for the system simulation program.

5.2 Generation of synthetic data

The implementation of the user and system simulation is a trade-off between simplicity and realism. This process may be iterative, where a simple version is first implemented and validated to identify whether a sufficient number of the properties considered important are preserved. If this is not considered good enough, some level of detail can be added.

In our experiments, both the user and system models were implemented as an event-driven simulations in *perl*, requiring about two thousand lines of code each.

User simulation. The goal of the user simulation is to produce a chronological list of actions for each simulated user. What makes these actions realistic is the accuracy of the statistical profiles used as input and level of detail in the user model implementation. Our simulation used a finite state machine to mimic user behavior (Figure 4).

In state 0, the user's set top box (stb) is switched off. The only thing he can

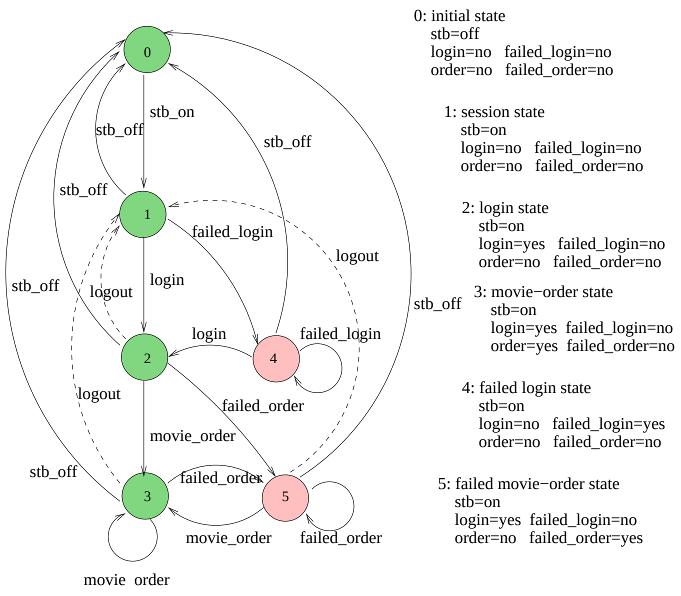

Figure 4: User state machine

do in this state is to turn the set top box on and enter state 1. In state 1, he can try to log in. If it is a successful login, he is transferred to state 2, where he can start to order movies. If he fails to log in, he is transferred to state 4, where he can continue to make failed logins until he succeeds or gives up and switches the set top box off. In state 2, the user can make movie orders. If he fails he is transferred to state 5, where he can continue to try to order movies until he succeeds, logouts, or switches the set top box off. This models the actions that affect output log data in the service. The statistics derived in the analysis of authentic data are used to set the probabilities of transitions to different states and the time between the entering of one state until a certain transition is done.

Fraudulent users do not always follow this model. For example, the break-in fraudsters skipped the login procedure and were able to order movies without successful authentication. This required some extra parameters and “fraud flags” in the user simulation program. Other types of deviant behavior may require further additions to the program, even though this was the only modification we found necessary in these experiments to mimic the behavior of the injected frauds.

With this simple automata we could simulate a large number of users and attackers covering extensive periods of time. Despite its simplicity it provided sufficiently detailed user actions for the next step, system simulation.

Video-on-demand system simulation. The actions of the virtual users serve as input to the simulated target system to generate synthetic data. We used statistics from the authentic data (e.g. data volumes and delays in networks and components) to configure the simulation. A further option was added to control the generation of billing records in order to simulate fraudsters that circumvent the billing subsystem.

We used an event-driven simulation. Each user action in the input data was first fed to the set top box module. From there it was added to the simulation list to be passed on to the right component module after a configurable delay. The possible user events were *turn on set top box*, *turn off set top box*, *login*, and *movie order*. Figure 5 illustrates the resulting event flow in the simulated system when it is fed with a login action. The login event is passed from the set top box to the access node (IP switch) via the simulation list ("sim" in the figure). From the access node, the event is passed to the router module, which calculates router statistics for the event, and also to the application server, where the login is done. The success of the login is determined during the user simulation and thus included in the input user action. The application server generates an authentication log entry (AuthenticationNotification) of the same format as that of the real system. In this way all events are passed around in the system, and the proper components generate log messages when events arrive.

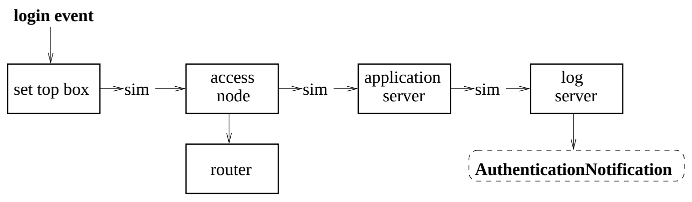

Figure 5: Flow in simulated system at a login action

Our simulation model is a coarse approximation of the real system environment. We do not model network behavior at packet level, for example, as the router statistics are only measured over a time interval. Other shortcomings are described in Section 8.

6 Input data for the detection experiment

We verified the generated data by using them to develop and train a fraud detection prototype, i.e. according to the first application area mentioned in Section 1. This section describes the data used in the experiment.

6.1 Authentic data

The authentic data needed for the detection experiments are the data created by authentic fraud users. These users have long periods of rather normal behavior and occasional fraud sessions. The frauds are described in Section 4.3. Since we had very few authentic fraud sessions, we had to use the same data set here as we used as seed data for the synthetic data generation process.

6.2 Generated synthetic data

The simulation described in Section 5, resulted in synthetic data for seven months containing six hundred “normal” users. Three fraud types, out of the four described in Section 4.3, were simulated. We used 100 break-in fraudsters, 100 fraudsters doing illegal redistribution, and finally 100 cases of fraudsters that hacked the billing server.

This resulted in several GBytes of synthetic raw log data in unpacked text format. The raw log data were converted to a common format where data were grouped into sessions in the same way as the authentic log data were converted. This resulted in about 80 MB of log data for each group of 100 users. It was possible to generate log data in less than a day on a normal PC for 100 users acting under a period of seven months.

6.3 Subjective assessment of data

To form a subjective concept of the simulated data and try to estimate its validity, we did a great deal of visualizations of the authentic and synthetic users. Below we show some of the plots produced in this process.

Normal users. The plots show the events for a user for a period of 15000 minutes (about 10 days). The first two plots (Figure 6 and 7) show the behavior of two different authentic users. The next plot (Figure 8) shows the behavior of one of the synthetic users.

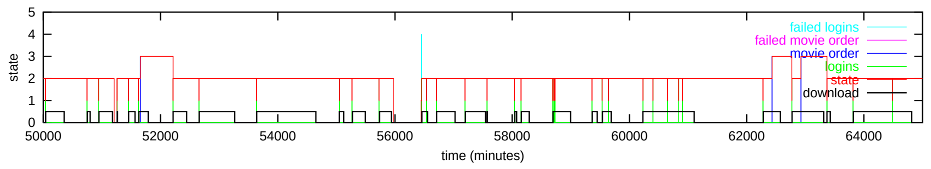

Figure 6: Plot of authentic user (user ID 0.123)

The scale of the x-axis is in minutes (from the start date of the log files). The y-axis shows which state the user currently is in. These numbers correspond to

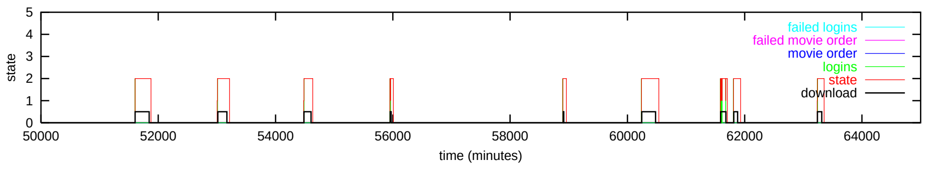

Figure 7: Plot of authentic user (user ID 0.128)

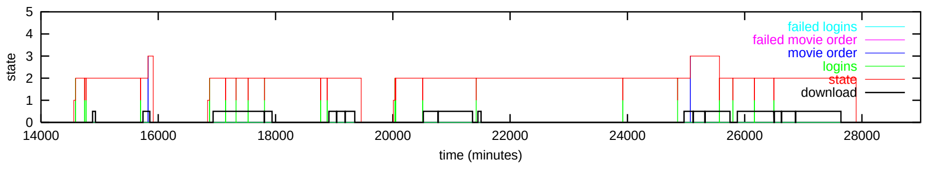

Figure 8: Plot of synthetic user (user ID 2.137)

the states in the user model (see Figure 4). The dotted red line (or grey if viewed in black-and-white printing) follows the user state.

level 1.0: User has switched on the set top box
level 2.0: User is logged into the application server
level 3.0: User has ordered a movie
level 4.0: User has made a failed login
level 5.0: User has made a failed movie order
Finally we use “level 0.5” to show downloading or uploading sessions (does not correspond to any user state).

The fourth and fifth levels are used less frequently since these correspond to failed logins and failed movie orders, which are not common. The events that cause the user to switch states are plotted as spikes in the graph, with the height of the state the user is entering, i.e. logins with a height of 2.0 and orders a height of 3.0.

The user in Figure 6 is very active. His set top box is switched on most of the time, several days in a row. The plot shows two sessions. In the first session he waits a short time before logging in. After some hours he orders a movie. Shortly after the movie order, download starts and he continues to download data for several hours. Then the session continues with several more logins and downloads. It seems that the user is logged out automatically after some time, but this is not possible to see in the log files. Downloads often start at login, and downloads are done even when there are no movie orders. The reason for this is that users can also watch free TV channels, and that they probably watch movie trailers before ordering a movie.

On the opposite end, the authentic user in Figure 7 is not an avid user of the service. He has several shorter sessions but no movie order. Still, downloading is going on during the sessions. The short sessions indicate that he always switches the set top box off when not watching TV. Here the downloads starts regularly, and a detailed examination revealed that all these sessions occur in the evenings, lasting less than a couple of hours. The conclusions that can be drawn from these plots are that the first user seems to be home during daytime, he has irregular TV habits, and leaves the set top box switched on even when it is not used. The second user probably has a more ordered life, works during daytime, watch TV in the evenings, and sleep at night.

Since we use only one normal user profile in the simulation, the synthetic user in Figure 8 should behave like a medium active user , which seems to be the case. Hence, this synthetic user also has very similar behavior to the other synthetic users. He has rather long sessions, but not as long as the first authentic user. He also orders fewer movies and downloads data less often. However, it seems that many of the download sessions are comparatively long, which may be an implementation mistake in the simulation.

Fraud users. An example of a break-in fraudster is shown in Figure 9. In this figure, the fraud behavior is not obvious. However, in the next Figure (10), only the logins and orders are shown for the break-in fraud user. Here we can see that there are orders in the sessions before the user has logged in. Comparing this behavior to that of a normal user, we see that there is always a login before an order in a normal session.

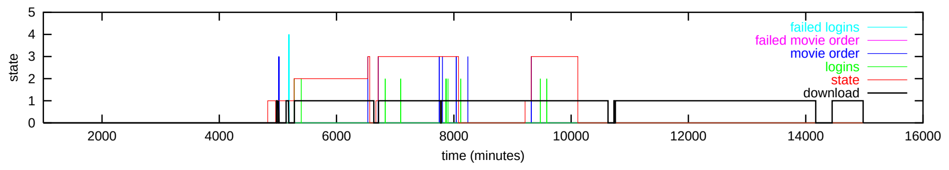

Figure 9: Plot of break-in fraudster in authentic data (user ID 0.131)

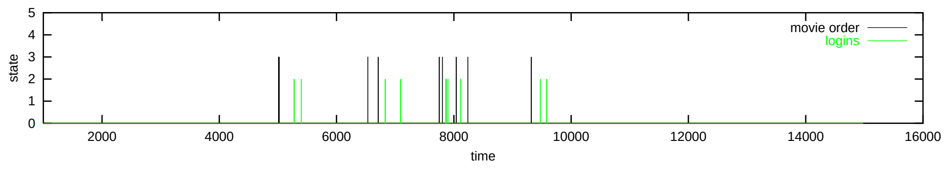

Figure 10: Plot of break-in fraudster showing logins and orders (user ID 0.131)

The billing fraud plots looks very much like those of normal authentic and synthetic users. The absence of billing records cannot be seen in these plots, and therefore we do not show the plots here.

A mistake was made in the realization of the illegal redistribution fraud in the authentic data which caused the router log file to be empty during this period. This meant that we did not have complete authentic seed data for this type of fraud. Since we wanted our detection system to be able to detect also this type of fraud, which was considered an important type, we decided to “manufacture” an illegal redistribution profile manually. The profile reflected the planned behavior, i.e. it contained traffic download after orders and a great deal of upload traffic some hours after the download. We used this to perform detection tests using a second set of data. However, since we could not test this case against authentic data our results are of limited value and we will address this fraud case further in this report.

It should be noted that all types of frauds that we wish to trigger our detection mechanism can be created “manually” in this way, by editing the statistical profile. This can also be used for testing the detection capability of a fraud detection system when frauds or background traffic are varied.

7 Detection experiments

The fraud detection system used in the experiments was a neural network with an added ability to handle temporal dependencies. The neural network was trained with synthetic data containing about 25% fraudulent and 75% normal behavior. The detection capability of the neural network was then tested with authentic data containing frauds. The detection results with authentic data were compared to detection results with test sets created from the synthetic fraud data to find out how much they differ.

7.1 The neural network

The neural network was trained using synthetic data and was then used to detect the attacks existing in the authentic data. A feed-forward neural network model was used, consisting of seven inputs, a single hidden layer with seven nodes, and a single output giving indications of fraud. The net was trained independently for each fraud type, thus requiring only a single output bit. The initial input weights were randomized before training to prevent the network from finding local minima. The neural network model was implemented in C with approximately 900 lines of code.

As conventional neural network architectures and models are not well suited for patterns that vary over time, an exponential trace memory was used [Mos01]. The memory can be viewed as a buffer maintaining a moving average (exponentially weighted) of past inputs: $\bar{x}_i(t) = (1 - \mu_i)x_i(t) + \mu_i\bar{x}_i(t-1)$. The moving average is calculated for all input parameters over a configurable time interval grouping events together. The configurable μ_i allows for the representation of averages spanning various intervals of time. The higher the value of μ_i, the fewer the current input patterns influenced by input from previous intervals. We used $\mu_i = 0.7$ in our detection tests, which proved to provide a sufficient decay rate for our purposes.

7.2 Neural network input

The sum of all input events (over an interval) was fed to the inputs of the neural networks. For simplicity, we used an interval of 1440 minutes (24 hours). This allowed our detection scheme to detect fraud with a granularity of 24 hours, which should be sufficient for most service environments using fraud detection. A finer granularity could have been chosen but would not have been useful, as most users only order a few movies over a period of this length. The input events were assigned to the neural network's inputs as follows:

1. Sum of successful login attempts
2. Sum of failed login attempts
3. Sum of successful movie orders
4. Sum of failed movie orders
5. Sum of movie delivery notifications
6. Sum of billing notifications
7. Ratio between uploaded and downloaded number of bytes [(dl / (1+ul)) / 1000]

The values were not normalized as most input values were roughly within the same range. An exception was input 7, which was divided by 1000 to fall within a range of magnitude similar to that of the other input values.

Parsing audit records, handling intervals and preparing audit data according to the network inputs were done using a *perl* script (approximately 500 lines of code).

7.3 Detection of billing fraud

Synthetic data containing occurrences of billing fraud (see Section 4.3) were used to train the neural network. Thirty-five synthetic users active in a total of 3,000 (1,440 minutes) intervals were randomly selected. Of these intervals, 2,023 were non-fraudulent and 976 contained fraudulent activities. In the plots below, we define *the epoch* as the time interval containing the first available log entry. The network was trained for 1,000 rounds, selecting input values from random intervals. One thousand rounds of training took approximately 15 minutes on 350 MHz Silicon Graphics O2+ having 650 MB of memory.

Figure 11 shows detection tests using a second set of synthetic data. Intervals containing fraudulent activities were successfully detected with few errors. A few occurrences of *false negatives* were found, which were caused by periods of user inactivity, i.e. the user did not order any movies during the interval, which could be perfectly normal even for a fraudulent user. In addition, at the end of

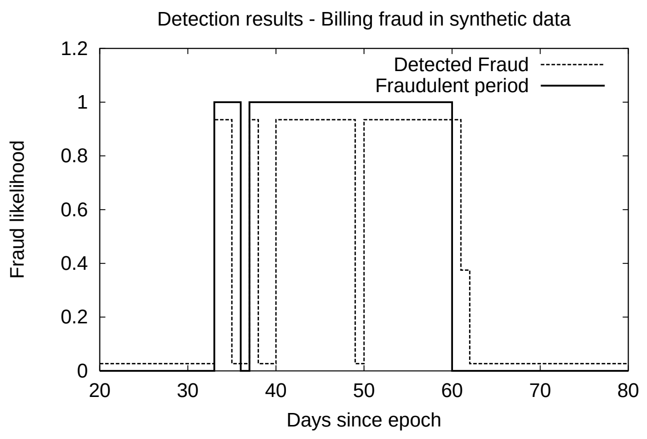

Figure 11: Detection of Billing fraud in synthetic data

the fraud periods, *false positives* were observed for one to two intervals. This was caused by the memory that handles temporal correlation of data.

If a significant number of events are processed during a few intervals, the trace memory could contain enough history data to trigger an alarm. This should not pose a problem in most cases, as the error follows a period of actual fraud. The value of μ_i, controlling the decay rate of the trace memory, plays a role in controlling false positives and negatives. A fast decay rate would lead to more false negatives while a slower decay rate would lead to a higher number of false positives at the end of fraudulent periods.

Next, the events contained in the authentic data (with labeled occurrences of fraud) were fed to the network. The data were known to contain billing frauds for a single user between May 18 and June 14 (day 10 - day 30 in Figure 12). The results are shown in Figure 12, which clearly shows that our neural network detection model works as trained.

The problem of false positives at the end of a fraudulent period remains, but does not prevent a security officer from successfully identifying the occurrence of those fraudulent activities, which can then be further investigated.

7.4 Detection of break-in fraud

The break-in frauds mimic a user who breaks into a customer's set top box and seriously alters the user's usage behavior (e.g. by ordering a substantially higher number of movies over some periods). As previously mentioned, break-in frauds were simulated by a team of "fraudsters", and from their actions synthetic data containing fraudulent users were generated and used for training and testing. Again, 35 synthetic users were chosen that were active in a total of 3,000

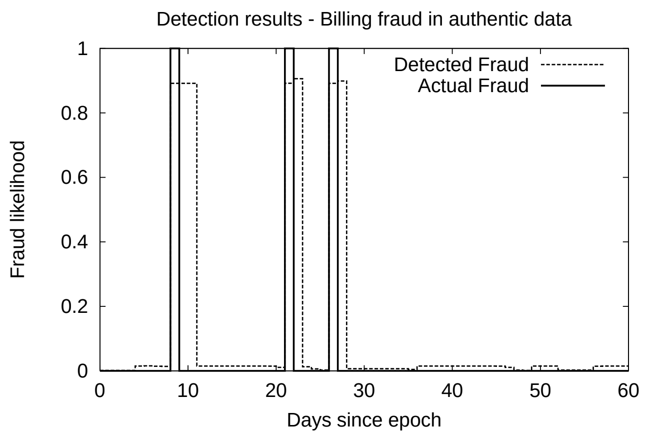

Figure 12: Detection of Billing fraud in authentic data

intervals. Of these, 2,016 were non-fraudulent and 984 contained fraudulent activities. Similar to the training in billing fraud, the network was trained for 1,000 rounds with random input values.

Figure 13 shows the detection results of a second set of synthetic data. A nearly perfect match is achieved in the interval shown. However, two intervals show false negative behavior, one in the beginning of the fraudulent period and one at the end. Again, this does not seriously affect the usefulness of the detection system.

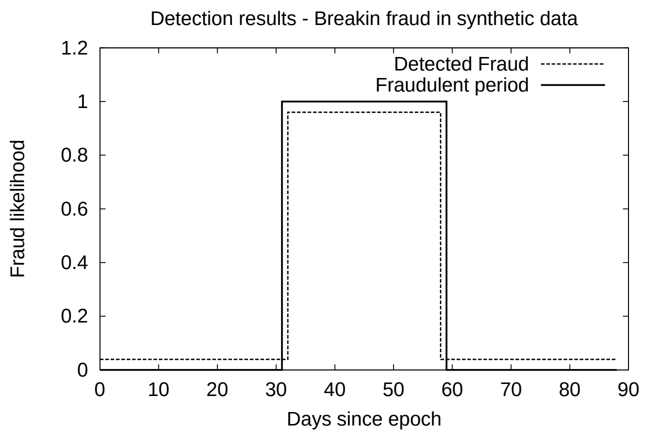

Figure 13: Detection of break-in fraud in synthetic data

The detection of break-in frauds in authentic data is shown in Figure 14. The result is not as promising as in the synthetic data. In addition to the period of actual fraud (days three to six), several false positives are shown at various intervals (approximately at days 33, 62, 68, 71, 75, 81, 85). An investigation of the reason for this showed that the fraudulent user did not deviate a great deal in consumption compared to the ”normal” user. This made it difficult for the neural network to distinguish between normal and fraudulent behavior.

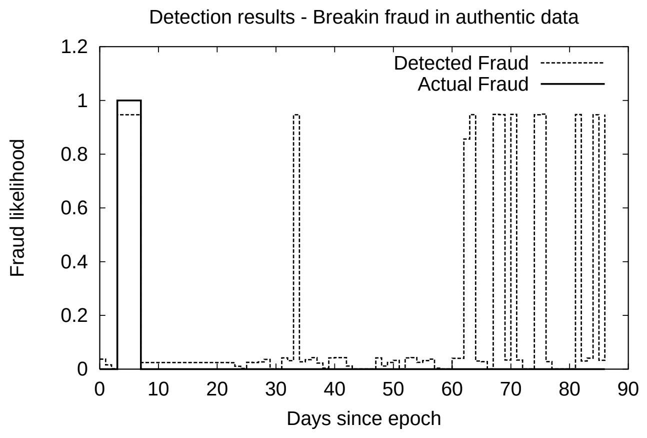

Figure 14: Detection of break-in fraud in authentic data

The average numbers of movie orders per day are plotted in Figure 15. The fraudulent period (day 4-7) did not exceed forthcoming consumption enough to allow the network to successfully differentiate fraudsters from normal users. Clearly, our team of ”fraudsters” did not succeed in their job of ordering an excessive amount of movies during that time period. The false positives were also affected by the temporal memory, as the false alarms were preceded by quite a few days of moderate usage which together, over time, built up sufficiently large input values for the neural networks to trigger an alarm. Once again, this illustrates the value of carefully balancing the decay rate of the memory function for each type of fraud.

7.5 Quantitative results

Table 1 shows the detection results. The *Sensitivity* and *Specificity* are shown in the first two columns of the table. *Sensitivity* [true pos / (true pos + false neg)] shows to which extent fraudulent activity is classified correctly. If detection is high in false negatives, the sensitivity becomes poor. The *Specificity* [true pos / (true pos + false pos)] indicates the degree of misclassification of non-fraudulent events. If a test shows high a false positive value, the specificity becomes poor. A goal of

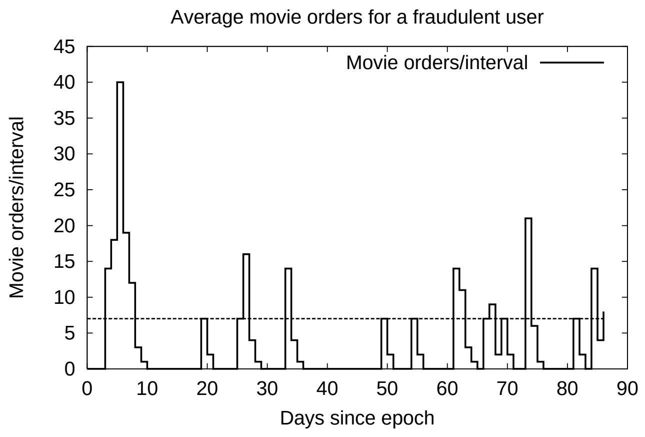

Figure 15: Movie order averages for a fraudulent user (authentic data)

Table 1: Detection results

Fraud	*Sensitivity*	*Specificity*	*# of periods*	*True pos.*	*False pos.*	*False neg.*	*True neg.*
Billing fraud (synthetic data)	0.89	0.98	365	97	3	11	254
Billing fraud (authentic data)	1.0	0.93	61	3	4	0	54
Break-in fraud (synthetic data)	0.92	1.0	89	26	0	2	61
Break-in fraud (authentic data)	1.0	0.86	87	4	11	0	72

classification systems is to be high in both specificity and sensitivity. The *Number (#) of periods* defines the total time frame during which detection was performed. In our tests, a single time quanta was 1440 minutes. The periods specified for each test are harmonized with the graphs illustrated in Figures 13, 14, and 15. For each fraud type, *True positives, False Positives, False negatives* and *True negatives* are shown.

As can be seen in the table, *specificity* is somewhat better for the synthetic test data than for the authentic data. This is expected as the neural network detector was trained using synthetic data and also these data are more regular. However, the *sensitivity* is better for the authentic data, which is more unexpected. This is likely a result from the fact that the synthetic data had a higher fraud rate. In short, the authentic data contained mostly of normal data with only short periods of fraud, while the synthetic data had a significantly higher percentage of fraud, which provided more opportunities for misclassification. Overall, we believe that the differences between synthetic and authentic data are reasonably small and indicate that the training using our synthetic data was successful.

8 Discussion of results and future work

Scalability versus complexity tests. We created hundreds of simulated users acting for a period of seven months. This was sufficient to train and test the fraud detection prototype. We are thus satisfied with the ability of the method and the implementation to scale the number of users and the time period of the synthetic logs.

An interesting scalability issue that should be studied further is the effects of a more complex modeling of the users and the system. Implementing outliers and several user classes should not affect the scalability very much, but the use of a more complex state machine for user behavior would probably affect the performance of the simulation tool.

Diversity of background data. The background data were rather homogeneous. There were, for example, no "outliers" among the users. One reason for this was that only one user profile was used, which meant that all the simulated users behaved according to the same statistical distributions. The obvious solution to this problem is to use several user classes with different profiles, which was our initial intention, but time limits in the project prevented this. For our detection experiments, it seems that the diversity of the background data was good enough for the billing fraud, since the normal behavior in the authentic data did not trigger very many false alarms. However, it posed a bigger problem for the break-in fraud, where normal users and fraudsters showed very similar behavior. It would be interesting to make tests using background data with different degrees of homogeneity.

The user model. The user model may be too simplistic for certain applications. Some behavior was deliberately left out of the model to keep it simple, such as use of the network for other things than downloading movies. Neither were long-term variations in users' activity modeled. It would be interesting to develop a more advanced user model. However, we believe that the user model is of sufficient detail in the VoD application.

The system model. We used static parameters for controlling delays and some of the router statistics in the system model. In a real system, these would vary depending on load etc. Future versions of our simulator will be more dynamic in simulating such dependencies. Neither did our simulator model distinguish "strange" user behaviors, such as malformed network traffic, spoofing etc, which limits the ability to simulate attacks based on bugs in the software and the hardware. This could also be improved in future versions of the simulator.

Fraud cases. Our process of injecting frauds in authentic data can be further improved. We injected frauds in only a few users' behavior and for only short periods of time. This was acceptable for our application but makes it more difficult to verify that the modeling of the users and the system is sufficient for more general use and other target applications. We plan to establish a more lengthy list of fraud scenarios for future experiments.

Suitability for other types of services. Future work will show whether the data generation process works equally well for other types of services and for intrusion detection.

9 Conclusions

We have developed a method for generating large amounts of synthetic log data that preserve statistical properties of a selected set of authentic data used as a seed. We have experimentally shown that the synthetic data generated can be successfully used for training and testing a fraud detection system. Future experiments will verify whether this also holds for more general classes of seed data and for other types of fraud detection systems. We learned several lessons in the process of generating and testing data which will help us to further improve our methodology.

10 Acknowledgments

We would like to thank Telia Research AB (nowadays TeliaSonera) and the people working on the VoD service for their cooperation and help. We would also like to thank the participants of the EURESCOM project ”P1007”.

References

[BSTM+97] Peter Burge, John Shawe-Taylor, Yves Moreau, Bart Preneel, Christof Stoermann, and Chris Cooke. Fraud Detection and Management in Mobile Telecommunications Networks. In *Proceedings of the European Conference on Security and Detection ECOS 97*, pages 91–96, London, April 28-30 1997. ESAT-SISTA TR97-41.

[CFPS99] Philip K. Chan, Wei Fan, Andreas L. Prodromidis, and Salvatore J. Stolfo. Distributed Data Mining in Credit Card Fraud Detection. *IEEE Intelligent Systems*, 14(6), Nov / Dec 1999.

[DDWL98] H. Debar, M. Dacier, A. Wespi, and S. Lampart. An Experimentation Workbench for Intrusion Detection Systems. Technical Report RZ2998, IBM Research Division, Zurich Research Laboratory, Zurich, Switzerland, mar 1998.

[HLF$^+$01] Joshua W. Haines, Richard P. Lippmann, David J. Fried, Eushiuan Tran, Steve Boswell, and Marc A. Zissman. 1999 DARPA Intrusion Detection System Evaluation: Design and Procedures. Technical Report Technical Report 1062, MIT Lincoln Laboratory, February 2001.

[KLJ00] Håkan Kvarnström, Emilie Lundin, and Erland Jonsson. Combining fraud and intrusion detection - meeting new requirements. In *Proceedings of the fifth Nordic Workshop on Secure IT systems (NordSec2000)*, Reykjavik, Iceland, October 2000.

[LKJ02] Emilie Lundin, Håkan Kvarnström, and Erland Jonsson. A synthetic fraud data generation methodology. In *Lecture Notes in Computer Science, ICICS 2002*, Laboratories for Information Technology, Singapore, December 2002. Springer Verlag.

[LX01] Wenke Lee and Dong Xiang. Information-Theoretic Measures for Anomaly Detection. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, May 2001.

[Mos01] Michael C. Moser. *Neural net architectures for temporal sequence processing.* Addison-Wesley Publishing, Redwood City, CA, 2001.

[MT00] Roy A. Maxion and Kymie M.C. Tan. Benchmarking Anomaly-Based Detection Systems. In *International Conference on Dependable Systems and Networks*, New York, New York, June 2000. IEEE Computer Society Press.

[PZC$^+$96] Nicholas J. Puketza, Kui Zhang, Mandy Chung, Biswanath Mukherjee, and Ronald A. Olsson. A methodology for testing intrusion detection systems. *Software Engineering*, 22(10), 1996.

[TM03] Kymie M. C. Tan and Roy A. Maxion. Determining the Operational Limits of an Anomaly-Based Intrusion Detector. *IEEE Journal on Selected Areas in Communication*, 21(1), January 2003.

Paper D

Combining fraud and intrusion detection - meeting new requirements

Reprinted from

Proceedings of the Fifth Nordic Workshop on Secure IT Systems
Reykjavik, Iceland
October 12-13, 2000

Combining fraud and intrusion detection - meeting new requirements

Håkan Kvarnström [*] Emilie Lundin Erland Jonsson

{*emilie, erland.jonsson*}*@ce.chalmers.se*
hakan.kvarnstrom@teliasonera.com
Department of Computer Engineering
Chalmers University of Technology
412 96 Göteborg, Sweden

Abstract

This paper studies the area of fraud detection in the light of existing intrusion detection research. Fraud detection and intrusion detection have traditionally been two almost completely separate research areas. Fraud detection has long been used by such businesses as telecom companies, banks and insurance companies. Intrusion detection has recently become a popular means to protect computer systems and computer based services. Many of the services offered by businesses using fraud detection are now computer based, thus opening new ways of committing fraud not covered by traditional fraud detection systems. Merging fraud detection with intrusion detection may be a solution for protecting new computer based services. An IP based telecom service is used as an example to illustrate these new problems and the use of a suggested fraud model.

Keywords: intrusion detection, fraud detection, modelling, abuse

1 Introduction

Fraud in telecom and datacom services causes a substantial annual loss of revenue for many companies. Fraud can in this context be defined as a deliberate act of obtaining access to services and resources by false pretenses with no intention of paying. For telecom services, such as telephony, signalling abuse (e.g. blueboxing) has been successfully used to place free telephone calls worldwide. Although improved technology such as digital switches has made signalling abuse

[*]The author is also with TeliaSonera AB, SE-123 86 Farsta, Sweden

difficult to exploit, new technological trends such as the introduction of mobile telephony induce new categories of fraud. Subscription fraud, call selling and cloning handsets are examples of the types of fraud that exist in mobile services. Abuse often utilizes a valid customer's service subscription to conceal the crime. A more elaborate discussion of fraud and telecom crime can be found in [Col99a] [Col99b] [Col00].

Telecom companies, insurance companies, banks and other businesses, have been studying fraud and fraud detection for many years and have probably spent more time and money on this than the research community. However, most of their efforts do not reach beyond the limits of the companies and have not been available to the public research community. Still, a number of published papers on the subject are available, most of which focus on detection methods. AI methods are the most common in these studies, including different kinds of neural networks, data mining, case based reasoning etc. There are also statistical methods for user profiling and combinations of methods [GC94]. Often these papers include experiments on fraud data collected in an authentic system, such as mobile phone call records [HTS99], data from an insurance company [GC94] and financial transactions [AME98]. Most papers do not measure the actual gain from using their fraud detection method but in [BGMP99] a cost model have been used to evaluate the classification of tax declarations. Some papers, e.g. [Col99a], [Col99b], [Col00] and [Hoa99], describe the current fraud situation that telecom companies face, and well-known frauds. However, these papers do not discuss any details of the detection process or any organised fraud model.

A great deal of research has been done in the area of intrusion detection (ID). The detection methods studied, e.g. [DDW99], are in many cases almost the same as those used in fraud detection. More work seems to have been done on intrusion models, e.g. [HL93] and [LJ97], than on fraud models. We found very few cross references between research papers in the two areas and none of them discuss effects of combining the areas.

This paper addresses the new problems that are appearing in IP based services (Section 2) and proposes a new fraud model (Section 3) that can be used in the detection process. In Section 4, our fraud model is applied to an IP based gambling service. We also discuss detection techniques and requirements for combining fraud and intrusion detection (Section 5). Some conclusions are given in Section 6.

2 Fraud detection for IP based services

2.1 Towards IP based telecom services

The increasing use of IP, i.e. the Internet Protocol, in telecommunication raises new problems. New telecom services are being rapidly developed today and many of these are heavily based on datacom services (e.g. IP). The market penetration of Internet and the increase in e-commerce applications leads us to believe that new types of fraud are to be expected in the near future. Many of these services, such as online gambling and media-on-demand services, require payment transactions susceptible to fraud. Further, the business model for IP based services differs to a large extent from traditional telecom services. Several different types of actors interact with each other to offer these services. In the traditional case, the main point of interest when looking for fraud is the interaction between the customer and the telecom operator. The new business models involve business relations between several actors (e.g. network, service and content providers), all of which must be taken into consideration. Any interaction where one actor provides a service and another actor pays for it, may be a source of fraud.

2.2 Fraud detection - traditional vs IP based

Traditional fraud detection (FD) applications for telecom services are often tailored to telephony applications. In future services, new technical components will be used for which traditional fraud detection techniques will not be directly applicable. It is evident that new methods of fraud detection must be developed for future telecom services. As new services are more computer and IP network based, there is reason to consider not only traditional fraud detection but also more general computer intrusion detection methods. Fraud detection is often very application specific and aims at detecting ongoing frauds or situations that precede frauds, i.e. events that lead to a person or company being able to profit at the expense of the company providing the service. Intrusion detection aims at detecting computer misuse, i.e. any unauthorized use of a computer system. This includes attacks against service availability and sabotage that are equally as important to detect. Computer intrusions are likely to be a part of the fraud model for new services.

There are some general differences between traditional telecom services and new IP based services. One positive thing about the differences is that some traditional frauds will die out. This can occur because the vulnerabilities exploited in the fraud do not exist in the new services or because the billing model has changed in a way that makes the fraud either impossible or unprofitable. Unfortunately, there is reason to believe that a larger number of new frauds will crop up than will disappear. Many of the old frauds are still applicable and as the Internet gains users' trust and acceptance as a market place, the value and volume of payment transactions will increase. Below are some general problems we expect to find in the new services:

- Values involved in the services are probably greater, which makes the motive for fraud even more obvious. This also means that the actors involved can lose more money.
- More actors are involved, which makes the services more complex. It is always harder to perform a fraud investigation when several actors must co-

operate and data must be collected from several different companies. There are also more candidates for performing frauds.

- The services are more dynamic and probably have a shorter lifetime. This means that fraud management systems must be more flexible and have the capability to be efficiently implemented in a short period of time.

- New technical components will be used, and new weaknesses will thus be found. Many new components will probably be more insecure than old ones. For example, the IP protocol has many weaknesses that are not found in the network protocol used for telecom services today.

- No standardized billing information exists in the new services. The main source of information for fraud detection has traditionally been the call detail record (CDR), which is not applicable to the new services in its current form. Either a new type of standardized billing record must be developed or fraud detection must be adapted to different data for each application.

In conclusion, it can be stated that the new IP based services are more complex and that the fraud management for those services is likely to be more problematic.

2.3 Actors

The expected actors in IP based services are described below. Other actors may appear for other types of services and some will probably disappear, but the list is still applicable for a large group of different services.

Customer. The customer is the end user of the service, i.e. the receiver of service content. In the traditional fraud management system, the customer is the main subject of interest in the search for fraud.

Content providers. The content provider is the actor that delivers the content that finally is delivered to the customers. For example, this may include video films, music tracks and other media suitable for digital distribution such as books, magazines and advertisements.

Service providers. The service provider bears the main responsibility for the service. Service providers come in many different shapes and are difficult to define. For example, a company providing a WAP / SMS service or a web hostel would qualify as a service provider. In some cases the term sub-service providers is used to denote a service provider that offers a specific service. Examples of sub-service providers are:

Application service providers (ASP). A service provider offering (remote) access to certain applications such as word processing, business systems (e.g. ERP) or other applications normally installed on the user's local computer. One of the key advantages of using an ASP is that software upgrades and maintenance are handled by the service provider, which gives large scale advantages and thus brings down the cost per user.

Internet Service Providers (ISP). Internet Service Providers offer access to the Internet by some form of access network. A dial- up service is one commonly used form of access.

Network Providers. The network provider operates the underlying network (backbone) that connects different actors. Owing to the heavy investment cost of establishing a backbone, typical network providers are large telecom companies.

Access network providers. The access network is the network to which the customer is physically connected. For simple modem dial-up services, the PSTN network is used for access. For broadband services, cable-TV(CATV), xDSL or fiber connections may be used for access. The access network is usually connected to the network providers and one or more service providers.

E-payment provider (EPP). E-payment providers serve as mediators in business transactions. An e-payment provider often acts as a trusted third party known to both parties in the business transaction. The buyer knows that he will receive his products once the payment is complete at the same time that the seller knows that he will receive payment as soon as the product is delivered to the customer. Often, an e-payment provider handles transactions of relatively small value (micro payments).

Non-customers. The non-customer is a person or company not registered as a user of the service. He is an important actor in the fraud model. The non-customer must be taken into account in all business relations between the actors.

Each business relation in which money flows from one actor to another is a potential point of fraud. Figure 1 shows an example of how money and services may flow between the actors. All of these relations do not exist in all types of services. A telecom operator may take on several roles, e.g. as both network operator and service provider. This means that for specific services in which this is the case, it is not necessary to consider all relations between possible actors.

2.4 Fraud indicators

Fraud indicators are facts about service usage that may indicate fraudulent behavior. Some indicators that can be used for telephony fraud detection are long duration calls, high call volume and hot destinations. Fraud indicators are often not sufficient by themselves but should be used in combination with other indicators, time windows and threshold variation by customer group and / or time of day.

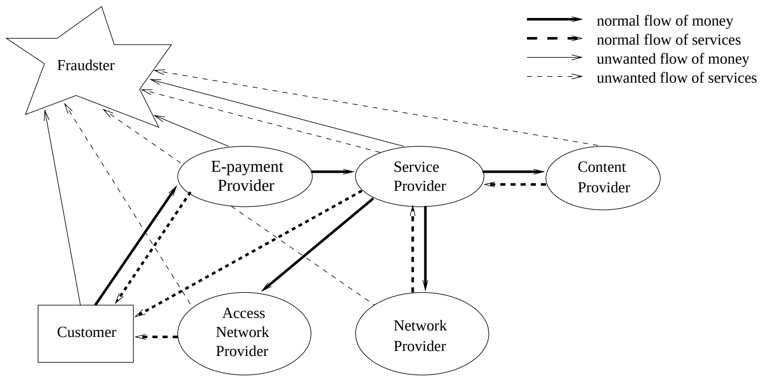

Figure 1: Relations between actors.

These indicators are often found after the service has been used for a long time and it takes a great deal of expertise or statistical material to tune a ”thresholds” for the indicators. Most of the known fraud indicators for telephony services are not directly applicable in the new IP based services. A reason for this is that they are based in most cases on information found in the CDR, which does not exist in the new services. Many of these indicators are also very application specific.

Therefore, there is a need for development of new fraud indicators. Preferably, these indicators should be more general and applicable for a larger group of services. A requirement to make this possible is to standardize the format and contents of collected data.

3 A fraud model

It is obvious that it is necessary to have a clear definition of what constitutes a fraud in order to be able to successfully design a detection system. As mentioned above, while much work has been done to categorize and model intrusions, this is not the case for frauds. Some papers, e.g. [Col99a] and [Hoa99], present frauds but not in a structured way. The purpose of these papers seems mainly to be to present existing, application specific frauds for telecommunications networks and mobile telephone networks. Their usability in designing fraud detection systems for new types of services is limited. This section suggests a model and way to categorize frauds.

3.1 The model

The objective of our fraud model is that it should be helpful in the process of designing fraud detection systems. The main idea of the model is to define groups of fraud characteristics important for fraud detection. The fraud can then be described by its characteristics, which determines what kind of ”detection rules” can be applied. The higher levels of the model deal with general characteristics and the lower levels deal mostly with more application specific characteristics found only in a few services.

A fraud detection system can then be built in modules, where ”detection rules” for frauds with certain general characteristics can be included in a standard module and modules with application specific ”rules” can be added as needed. A high level ”rule” can, for example, check whether the amount of money coming in to the service provider corresponds to the amount of services provided. A low level application specific module can contain rules applying to services using a specific billing model or technical component, e.g. rules for pay-per-view services or IP multicast services.

The fraud model is hierarchical, with more general parameters in the higher levels and more application specific parameters in the lower levels. The levels are presented in Figure 2. The highest level in the model is the general fraud goal, which applies to almost every commercial service. The second level deals with the actor involved in the fraud. The third level deals with the service specific risks that makes the service a target for fraudsters. Parameters on this level concern the content of the service, what billing models are used and vulnerable components. The last level deals with the technique used to accomplish the fraud. Here we must look at the ”implementation” of the components used in the service, both physical and logical. Fraud indicators can be defined on the basis of a selection of fraud characteristics. Some indicators may include knowledge of characteristics on just one level of the model, but knowledge on several levels is often necessary. For example, the fraud technique used to commit a specific fraud may be to attack the network protocols. Then, a fraud indicator based on knowledge of only the fraud technique may be to look for deviations from normal behavior in the network traffic. A fraud indicator using knowledge from several levels may be that the number of bytes of media delivered to the customer does not correspond to the service provider’s billing log. Fraud indicators using characteristics in the higher levels of the model may be added without great knowledge of the target service, but indicators using characteristics in the more application specific layers require detailed knowledge of the service.

Fraud detection may be used by any of the actors involved in a service. The service provider is presumably the actor most interested in using fraud detection because he is involved in several sensitive business relations.

3.2 Fraud goal

The following is the dictionary definition of fraud: ”Fraud is the intentional deception or misrepresentation that an individual knows to be false or does not believe to be true and makes, knowing that the deception could result in some unauthorized benefit to himself/herself or some other person”. In telecom services the idea of a fraud is often that a customer wants to use a service without paying. However, there are also other types of fraud we want to include in our fraud model. Especially in the case of IP based services there are events we wish

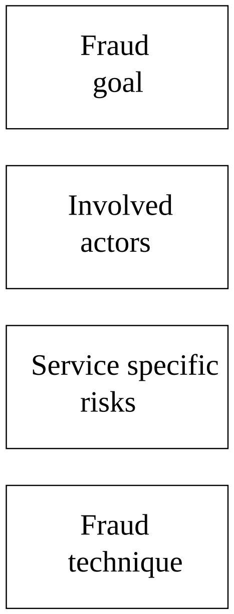

Figure 2: Fraud model

to detect that are not classical frauds but merely events of misuse that would be expected to happen in a computerized system. This includes sabotage and unauthorized use of resources. We assume that the following four classes of general fraud goals will cover what we wish to detect in a future fraud detection system.

1. Evasion of payment for service.
2. Service Level Agreement (SLA) violations. This means that an actor does not deliver contents of the agreed quality.
3. Improper use of service content, resources or procedures. This includes e.g. use of services in a unintended way and unauthorized use of resources, such as networks.
4. Sabotage of service.

A fraud detection system designer should not forget any of these classes when he considers what frauds can be expected for a specific service. The number of fraud indicators that can be defined by using only knowledge of the characteristics found on this level is limited, since not even the actors involved are known here. Something more than the fraud goal must probably be known about the fraud to make it possible to detect.

3.3 Involved actors

The second highest level in the model defines the actors involved in the fraud. The actor that commits the fraud is especially important, but it is also interesting to define the actors that may be affected by the fraud.

By studying the business relations in the service and the general fraud types, we can make reasonable predictions about which actors are likely to commit fraud and we can also analyze which actors will be affected by different frauds. There are six different actors in the gambling service described in Section 4.1. All these actors may have a motive for committing fraud. From the knowledge of fraud type and actors involved, it may be possible to design very general, and therefore reusable, fraud detection rules.

3.4 Service specific risks

The third level is the service specific risks. What is examined on this level is what it is that makes it interesting to commit fraud in this service. As mentioned before, this depends on the specification of the service, e.g. what billing models are used, the content of the service and the type of components used.

A specification of the service is required to define fraud characteristics on this level, but not necessarily an implementation. During the design phase of a service, we can predict frauds by studying the specification of the service and make comparisons with other services with similar contents, billing models or components. If the study shows that high risk frauds that are difficult to detect may appear, a revision of the service specification may be appropriate.

3.5 Fraud technique

The lowest level in the model is the technique used for committing fraud. The technique depends on what components are used in the service. Fraud techniques can be divided into two sub-categories technical frauds and social engineering frauds. A technical fraud involves attacking a technical component, while a social engineering fraud attacks service procedures controlled by employees. The service procedures can be viewed as logical components. Of course, a combined fraud attacking both types of components is possible.

Some technical components expected to be used in most services are:

- Authentication mechanism
- Customer database
- Application server
- Network and network protocols
- Billing data generation and storage
- Service content storage

- Examples of procedures controlled by employees:
- Customer registration procedure
- Customer support

To predict actual fraud techniques, we must have detailed knowledge of the service components and how they are implemented. Fraud indicators on this level are connected to specific service components and can be reused only for other services using the same components. If frauds are to be detected on this level, intrusion detection may be useful because many of the technical components used in an IP based service are general computer system components.

4 Application of the fraud model

This section describes an IP based service and some possible frauds that may occur in this service. The frauds are then studied using our fraud model.

4.1 Gambling service

To give an example of a possible new IP based service, we here describe an online gambling service.

4.1.1 Service description

Online gambling such as Lotto is sold through a network of retail stores that sells gambling services. Figure 3 shows a fictitious gambling service that is provided to customers. Several types of actors are involved, such as a content, network, Internet, and service providers. Content providers may offer a variety of games such as sports betting (e.g. horse races), lottery (e.g. lotto) and gambling (e.g. cards, roulette). In this fictitious example the content provider (CP) specializes in various card games that are offered to the public. The CP allows customers to place bets by utilizing a network of service providers such as online retails stores, Internet gambling sites and affiliates that resell the gambling service (possibly branded under a different name). In addition, customer credit checks and payment transactions are handled by a third party specializing in online payment. Depending on the type of service provider, the payment scheme may vary, ranging from micro payments to credit cards and prepaid accounts. The advantages of using an e-payment provider (EPP) is that customers do not have to release their credit card information or establish accounts at a large number of service providers. Once a customer is registered to an e-payment provider, he can access all services offered by service providers using that EPP. At the same time, the service providers do not have to establish processes for customer billing and verification of customers credit status.

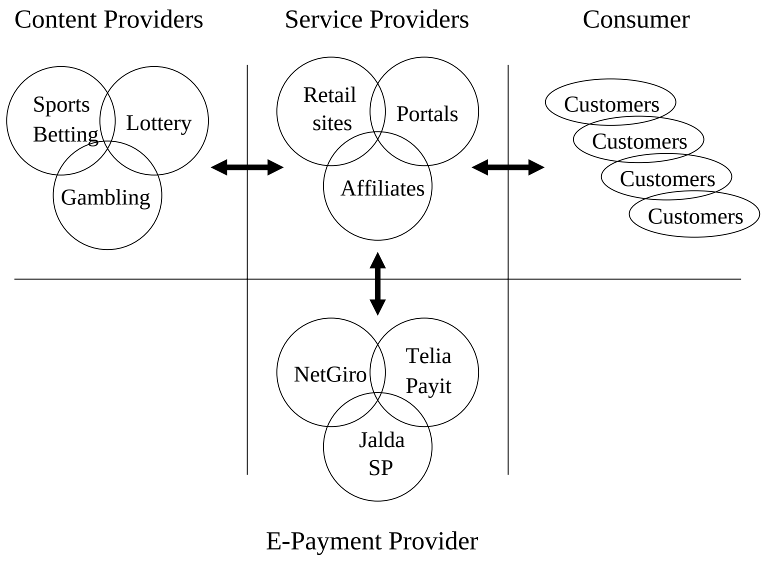

Figure 3: Gambling service

Customer transactions. A typical service transaction involves a series of steps: (1) The customer contacts a service provider that offers a card game; (2) The user identifies himself and is authenticated using some authentication mechanism (e.g. smart card, one-time password etc.); (3) The service provider verifies that the user (still) has a good credit standing by contacting the payment provider; (4) The customer is then allowed to enter a card game where bets are placed. All losses and wins are debited from / credited to the user's account.

Service provisioning transactions. The transactions between the content provider, service provider and payment provider are handled by a clearinghouse function. 50% of the revenue / loss goes to the content provider, 40% goes to the service provider and 10% goes to the payment provider. The risk is equally divided between the parties. The clearinghouse function is managed by the service provider. At regular intervals (daily), payment transactions are made to clear withstanding balances between the different actors.

4.1.2 Fraud scenarios

The possible frauds in this model are many. Here we present a few known frauds that are applicable and may occur in the gambling service. Starting with the customer related fraud cases, the possible frauds include:

Subscription fraud. A user registers himself under false identity and manages to access the gambling service. If the customer loses money, he has no intention of paying the bill. If he wins money, however, the positive account balance can be used to buy services from other services providers associated with the same payment provider.

Identity theft. A stolen user identity and authentication token may be used to gamble on the account of another customer. This could be caused by bad or broken authentication mechanisms or customer negligance. In the intrusion detection area, this type of attack is often called *masquerading*.

The fraud cases for the *service provider* include:

Withholding revenue. The service provider may sell services for which he is withholding revenue. This could be the result of defects in the gambling software, preventing the content provider from verifying each transaction. A loss may be reported as usual, while a win is withheld from the content provider. The fact that the service provider acts as a clearinghouse increases the risk for fraud.

The fraud cases for the *payment provider* include:

Overcharging customer. The payment provider may claim payment for services not consumed by the customer. For transactions of small values, the customer may not notice the charges at first glance (salami attack).

The *content provider* fraud types include:

Unfair gambling odds. The content provider may manipulate the gambling application such that the odds of winning decrease. This is not only unfair to end customers but also to the service and payment providers that put their reputation on stake by offering the gambling service to their customers.

4.2 Fraud study

Five possible fraud scenarios were suggested for the gambling service. Here we study them using the proposed fraud model. Table 1 shows a classification of frauds based on the characteristics of the first two levels of the fraud model. The rest of the characteristics of each fraud are described below the table.

4.2.1 Subscription fraud

Fraud goal: Evasion of payment.

Actors involved: The customer commits fraud against the service provider and / or e-payment provider.

Table 1: Fraud classification

Fraud goal	Actor	Fraud
Evation of payment	Customer	Subscription fraud
	SP	
	CP	
	EPP	
	ANP	
	NP	
	Non-customer	Identity theft
SLA violations	Customer	
	SP	Withholding revenue
	CP	Unfair gambling odds
	EPP	Overcharging customer
	ANP	
	NP	
	Non-customer	
Improper user	Customer	
	SP	
	CP	
	EPP	
	ANP	
	NP	
	Non-customer	
Sabotage	Customer	
	SP	
	CP	
	EPP	
	ANP	
	NP	
	Non-customer	

Service specific risks: It is desirable to gamble with no risk of losing. Service is not prepaid. It is difficult to identify a customer over a computer network.

Fraud technique: Social engineering fraud. Customer registration process is attacked.

4.2.2 Identity theft

Fraud goal: Evasion of payment.

Actors involved: The non-customer commits fraud against the customer and indirectly also against the service provider and / or the e-payment provider.

Service specific risks: It is desirable to gamble using someone else's money. It is possible to use another persons identity and authentication token.

Fraud technique: May be done either by social engineering or technical methods. Social engineering techniques include e.g. physically stealing an authentication token and ”shoulder surfing”. Technical methods include e.g. eavesdropping on network traffic to pick up authentication information.

4.2.3 Withholding revenue

Fraud goal: SLA violations.

Actors involved: The service provider commits fraud against the customer and indirectly also the content provider. What looks like a fraud where the service provider is withholding revenue may also be caused by an external fraudster who is in control of service components.

Service specific risks: The service provider is in a trusted position, as he mediates information about winnings and losses from the content provider to the customer. The customer may not be able to verify the transactions if he is not in close contact with the content provider. The content provider's software may have inadequate or defective monitoring functions.

Fraud technique: This may be a technical fraud if the service provider exploits a defect in the gambling software which prevents the content provider from verifying each transaction. It may be a non-technical fraud if the service provider exploits the content provider's poor fraud awareness and service management.

4.2.4 Over-charging customer

Fraud goal: SLA violations.

Actors involved: The e-payment provider (or an external fraudster) commits fraud against the customer and indirectly against the service provider.

Service specific risks: The payment provider is in a trusted position and can exploit trust. It can be difficult for the customer to verify the correctness of the transactions if he is not in close contact with the service provider or the customer may not be monitoring the transactions at all, which makes it a low risk fraud.

Fraud technique: Can be a non-technical fraud if the payment provider manually generates false information for the service provider. If the payment provider modifies billing software to generate a false billing log for the service provider, it may be considered a technical fraud.

4.2.5 Unfair gambling odds

Fraud goal: SLA violations.

Actors involved: The content provider (or an external fraudster) commits fraud against the customer and indirectly against the service provider.

Service specific risks: It is difficult for the customer and service provider to verify the quality of the service offered when the service content consists of remote use of software. The content provider can therefore ”save” a reasonable amount of money at a low risk.

Fraud technique: Modify gambling software to generate fewer winners, lower winnings and/or unfair distribution of winnings. Log files with statistics may also be destroyed or modified to cover up the fraud.

4.3 Discussion

The fraud model may act as support when a new service is designed. By defining the actors, billing model, content and components, we can predict what known frauds are applicable in the new service. A complete database of known frauds with descriptions following the model would be of great help in this task. If it is found that some known problematic frauds are expected, a redesign of the service can be considered at an early stage to decrease the fraud risk.

The model may also be used to predict new frauds. By studying the combinations of fraud goals and actors in Table 1, we can consider for each combination whether the service encourages a fraud of this type and define the possible service risks. If we instead study e.g. combinations of actors and service risks, we can predict possible or probable fraud methods.

The goal of the work of defining possible frauds for a service is of course to develop ”detection rules”, or fraud indicators that can be used in the detection process. For example, a possible combination of fraud indicators to detect a submission fraud may be to look for new subscribers with low credit and high service usage.

Fraud indicators are important because they are needed regardless of the detection methods used. We have not tried to develop fraud indicators for the gambling service. To be able to derive usable fraud indicators from fraud characteristics, we must have a methodology. Hopefully, our fraud model will be of use in this area as well. This is an interesting research area that we intend to include in future work.

5 Detection techniques

Given the fraud model in the previous section, we will here discuss various techniques that can be applied in a new fraud management system and in particular address techniques already known from the area of intrusion detection. We will also investigate the possible benefit of combined ID-FD approaches.

5.1 Telecommunication Fraud Management Systems

5.1.1 Traditional FMS

A Telecommunication Fraud Management System (FMS) is an automated tool for detecting and managing frauds in telecommunication services. The FMS usually has an advanced operator interface and includes tools for manual fraud investigation. Most commercial Fraud Management Systems (FMS) use Call Detail Records (CDR) as input. CDRs are the basis for customer billing. These records can be used for creating customer profiles and for different kinds of threshold detection etc. It is also possible to use other sources of input data, such as SS7 signalling data [ss7].

Typically, call detail records (CDRs) are collected that contain information about the subscriber and the duration and destination etc. of a telephone call. A combination of thresholds, pattern recognition and statistical subscriber profiling can be used to find indications of fraud in the CDRs. Indications of fraud may also be found using very simple rules such as a list of hot numbers that are known to have been common in previous fraud investigations. Other rules trigger alarms when a certain parameter deviates substantially from a given subscriber profile (e.g. the average duration of calls, the total number of calls during a certain period of time etc.)

5.1.2 Fraud detection

Fraud detection performed in services such as the one described in section 4.1 may have much in common with general computer system intrusion detection but there is a difference in the meaning of the terms. Fraud detection means detecting people misusing a service at the expense of some other party (e.g. a service provider). A service provider may lose money directly or indirectly because of fraud. Direct losses occur when resources are consumed for which the service provider does not receive payment. Indirect loss can occur if a user succeeds in damaging the reputation or market value of the service by for example denial-of-service attacks or by exploiting a service at the expense of another customer. If the service provider is not willing to take the cost, the company will lose goodwill and, eventually, customers.

Fraud detection includes social engineering scenarios but possibly excludes other interesting events that could be an indication of future attacks or fraud. Snooping around in the system may not be considered a fraud even though the person has violated the security policy.

5.1.3 FMS for IP based services

It is clear that the fraud management systems available today can not easily be applied to the new IP based services, mainly because no CDRs are generated by the new services. A FMS must thus be adapted to new types of input in order to be useful and/or some new type of CDR must be defined. What this new input data will look like is not yet clear, but ongoing initiatives are addressing the problem [ndm].

Billing data are often the main source of information in traditional fraud management systems. These will probably be the most useful data source in the new services as well. The main fraud detection system should therefore primarily detect potential frauds in the billing data. Much of the traditional fraud detection techniques can probably be used here. If the format of the billing data is standardized, a detection component can be general for all kinds of different services. However, billing data is not sufficient to make a complete detection system. One idea is to use separate detection components to monitor important service components that are not fully covered by the main detection system. Other forms of input data may be used here, e.g. network traffic and operating system log files. Furthermore, service components that are known to be vulnerable, should be subject to extra monitoring.

Since the new IP based services are predicted to be more dynamic, the detection system must also be dynamic. It should be possible to remove or add detection components to the system without a great deal of trouble. Correlation functions should also be easy to add for new components.

5.2 Intrusion Detection Systems

Intrusion detection can be defined as a technique to detect events in a system (or a domain) that violates a security policy. The security policy can be explicit or implicit. The term intrusion detection normally excludes detection of frauds committed without misusing some technical component of the system. Purely social engineering attacks, such as giving a false identity at registration for a user account, is difficult to detect by technical means, and is thus not likely to be considered by an intrusion detection system.

A great number of commercial Intrusion Detection Systems (IDS) have appeared in the last few years [Kva99]. Although research has been going on in this area for at least two decades, only very simple techniques have been adapted in commercial systems. In research, however, all kinds of techniques are discussed. Most of the techniques used for fraud detection have also been used for intrusion detection. Most commercial intrusion detection systems use network packets and host-based log files, often from the operating system, as input. In some cases, application log files such as firewall logs or logs from a web server can be input for analysis.

The most common detection strategy in commercial IDSs is to create rules for known intrusive behavior. This strategy is called misuse detection and is often implemented as rule-based expert systems or as simple pattern matching. Events that these systems detect are e.g. network denial of service attacks like syn flooding and buffer overflow attacks that are used to gain increased privileges. The other detection strategy, that is used in only a few commercial systems, is anomaly detection. Anomaly detection means creating behavior profiles and then detect deviations from these profiles. It can be implemented using e.g. statistical methods or neural networks. This strategy can be used for detecting masqueraders, password guessing, etc. In [HB95], a discussion of detection strategies and implementation methods can be found. Events are in most cases not correlated to any higher degree in intrusion detection systems, which means that some attacks are missed and some causes floods of alarms.

5.3 Fraud detection vs intrusion detection - differences and similarities

A detection system in a complex datacom service may include both fraud and intrusion detection to cover all events of interest. There are some fundamental differences in the design criteria between systems for intrusion and fraud detection. In general, a fraud detection system is designed to detect fraud above a certain threshold. That is, the detection threshold can be adjusted such that insignificant cases of fraud do not generate floods of alarms. An intrusion detection system is designed to detect intrusions and intrusion attempts. There is no such thing as an insignificant intrusion, which makes it difficult to tune an intrusion detection system not to raise too many false alarms. IDS techniques are not directly applicable to fraud detection. Fraud detection can be seen as an application specific form of intrusion detection. Application specific input and application specific rules for what is considered ”misuse” must be applied. It would be an advantage if it was possible to adapt the IDS in such a way that it could be used for fraud detection. However, most intrusion detection systems of today can not be adapted in a straightforward and effective manner.

A combination of IDS and FMS may be a way of achieving better coverage of the possible frauds and intrusions in future services. One key advantage of utilizing fraud detection techniques is that it enables an organization to put a price tag on losses caused by fraud. This can not easily be done for intrusion detection as it is difficult to predict the economic consequences caused by an intrusion. This could be used within an organization to help motivate security improving investments.

5.3.1 Vulnerability detection

There are also differences in the detection strategies in FMS and IDS. One possible strategy is to have a list of all vulnerabilities in the system and try to detect the use of these vulnerabilities. This strategy is often called misuse detection in the intrusion detection area. Here the focus is on detecting the technique used by the attacker or fraudster. This is the major method used in commercial intrusion detection systems today. The list of known intrusion / fraud techniques often becomes long and tiresome to maintain. It is also easy to leave out important entries and to miss variations of attacks. Currently unknown techniques will also be excluded. To collect and analyze the data necessary to detect the use of all different vulnerabilities is resource demanding.

5.3.2 Consequence detection

A detection method entirely based on these vulnerability lists is probably not feasible in a complex system such as the one we are dealing with. To reduce complexity, consequence detection may be used instead. This means looking at the ”attack goal” rather than the method the attacker has used. In the fraud case, this often means detecting that less money is flowing in to the service than expected, or detecting a more extensive usage of the service than expected.

The attack method is of course interesting, but it can be investigated in more detail after a fraud or intrusion has been detected. The detection strategy in FMS often resembles consequence detection. The consequence of a fraud may for example be that a user has access to a service he is not paying for or that a denial-of-service attack is in progress. This method may require fewer auditing points and less resources.

Among the disadvantages of consequence detection are that attempted attacks can not be detected and that the detection is not always timely. It is interesting to study attempted attacks because they give a hint of how large the threat is and what methods are most commonly used. The timeliness of the detection may be important because some attacks can cause damage before they are detected. The timeliness problems of this method are caused by the fact that the intrusion is not detected until the attacker performs an act considered to be the ”goal” of the attack. For example, he may have been snooping around and placed Trojans in the system before he completes the attack. He also may have modified or removed log data.

The disadvantages of consequence detection may be reduced or remedied by combining it with detection targeted at known vulnerabilities. Especially, extra effort may be used to monitor components of the system that are considered sensitive.

5.4 Combining fraud and intrusion detection

Intrusion and fraud detection applications serve different purposes even though they share many characteristics and functions. This section highlights issues that need to be addressed when combining fraud and intrusion detection.

To combine FMS and IDS in future fraud detection systems, we must have an idea about how to do this effectively. As we discussed in the previous chapter, the typical detection strategies in the two types of systems can complement each other. However, a detection system consists of more than a detection strategy and these other components also needs to be discussed. Figure 4 shows the components that make up a typical IDS / FMS system.

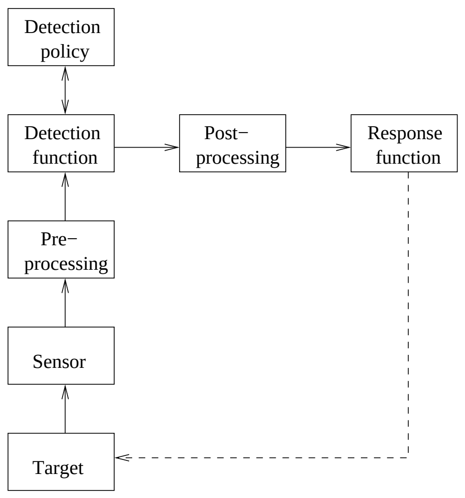

Figure 4: Components of an IDS / FMS system

Targets and sensors. The targets that are monitored are usually some kind of network element or computer system that produces the audit data to be analyzed. The type of analysis determines the features that are of interest, such as IP header data, performance and / or billing data. For traditional telecom fraud detection systems, CDRs are analyzed for detecting fraudulent behavior. However, as future telecom services migrate towards the IP protocol suite, the differences between IDS and FMS data collection will be less noticeable.

Preprocessing. Preprocessing involves filtering and formatting audit data such that it can be applied to different analysis (detection) mechanisms. Our studies of IDS / FMS lead us to believe that the preprocessing phase is similar for both types of systems.

Detection function and policy. Simple rule based detection schemes are successfully being used for both types of systems in current products and prototypes. Pattern matching and threshold detection mechanisms are examples of rule based mechanisms. In addition, Telecom FMS often build and manage customer profiles describing some characteristics of a subscriber / user. For example, a profile may be created that contains the average length of a telephone call over a certain period of time. Similarly, anomaly based IDSs seek to find anomalies over some parameter of interest. Our conclusion is that the detection function is not limited to either IDS or FMS but could be used for both applications.

Postprocessing. The postprocessing of alarms can involve several different actions, such as classification, prioritization and storage of alarms. In FMS, a case building facility is often utilized to cluster events that relate to each other in some respect (case building). This is a feature that may also be useful for IDS. However, few existing IDS products have this functionality. In most cases, if an attack triggers multiple alarms, it is up to the operator to interpret these as a single event (case).

Response. The response function is perhaps the component that differs the most between IDS and FMS. Once an intrusion has been detected by an IDS, automatic response actions can be performed that terminate the attack or try to limit the damage caused by an attack. Automatic response functions must be used with caution, however. It is always a balance between the value of the assets that the IDS tries to protect and the availability of the service that the asset realise. FMS typically report indications of fraud that can not necessarily be stopped by automatic response actions. Indications of fraud will in most cases require an extensive investigation involving several non-technical parties such as the company financial department, law enforcement agencies etc.

Architectural issues. To make the detection system effective, alarms from different detection components should be managed centrally by the combined FD/ID system.

Other functions can also be managed centrally, such as storage of log data. The components must therefore be compatible or use a common information exchange format. Correlation of data from different data sources and different detection components is also desirable in a system like this and it will require far-reaching standardization in order to be effective.

More issues are likely to appear in efforts to combine fraud and intrusion detection in practice. It seems as though the combination of knowledge from the two areas can result in a much better system, even though many details of how to accomplish it still remain.

6 Conclusions

Although the areas of fraud detection and intrusion detection have many common denominators, we found that there is very little crossover research. It becomes clear that integration of the two areas would be beneficial in looking at the new problems that will appear in new telecom services, especially in IP based services. We also found that there is a lack of applicable fraud models, and we believe that our fraud model may be of use in predicting frauds for new services and for the design of reusable modules for fraud detection systems. However, there is still some work to be done on the model, and we have plan to test it in the design of a fraud detection system for an IP based service.

References

[AME98] D.W. Abbott, I.P. Matkovsky, and J.F. Elder. An evaluation of high-end data mining tools for fraud detection. In *1998 IEEE International Conference on Systems, Man, and Cybernetics*, volume 3, pages 2836–2841. IEEE, October 11-14 1998. ISBN: 0-7893-4778-1.

[BGMP99] F. Bonchi, F. Giannotti, G. Mainetto, and D. Pedreschi. A classification-based methodology for planning audit strategies in fraud detection. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 175–184, San Diego, CA USA, August 15-18 1999.

[Col99a] Michael Collins. Telecommunications Crime - Part 1. *Computers & Security*, 18:577–586, 1999.

[Col99b] Michael Collins. Telecommunications Crime - Part 2. *Computers & Security*, 18:683–692, 1999.

[Col00] Michael Collins. Telecommunications Crime - Part 3. *Computers & Security*, 19:141–148, 2000.

[DDW99] H. Debar, M. Dacier, and A. Wespi. Towards a Taxonomy of Intrusion Detection Systems. *Computer Networks*, 31(8):805–822, April 1999.

[GC94] B. Garner and F. Chen. Hypothesis generation paradigm for fraud detection. In *Proceedings of TENCON '94, IEEE Region 10's Ninth Annual International Conference.*, 1994. Theme: Frontiers of Computer Technology.

[HB95] Lawrence R. Halme and Kenneth R. Bauer. AINT misbehaving - a taxanomy of anti-intrusion techniques. *Proceedings of the 18th National Information Systems Security Conference*, pages 163–172, October 1995.

[HL93] P. Helman and G. Liepins. Statistical Foundations of Audit Trail Analysis for the Detection of Computer Misuse. In *IEEE Transactions on Software Engineering*, volume 19(9), pages 886–901, September 1993. ISSN: 0098-5589.

[Hoa99] Peter Hoath. What's New in Telecoms Fraud? *Computer Fraud & Security*, vol 16(issue 2):13–19, February 1999. Elsevier Science Ltd.

[HTS99] Jaakko Hollmén, Volker Tresp, and Olli Simula. A self-organizing map for clustering probabilistic models. In *Ninth International Conference on Artificial Neural Networks (ICANN 99)*, volume 2, pages 946–951. IEEE, September 7-10 1999. Conference Publication No. 470.

[Kva99] Håkan Kvarnström. A survey of commercial tools for intrusion detection. Technical Report 99-8, Department of Computer Engineering, Chalmers University of Technology, SE-412 96 Göteborg, Sweden, 1999.

[LJ97] Ulf Lindqvist and Erland Jonsson. How to systematically classify computer security intrusions. In *Proceedings of the 1997 Symposium on Security and Privacy*, pages 154–163, Oakland, California, May 4-7 1997.

[ndm] Network Data Management - Usage (NDM-U) for IP-based services. www.ipdr.org. IPDR Working group.

[ss7] ISO SS7 Standard. www.iso.ch. International Organization for Standardization.

Part II: Protecting intrusion and fraud detection systems

Paper E

Security Implications of Distributed Intrusion Detection Architectures

Reprinted from

Proceedings of Fourth Nordic Workshop on Secure IT Systems (NordSec 1999)
Stockholm, Sweden
November 1-2, 1999

Security Implications of Distributed Intrusion Detection Architectures

Håkan Kvarnström[*] Hans Hedbom[†] Erland Jonsson

{*hkv,hansh,erland.jonsson*}*@ce.chalmers.se*
hakan.kvarnstrom@teliasonera.com
Department of Computer Engineering
Chalmers University of Technology
412 96 Göteborg, Sweden

Abstract

This paper addresses the problem of protecting an Intrusion Detection System (IDS) against intrusions as a function of the degree of distribution, i.e. it deals with the problem of the security of the IDS itself. First, we define three basic Intrusion Detection Architectures (IDAs), denoted *strictly centralized* and *distributed*, and a suggested *fully distributed architecture*. It is clear that evolution is proceeding towards more and more distributed systems. One problem is that a higher degree of distribution makes systems increasingly exposed to tampering and leakage of information. To cope with this problem we suggest a number of fundamental security requirements and discuss to what extent they may be applied to the architectures and to what benefit. Finally, a few existing IDSs are analyzed in view of these requirements, showing that there are no systems that completely solve the problem of self-protection.

Keywords: intrusion detection architectures, security policy, information flow, distributed systems, security requirements

1 Introduction

Intrusions in computer systems pose an increasing problem that may lead to loss or theft of information or computer resources. Distributed system architectures connecting a large number of computers raise questions as to how to better protect the integrity, confidentiality and availability of the systems. Intrusion detection (ID) is an emerging technology for detecting unauthorized users and sus-

[*] Håkan Kvarnström is also with TeliaSonera AB, SE-123 86 Farsta, Sweden [†] Hans Hedbom is also with the Department of Computer Science, Karlstad University, SE-651 88 Karlstad, Sweden

picious behavior in computer systems. During the last decade, a large number of different intrusion detection systems (IDS) has been presented in the research community. The earlier of these systems had a centralized architecture but, in later years, many systems presented have a partially distributed architecture, and the trend is towards an even greater degree of distribution. Much previous research in the field has been biased towards the mechanism of detecting intrusions and to some extent the architecture and topology of intrusion detection systems. Few, however, have discussed the security implications and requirements that the intrusion detection architecture (IDA) imposes on the IDS itself and its ability to meet existing confidentiality and integrity requirements in distributed systems. Most current architectures use a series of components to provide collection of audit data, analysis of audit data, storage of audit data and response to security events. In a distributed environment, the analysis of audit data requires input from hosts distributed throughout the network. Depending on the detection policy of the domain, collecting audit data from these distributed hosts may violate the confidentiality requirements in practice. In addition, heavily distributed IDS impose new threats on the confidentiality of the detection policy.
To give the reader a glimpse of the types of configurations we envision and to set a mindframe for the following discussion, we offer the following example. Companies A, B and C are cooperating on an important project. As part of this cooperation, communication channels are established for the purpose of exchanging information related to the project. As a step in this interconnection, the companies also want to interconnect their intrusion detection systems. This allows security related events in different domain to be correlated, resulting in a higher probability of detecting suspicious behavior. However, this type of interconnection will raise new security concerns and requirements on the IDS. In a sense, a communication channel between cooperating parties could in itself impose a threat to the respective organizations. A major concern is the requirement not to disclose internal events and detection policies outside the company IDS. Events and policies must be viewed as sensitive and valuable assets in a system. Given the knowledge of the detection policy, a potential intruder would potentially know what to do to penetrate the target systems and how to circumvent the IDS to avoid detection of his actions. The danger of disclosing the configuration of a company ID capability has previously been highlighted in the work of Ptacek and Newsham [PN98].

The ability to resist attacks against itself is an essential property of any IDS. For example, a compromised IDS will probably not report an intrusion no matter how clever the detection mechanisms are. In addition, a compromised IDS may be a source of severe information leakage as described above. This leakage is not limited to information that originates from the target systems (i.e. the systems under surveillance), but may also contain information related to the IDS and its operation.
This paper addresses the security implications and requirements that the IDA puts on the IDS in various distributed environments. We claim that, although

they are more accentuated in what we call a fully distributed architecture, these requirements hold for any type of IDS that consists of interconnected cooperating components. The authors also believe that those requirements have in large part been overlooked in today's systems as a consequence of the bias toward detection mechanisms. For the reader interested in the technical aspects of interconnecting IDS we suggest the work in progress of the CIDF [KBS98][SCTP+98] standardization effort.

2 Terminology

A common nomenclature is needed in order to discuss and compare aspects of different intrusion detection systems. Below we define a number of concepts which will be used in this paper.

Target. A target is defined as a component or information system for which intrusion detection is deployed. A target is usually a network, a network host, an application, a service or any other component monitored by the IDS.

Detection policy. The detection policy is defined as the set of rules that together state what events or occurrences are considered authorized versus unauthorized. The scope, granularity and purpose etc. of the detection policy may vary depending on the target and its application. Examples of detection policies for IDS systems are:

- Rules for accesscontrol to targets
- Misuse signatures of audit data
- Statistical user or system normal behavior

Nodes. Nodes are the entities that together form the intrusion detection system. Each node can have a *passive* or *active* role. Passive nodes (a.k.a. probes) collect audit data which are then further distributed without analysis of the information. In CIDF [SCTP+98] terms, these nodes may consist of event generator, response units and the less complex forms of analyzers. An active node may collect audit data originating directly from the targets or via passive nodes. In addition, the active nodes analyze audit data searching for detection policy violations based upon the audit data and the detection policy. Active nodes may also have response capabilities such as the ability to notify a security officer or to trigger alarms. This means that an active node in the CIDF universe would consist of one or more analyzers of the more complex type, besides the components found in the passive nodes, and could also contain a component capable of producing prescription gidos. The division into passive and active nodes is merely a simplification made by us to emphasize an important property of the node, i.e. whether or not it needs knowledge about the detection policy. A typical distributed intrusion detection system would consist of one or more active nodes and zero or more passive nodes. A set of nodes could logically form any topology or structure. For example, a tree structure of nodes can be used to form a hierarchical distributed IDS.

Domain. A domain is defined as the area that is under the supervision of an active node and its corresponding passive nodes. A domain may be a network, a subnet, part of a subnet, a network host, an application or even a single process. However, in most cases, a domain would be a network or a single network host. It is important to understand that the domain is defined from the ”eye of the beholder”, i.e. from the active node's point of view. This implies that more than one active node can supervise the same area of the system and still have different domains. These domains would then fully or partially overlap. We believe that the domain is a central property of an active node since it constitutes what parts of the detection policy the active node must know. The domain can also give an indication of which part of the system is potentially untrustworthy in the case of a security breach. If a security breach occurs in one of the nodes, a the whole domain must be treated as unreliable (or only partly reliable). Domains containing partly reliable information sources must be identified and isolated from the trusted IDS. Domains could also be a mean to control the flow of information in the system or between systems.

Events. An event can be defined as an occurrence or activity, usually related to a target or a domain. An event can originate from a variety of sources, such as network-layer datagrams, application and security logs or security audit data.

3 Intrusion Detection Architectures

There are currently a number of different architectures for intrusion detection systems. These include both centralized and distributed architectures and combinations thereof. Centralized systems collect raw audit data from some source such as a network interface, a host computer or an application. After analyzing the collected data, a decision is made whether a detection policy violation has occurred.

In a *strictly centralized system*, both the data collection and the decision process are performed by the same entity (Figure 1). In a distributed environment, a distinction must be made between the topology of data collection and the decision process. A common architecture is to have a centralized decision function whereas the data collection function is distributed (Figure 2).

In a *fully distributed system*, both the collecting function and the decision function can be distributed (Figure 3).

As of today, many systems have been presented that have a distributed collection function using remote probes (or agents) for that purpose. However, few of

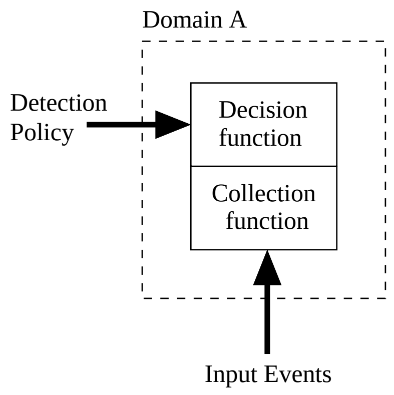

Figure 1: Strictly Centralized IDS

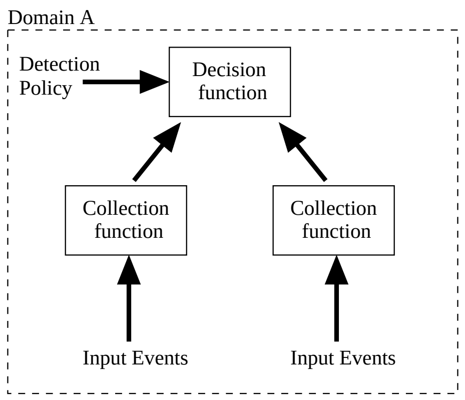

Figure 2: Distributed IDS

these have a distributed decision function. Notable exceptions to this are EMERALD [PN97], JiNao [JGS$^+$97] and CMS [WP96].

3.1 Intra domain versus inter domain detection

In an intrusion detection system consisting of several domains, there is a purpose served by differentiating between *inter* and *intra* domain intrusion detection. Inter domain detection is handled by a single active node whose purpose is to detect intrusions within (or against) a single domain. Inter domain detection is accomplished by interconnected cooperating active nodes in different domains. The purpose of this cooperation is to find situations in which seemingly harmless events from different domains together form a violation of the detection policy.

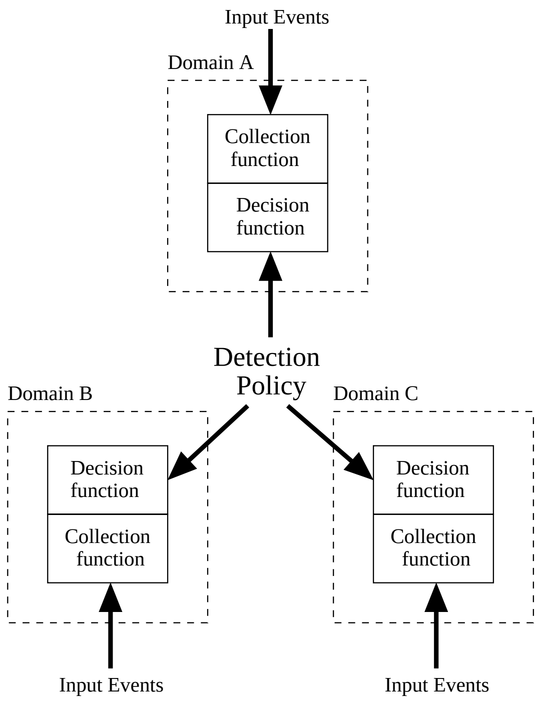

Figure 3: Fully Distributed IDS

3.2 The ”Knowledge” of a distributed IDS

Information flow between components is inevitable in distributed environments. In the case of intrusion detection, information flows between ID components using some form of communication channel. One of the most interesting information flows occurs between centralized managers and distributed agents. The flow between these components depends greatly on whether the agents have a passive or active nature. Since passive agents (or nodes) have little or no logic, they serve as mere collectors of raw events. These data are distributed to one or several managers for further processing.

Distributed knowledge of the detection policy. The detection policy defines a distinction between authorized and unauthorized events and behavior. With knowledge of the detection policy of a domain, a malicious adversary has a higher probability of circumventing existing security measures. Active distributed nodes have the ability to collect and analyze audit logs for the purpose of detecting intrusions. To be able to pursue this task, the node must be aware of the detection policy in practice. The detection policy must therefore be regarded as a form of ”distributed knowledge” known to the entire population of active nodes. Assuming the existence of some centralized manager which defines the detection policy, this information needs to be securely distributed to all active nodes in the system. However, secure communication channels between components is not sufficient to protect the detection policy. It is relatively safe to assume that the more nodes that have knowledge of the detection policy, the higher is the probability that it eventually is disclosed to malicious adversaries. A security breach in one of the nodes may lead to a disclosure of the detection policy in the entire domain or, even worse, the whole cluster of cooperating domains.

Distributed knowledge of security events and audit logs. Distributed systems of passive nodes need a centralized decision function. Raw input events are collected by the passive nodes and sent to a centralized manager for analysis. Depending on the site security policy in practice, the distribution of events may violate the security policy as it may contain sensitive or classified information.

Confidentiality concerns in distributed IDS Depending on where the decision function is located, either audit data or the detection policy must be distributed in some form. Using a centralized decision function, event records must be collected from several domains which may violate confidentiality requirements of audit data. A malicious adversary with access to event records originating from a domain may be able to deduce information about users and their activities as well as classified application-level data. Anonymization of events is one viable solution to this problem. However, anonymization only partly solves the confidentiality problem.

On the other hand, using a distributed decision function, the detection policy must be distributed to several domains, which may violate the confidentiality requirements of the detection policy. From the discussion above, we conclude that the confidentiality of input events and the confidentiality of the detection policy are two *conflicting parameters* in current IDS.

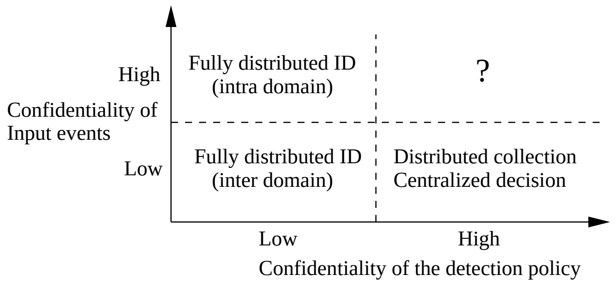
High
Confidentiality of Input events
Low
Fully distributed ID (intra domain)
?
Fully distributed ID (inter domain)
Distributed collection Centralized decision
Low
High
Confidentiality of the detection policy

Figure 4: Confidentiality in distributed IDS

Figure 4 shows how the confidentiality of input events and the detection policy relate to each other. Confidentiality can be regarded in this respect as the degree of exposure a certain architecture imposes on the input events and the detection policy. An *inter domain fully distributed IDS* has the capability to share audit data between domains, whereas an *intra domain fully distributed system* limits the distribution of audit data to nodes within its own domain. The question mark in the upper right quadrant of Figure 4 reflects the lack of an IDA that provides a high degree of secrecy for the detection policy and the input events at the same time.

4 Security Requirements for IDS

The previous section discussed the knowledge needed by different components in an IDS and the flow of information between components that this knowledge produces in different IDAs. We also discussed the security implications of the different information flows. The present section presents a number of security requirements for an IDS based on the flow of information and security implications previously discussed.

4.1 Background and previous work

A majority of the research on security and intrusion detection has addressed the security of the target systems. However, only a few attempts have been made to address the security of the IDS itself. In the work of Debar and Dacier et al. [DDW99], *faulttolerance* is introduced as a property that addresses the IDS' ability to resist attacks. Axelsson [Axe98] also identifies the lack of research in this field and introduces *security* as "the ability of the system to withstand hostile attack against the system itself".

The security that does exist in modern commercial IDS systems is centered primarily around concealment, e.g. unaddressable network cards. This, in our opinion, is security by obscurity. There are a few notable exceptions in this area [JGS+97][Pax98].

An extensive set of requirements for tamper proofing NIDES [Lab95] is identified in a research report by Neumann [Neu95]. He suggests that tamper proofing NIDES can be achieved by fulfilling a series of goals related to the authenticity, integrity and confidentiality of the *analysis system* (NIDES) and its components. The report proposes the protection of audit data and the analysis system rulebase via subsystem encapsulation. In addition, proper authentication and separation of roles play an important role in securing NIDES. To prevent reverse engineering of the rulebase (i.e. detection policy), Neumann proposes the use of encryption. Although encryption prevents external users from reading or modifying the rulebase, it does not prevent the rulebase of a subverted node from being disclosed or modified. A malicious user that has gained control of a node may find a stored encryption key by exploiting the random nature of such keys [SvS98]. Further more, password sniffer attacks can be utilized to obtain encryption keys as they are entered by the user.

4.2 Separation of ”knowledge” and control logic

A simplified intrusion detection system can be seen as a function or (computational) machine taking a set of inputs. Figure 1, presents two types of input: (1) input events and (2) detection policy. The input events are analyzed according to the detection policy; taken together, they deterministically determine the output of the system. It is reasonable to say that these two inputs form the basis for the whole detection process and must be protected from malicious disclosure, deletion, fabrication and modification. Analogous to cryptographic schemes, where the confidentiality of the encryption key and the confidentiality of the plaintext are the only parameters that affect the strength of the encryption scheme, the protection of input events and the detection policy define the strength of the security of the IDS. The code base (i.e. control logic) responsible for the processing of the input information must be integrity protected to prevent tampering. However, the code base is not sensitive to malicious disclosure. On the contrary, the codebase should be open for public scrutiny similar to other security applications and encryption algorithms. This implies that there must be a clear separation between the code base and the protection policy, i.e. the code base is simply an abstract machine executing and enforcing the detection policy.

4.3 Information dominance - Toward a stronger notion of security for IDS

For the purpose of discussion, one can model a IDS as a trusted entity surrounded by untrustworthy (potentially malicious) adversaries. The goal of the IDS is to detect any attempt to violate the boundaries of its domain and the target systems contained. This goal can only be accomplished as long as the IDS succeeds in maintaining its integrity to the extent that it still has an operational advantage. That is, even if some information about an IDS is disclosed to a malicious adversary, it may still be possible to meet operational requirements. To meet these requirements, we introduce a new property described as *information dominance*. The meaning of information dominance for IDS is that information (knowledge) contained within an intrusion detection entity must be kept private to malicious adversaries (confidentiality requirement). In addition, the information must be protected from unauthorized alteration, fabrication and deletion (integrity requirement) as it may lower the operational advantage of the IDS. The IDS must always maintain an information dominance as compared with any external adversary. This allows the IDS to use its information system and capabilities to achieve an operational advantage while denying these capabilities to an intruder. In the field of information warfare, information dominance is an essential property [Arm95].

We propose that the property of information dominance for IDS should include:

Confidentiality of audit data. Audit data generated by entities within a domain may contain sensitive information not to be disclosed outside the members of the domain. Such information may include information about users or their activities as well as application related data possibly containing classified information. In some cases, the mere existence of an event may be confidential as it reveals some form of activity. A principle of minimum knowledge transfer should be followed to avoid disclosure of confidential data. Consequently, raw audit data should never be distributed outside the boundaries of the domain.

Confidentiality of the detection policy. In security services, such as firewalls, the detection policy is distributed to a small number of entities. A distributed security service, like a fully distributed IDS, requires the policy (or parts of the policy) to be known to all domains. Assuming an architecture where the number of domains is large, there is a non negligible probability that the policy is disclosed to a malicious adversary who succeeds in penetrating one or more of the domains. Clearly, the secrecy of the detection policy can not rely upon the integrity of neighboring domains or even upon the integrity of its own domain. The detection policy should be protected against disclosure to malicious adversaries. It should not be possible to deduce the detection policy given the information gained by penetrating a domain or a node contained within a domain.

Integrity of audit data. The audit data are the basis for all analysis in search of intrusions. Hence, an intruder violating the integrity of the audit data may seriously affect the detecting capability. Even the most advanced IDS will fail to meet its operational requirements if the integrity of audit data has been violated. Therefore, the audit data should be protected against unauthorized alteration, deletion and insertion.

Integrity of the detection policy. The detection policy states which activities are considered as intrusions and which are not. Hence, manipulation of the detection policy may cause the IDS to fail to detect an intrusion. The detection policy should thus be protected against unauthorized alteration, deletion and insertion.

5 Research prototype compliance

This chapter compares modern intrusion detection systems in order to judge how well they adhere to the requirements presented in this paper. The systems chosen are all research systems and have been developed recently. Moreover, they are all distributed or claimed by their developers to have the potential to work in a distributed manner. The systems compared with our requirements are EMERALD [PN97], JiNao [JGS$^+$97], GrIDS [CCD$^+$99][SCCC$^+$96] and Bro [Pax98].

5.1 Description of the analyzed systems

5.1.1 EMERALD

EMERALD [PN97] is a real-time intrusion detection system developed by SRI International. SRI has a long history of research in the field of intrusion detection. EMERALD operates by analyzing audit logs collected from system or network resources. EMERALD consists of light-weight surveillance monitors operating stand-alone or in concert. Analysis hierarchies can be created which allow EMERALD to detect more global threats within or in between domains. Due to the flexible nature of the monitors, the system can scale from monitoring a single target up to an enterprisewide deployment having large numbers of cooperating monitors. Monitors can exchange alarm information with each other by utilizing a subscription list. Both signature analysis and statistical profile-based anomaly detection are supported by the monitors. EMERALD's signature analysis subsystem employs an expert system (P-Best) that allows rules to be defined for each target. Each monitor has a resolver that coordinates reporting and response based upon input from the statistical and signature engines. A fundamental design feature of EMERALD is its separation of analysis semantics from the monitor's code base. All analysis semantics are encapsulated into resource objects that define the behavior of the monitor. The resource objects contain information such as event collection methods, analytical module parameters, response methods and policies.

5.1.2 JiNao

JiNao is a system that was primarily developed to detect intrusions or attacks on network infrastructures. It consists of one or more remote management applications and a number of local JiNao systems. The management application is a normal network management station that can control and get information from the local JiNao system. The local JiNao system resides inside a network router and is divided into a number of modules. The most central modules are a detection module, a decision module, a prevention module and a management information base (MIB). The prevention module acts as a firewall to stop obviously erroneous packages from reaching the rest of the system. The detection module performs both anomaly detection and normal pattern matching on the incoming packages. Suspicious packages are sent to the decision module. The decision module decides whether or not an intrusion has occurred on the basis of the information it receives from the detection module and information in the MIB. This module can also change parameters in the detection module and the prevention module, on the basis of the information it has and the decision it makes. It is can also change the information in the MIB on the basis of its decision. The MIB acts as a repository where the management application can change parameters in the local JiNao system and also get information from the local system. The idea is that local intrusion attempts should be detected by the local system, while more widespread attempts should be detected by the remote management application.

5.1.3 GrIDS

GrIDS [CCD+99][SCCC+96] is an IDS developed at UC Davis for detecting coordinated attacks on information infrastructures. It does this by constructing and analyzing graphs based on events in the system. Basically, GriDS consists of three types of components: data sources, graph builders and software managers. The system is built in a hierarchical architecture centered around departments. Every department consists of a graph builder and its data sources and a software manager. The software manager's responsibility is to mange the underlying hierarchy and to start and stop components. The data sources collect data and report events to the graph builder, which constructs and analyses graphs based on these events according to a programmable ruleset. The graph builder also acts as a data source to graph builders higher up in the hierarchy by sending them compressed versions of its own graphs. In this way, only some of the information is spread upwards. GrIDS is primarily built as a proof of concept of a scalable IDS that can handle large amounts of information.

5.1.4 Bro

Bro is not primarily a distributed intrusion detection system. It is a real-time monitoring architecture consisting of stand-alone monitors. Each monitor passively monitors a network link by collecting and analyzing the underlying network packets in transit. Conceptually, Bro is divided into several parts. An ”event engine” reduces network event streams to higher-level events. The high-level events are analyzed using a specialized programming language used to describe the site security policy (detection policy). Considerable effort has been made to provide a clear separation between the detection mechanism and the detection policy. Although the monitors of Bro operate independently of each other, notifications of suspicious activities can be sent to a centralized location using the UNIX syslog function. This allows Bro to be used in distributed environments.

5.2 Compliance to security requirements

Most existing architectures for intrusion detection adhere to the information dominance property by using security mechanisms such as accesscontrol and encryption. Accesscontrol can be used to prevent unauthorized users from accessing the detection policy or the events stored in a node. Events that are collected and further distributed can be protected using encryption. This prevents an intruder from intercepting the information while in transit, although it does not prevent a subverted node from disclosing information about events.

5.2.1 Emerald

EMERALD is an extremely flexible system allowing various architectures to be realized. Although single monitor configurations are possible, the possibility to design large distributed architectures is one of EMERALD's key advantages. In distributed architectures, event streams are distributed among monitors using a subscriber mechanism. Thus, knowledge about events is distributed throughout the network for the purpose of providing a domain-wide perspective of unauthorized events. In addition, EMERALD adopts a decentralized analysis scheme where each monitor has a defined detection policy in the form of resource objects. These resource objects control the operational semantics of the analysis engine. The resource objects also provide clear separation of control logic and detection policy. However, an intruder that successfully penetrates a monitor may find a spectrum of information from the resource objects, including the monitor's detection policy. Altogether, EMERALD adheres to the information dominance property in a weak sense. The detection policy and events form distributed knowledge known to large parts of the intrusion detection system.

5.2.2 JiNao

At a high level of abstraction, there are basically two flows of information in the Ji Nao system. One is the flow of information between the management application and the MIB agent. This flow includes queries to the MIB agent about the status of the remote JiNao system and requests to change or update that system. The other flow is in the opposite direction and consists of answers to the queries and notifications to the management application. Roughly, one could say that the requests correspond to a partial transfer of the detection policy and the queries and the notifications to a transfer of input events but, since the latter is a set of selected events, these events also form parts of the detection policy.
The transfer of these flows is handled by the SNMP protocol (see RFC 1157, RFC 1901-1910, RFC 2570-2575). The integrity and confidentiality of the flows are totally dependent on the security mechanisms of this protocol. In their report, the authors point out that it is vital that this channel is secure, i.e. encrypted and authenticated. Early versions of SNMP have very few of these mechanisms or, as in the case of SNMPv2, while some mechanisms are suggested in the standard, they are rarely present in practical implementations of the protocol. With these variants of SNMP, which lack both authentication and encryption, an attacker may both read the policy and change it without very great effort. This is also recognized by the authors themselves and is given as a reason for using SNMPv3.
Moving down in abstraction level and considering a local JiNao system, we find that the policy is spread across a number of modules. This dispersion of information, we believe, makes it harder to protect the detection policy in the local

system. As far as we understand, no explicit measures have been taken to protect the detection policy or the event information. There are however some firewall capabilities that are built into the local system, making it more difficult for an attacker to penetrate the JiNao agent.
All in all, our opinion is that, without SNMPv3, neither the protection of the local system nor the communication adheres to the information dominance property. With SNMPv3, the adherence is strongly dependent on the actual implementation of the protocol used.

5.2.3 GrIDS

In GrIDS, the flow of information is basically a stream of events traveling upwards in a hierarchical manner. In every step in this hierarchy, the amount of information is reduced according to rules in the nodes. The prototype system is, according to the authors, a proof of concept regarding scalable intrusion detection systems. In the authors' own words, ”A lot of our effort has gone into making the aggregation mechanism scalable, and allowing the system to be dynamically configurable so that it is still manageable when deployed on large scale”. In this process, no or little work has gone into securing the confidentiality and integrity of the system. This is something that the developers very clearly point out. This means that GrIDS on no account adheres to the information dominance property.

5.2.4 Bro

Bro is interesting because the developers have made great efforts to provide a secure and reliable intrusion detection system. Bro's clear separation between mechanisms and the detection policy is one of key features that support this claim. An installation of several Bro monitors can be viewed as a distributed IDS. Both event collection and detection are performed in a distributed manner. Because of the distributed detection scheme, no raw input event leaves its local domain. Hence, the confidentiality requirements of input events are fulfilled. A security breach in a single monitor would allow an intruder only to see events originating from a single domain. Unfortunately, Bro's design also requires the detection policy to be known to all monitors. This imposes a threat to the entire IDS, as a single monitor has sole knowledge of the site detection policy.

6 Conclusion and future work

This paper has discussed the information (knowledge) needed by different components of an IDS to operate. We have analyzed the information flow induced by organizing event collection and detection in various intrusion detection architectures (IDA). The knowledge needed, and the information flow produced by components of an IDA, lead to new security implications. We argue that these security implications suggest that there are certain security requirements on the

IDS itself, and we presented a number of concepts and requirements that we believe are essential to build a secure IDS. The concepts and requirements presented are separation of ”knowledge” and control logic, information dominance, and the confidentiality and integrity of audit data and the detection policy.
In the analysis of information flows, we introduced the notion of a fully distributed architecture and purposed that such a system has a high distribution of knowledge and therefore a large number of information flows. Such a system is therefore more vulnerable than are systems with a lower degree of distribution and must therefore, we believe, fulfill all of the requirements stated in this paper to be considered secure. However, even systems with a lesser degree of distribution still have information flows. This implies that our requirements should be considered in those cases as well and applied where appropriate.
We also introduced the idea that that confidentiality of input events and the confidentiality of the detection policy are conflicting parameters in current IDS systems (Figure 4). This leaves us with an open question: How is it possible to detect intrusions without violating the confidentiality of either? Preferably, we would like to have a scheme in which the input events are kept local to their source (not distributed outside the local domain), while, at the same time, the semantics of the detection policy are only known to a single centralized entity. Such a system would replace the question mark in Figure 4. Our research group is currently in the process of designing such a system.
Finally, we presented an analysis of some recent research prototypes to see to what degree these prototypes have taken into account the ideas presented in this paper. It is found that none of the research prototypes succeeds in keeping both input events and the detection policy confidential and integrity protected in fully distributed environments. However, some systems provide mechanisms that partly adhere to these requirements.

References

[Arm95] US Army. Field Manual (FM) 100-6, Information Operations (Working Draft), 8 July 1995. Department of the Army.

[Axe98] Stefan Axelsson. Research in intrusion detection systems: A survey. Technical Report 98-17, Department of Computer Engineering, Chalmers University of Technology, SE-412 96 Göteborg, Sweden, December 15 1998. Revised August 19, 1999.

[CCD+99] S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, J. Rowe, S. Staniford-Chen, R. Yip, and D. Zerkle. The Design of GrIDS: A Graph-Based Intrusion Detection System. Technical Report CSE-99-2, U.C. Davis Computer Science Department, January 1999.

[DDW99] H. Debar, M. Dacier, and A. Wespi. Towards a Taxonomy of Intrusion Detection Systems. *Computer Networks*, 31(8):805–822, April 1999.

[JGS+97] Y. Frank Jou, Fengmin Gong, Chandru Sargo, Shyhtsun Felix Wu, and Cleaveland W Rance. Architecture design of a scalable intrusion detection system for the emerging network infrastructure. Technical Report CDRL A005, Dept. of Computer Science, North Carolina State University, Releigh, N.C, USA, April 1997.

[KBS98] C. Kahn, D. Bolinger, and D. Schnackenberg. Communication in the Common Intrusion Detection Framework v 0.7. CIDF Working Group DRAFT Specification, June, 1998.

[Lab95] SRI Computer Science Laboratory. Next-generation Intrusion Detection Expert System (NIDES) - A Summary. Technical report, 1995.

[Neu95] P. G. Neumann. Architectures and formal representations for secure systems. Technical report, Final Report; SRI Project 6401; Deliverable A002, 1995.

[Pax98] Vern Paxon. Bro: A System for Detecting Network Intruders in Real-Time. In *Proceedings of the Seventh USENIX Security Symposium*, pages 31–51, San Antonio, Texas, January 1998. USENIX.

[PN97] P.A. Porras and P.G. Neumann. EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances. In *Proceedings of the Nineteenth National Computer Security Conference*, pages 353–365, Baltimore, Maryland, NIST / NCSC, July 1997.

[PN98] Thomas H. Ptacek and Timothy N. Newsham. Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection. Technical report, Suite 330, 1201 5th Street S.W, Calgary, Alberta, Canada, T2R-0Y6, 1998.

[SCCC+96] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle. GrIDS - A Graph-based Intrusion Detection System for Large Networks. In *Proceedings of the 19th National Information Systems Security Conference*, 1996.

[SCTP+98] S. Staniford-Chen, B. Tung, P. Porras, C. Kahn, D. Schnackenberg, R. Feiertag, and M. Stillman. The Common Intrusion Detection Framework - Data Formats, Internet Draft. CIDF Working Group DRAFT Specification, September, 1998.

[SvS98] A. Shamir and Nico van Someren. Playing hide and seek with stored keys, 1998. Weizmann Institute of Science, Israel; nCipher Corporation Limited, England.

[WP96] G. White and V. Pooch. Cooperating security managers: Distributed intrusion detection systems. *Computers & Security*, Vol. 15(5):441–450, 1996. Elsevier Science Ltd.

Paper F

Protecting Security Policies in Ubiquitous Environments Using One-Way Functions

Reprinted from

Proceedings of the First International Conference on Security in Pervasive Computing
Boppard, Germany
March 12-14, 2003

Protecting Security Policies in Ubiquitous Environments Using One-Way Functions

Håkan Kvarnström[*] Hans Hedbom Erland Jonsson

{*hkv,hansh,erland.jonsson*}*@ce.chalmers.se*
hakan.kvarnstrom@teliasonera.com
Department of Computer Engineering
Chalmers University of Technology
412 96 Göteborg, Sweden

Abstract

This paper addresses the problem of protecting security policies and other security-related information in security mechanisms, such as the detection policy of an Intrusion Detection System or the filtering policy of a firewall. Unauthorized disclosure of such information can reveal the fundamental principles and methods for the protection of the whole network, especially in ubiquitous environments where a large number of nodes store knowledge about the security policy of their domain. To avoid this risk we suggest a scheme for protecting stateless security policies using one-way functions. A stateless policy is one that only takes into consideration, the current event, and not the preceding chain of events, when decisions are made. The scheme has a simple and basic design but can still be used for practical implementations, as illustrated in two examples in real-life enviroments. Further research aims to extend the scheme to stateful policies.

Keywords: intrusion detection systems, detection policy, protection schemes, one-way functions

1 Introduction

The protection of computers and information systems is vital to the success of virtually every enterprise. Distributed system architectures connecting a large number of computers raise questions on how to better protect the information and resources of these systems. Access-control services such as firewalls [CZ95][CB94], are most often used to control access to systems and services, but the use of

[*]Håkan Kvarnström is also with TeliaSonera AB, SE-123 86 Farsta, Sweden

access-control components only may constitute a single-point-of-failure. A flaw in an access-control component could lead to loss or theft of information or computer resources by allowing an intruder to circumvent existing security measures. Intrusion detection systems (IDS) [Nor99] are techniques that attempt to detect unauthorized activities and suspicious events (behavior), that is, events that violate the effective security policy of a certain domain. They provide a second line of defense, allowing intrusions to be detected in the event of a breach in the perimeter defense. Furthermore, intrusion detection systems allow detection of misuse or suspicious user behavior.

Our work addresses a problem not often recognized, that is the security of the intrusion detection system itself. This is important for several reasons. First, it is obvious that the functionality of the IDS, i.e. its ability to operate as expected, depends strongly on the security of the IDS itself, i.e. its ability to resist attacks. If an intruder succeeds in mounting an attack against the IDS, either by taking it over, corrupting its input data or its detection policy, or in other ways duping the IDS, it will no longer generate alarms for attacks launched against the target system. A report by Newsham and Ptacek [PN98], identifies several successful attacks against existing intrusion detection systems. Further, the information contained in the IDS (e.g. audit data) may be misused by an intruder to gain knowledge about the target system (e.g. weaknesses, protocols used etc.) that would indeed facilitate attacks. However, the most serious problem, especially in large distributed systems, is that the IDS or other security mechanisms could contain information such that the unauthorized disclosure of it would endanger the security of the whole computer network. Examples of sensitive information are overall security properties, lists of trusted hosts and information about unprotected vulnerabilities and dubious system configurations. Pervasive computing can be highly distributed, which seriously alters the threat model for an intrusion detection capability. We argue that traditional IDS designs are vulnerable when operating in such environments.

The argumentation above holds for firewalls as well, and indeed for any security mechanism that stores policies or other security-related information. Here we will mainly discuss IDSs. A primary issue for an IDS is how to conceal its policy. This can be achieved by using encryption, as discussed by Neumann [Neu95] for the NIDES system [Lab95]. His method requires storing keys on the local host, which virtually means that anyone that gains access to the host also gains access to the key and thereby the policy. Hiding the key does not help in the long run, as Shamir and van Someren demonstrates in [SvS98]. Besides, as intrusion detection systems extend from mainly having been a centralized component to a distrubuted computing platform (e.g personal firewalls), key distrubution becomes an issue. A high degree of distribution also increases the risk of the policy being disclosed as a result of increased exposure.

Here we suggest a policy protection scheme using one-way functions which to our knowledge, has not been discussed previously. Thus, this is a first attempt towards this type of protection. In particular, we suggest a method for protecting stateless security policies. We are aware that stateless policies can only be used to describe a limited set of policy rules but are nevertheless convinced that they can serve as a starting point for discussion and for constructing schemes to handle more complex types of policies for firewalls and IDSs.

2 The need for policy protection

2.1 Policies

Security mechanisms such as IDSs and firewalls are equipped with a decision function for when an IDS should send an alarm or whether a firewall should block or allow network traffic. To make these decisions the systems use some form of rule set (or rule base) as the basis for the decision. We call this rule set the *detection policy* in the IDS case and the *filtering policy* in the firewall case. When there is no need to separate the two we will collectively refer to them as a policy. Examples of policies are:

- Rules for access-control to objects (e.g. access control lists)
- Misuse signatures
- Statistical user or system normal behavior

The policy is usually expressed in a description language describing the *event* or *combination of events* that is considered inappropriate (or, in the firewall case, usually appropriate). In this paper, we define an event the occurrence of a single activity which is registered and stored in an audit log. The mechanisms presented in this paper are not limited to events of a certain type or having certain characteristics, as long as they can be coded or described as a (binary) string. A typical event would be an audit log record describing user activity (e.g. accessing a web page, logging into a service etc.). Table 1., shows five events generated as a result of remote logins to a server host.

Table 1: Five events generated by remote logins to the host ”jellybean”.

Jun 24 18:19:42:jellybean sshd[1084049]: log: Connection from 192.168.0.1 port 56722
Jun 24 18:21:12:jellybean sshd[3472342]: log: Connection from 192.168.0.244 port 16239
Jun 24 19:29:14:jellybean sshd[1265421]: log: Connection from 192.168.0.123 port 54346
Jun 24 20:19:01:jellybean sshd[9934742]: log: Connection from 192.168.0.220 port 16222
Jun 24 21:45:41:jellybean sshd[1124234]: log: Connection from 192.168.0.11 port 201

A rule in a filtering policy could state for example that connections between the internet and the intranet are allowed only if they originate from the intranet. A rule in an IDS could state that logfile entries containing the string ”login successful” and having a time stamp between 11pm and 4am indicate a possible intrusion.

A typical intrusion detection system's rule base can be divided into two parts, the first part consisting of a set of well-known (standard) attack signatures, often provided by the supplier of the IDS. This part is updated regularly, similar to virus scanners. The second part of the rule base consists of site-specific threats and vulnerabilities, which may be unique to the target system that it protects. For example, the target system could be configured to run old versions of software for compatibility with legacy systems or to use applications and protocols known to be vulnerable to attack such as NFS and NIS. In the next section, we show that the latter type of rule base poses a threat to the target systems.

2.2 The need for protection

The basic reason for using an IDS is to improve the security of the target system by adding a second layer of defense. However, there are also inherent security concerns for this second layer as it may introduce new security risks [HKJ99][HLJ99]. These concerns involve the protection of information and information flow within the IDS and how the information can be used for illicit purposes. This section discusses the security properties of an IDS and argues that the confidentiality aspect of the detection policy is one of the most important requirement.

2.2.1 Confidentiality and integrity of audit data.

Audit data generated by the target systems within an IDS domain may contain sensitive information not to be disclosed outside the domain. The information may be about users and target systems or may be application related data. In some cases, the mere existence of an event may be confidential as it reveals some form of activity. Audit data should neither be subject to insertion, deletion or alteration as demonstrated in a paper by Ptacek and Newsham [PN98]. On the other hand, a breach of confidentiality or integrity of audit data presents less of a risk than in the case of policies. The result of a confidentiality or integrity breach can often be limited to a missed detection or a false alarm.

2.2.2 Confidentiality and integrity of the policy.

Similarly, an attacker may break into an intrusion detection system and disable the detection mechanism so that target system attacks can be launched without being disclosed. However, if and when the attack is eventually detected, the system can be taken offline, at which integrity can be restored and it is finally brought back into operation. The damage is limited to the loss of detection capability over a period of time. In contrast, if the rule base is disclosed during the attack, the attacker can learn about inherent vulnerabilities of the target system, knowledge that would remain even when the integrity of the IDS is restored. The inherent vulnerabilities of the target system may not be possible to fix and, hence, permanent damage has occurred. Thus, the main reason for our concern about the confidentiality of the rule base is that a breach of confidentiality is irreversible.

As an example, consider a target system running NFS to share files between clients. A policy rule could be defined to detect the use of NFS (UDP) packets containing certain strings (e.g. /etc/passwd) between a certain NFS client and a dedicated server. If such a policy rule were stored in cleartext in the rule base, an attacker would learn about the inherent vulnerability that exists and possibly exploit it to gain access to the information in the system. However, this particular attack signature is target-specific and an attacker may not discover the vulnerability when a vulnerability scan of known attacks is made.
Our experience in applying intrusion detection systems is that the rule base containing target-specific attack signatures increases rapidly as the complexity (e.g. the number of hosts and services) of target systems grows. This is different from the standard attack rule base, which is not affected by the complexity of the target system. However, vendors can utilize our protection scheme to protect their rule base as it may provide a competitive advantage not to disclose the workings of attacks not known to competitors. Also, by keeping the rule-base confidential, software vendors (e.g. virus scanners and IDS) could include detection of innovative attacks (i.e attacks not yet known to the hacker community) without the risk of reverse-engineering the policy for the purpose of writing malicious code to exploit the vulnerability.

3 Architectural implications

An increased use of network encryption and virtual private networks providing end-to-end security between systems, makes it difficult for intrusion detection systems to monitor events in the target system or in the network. One solution to this problem is to execute the detection system on the end-systems, at which the encryption terminates (e.g. a personal IDS). This means that the security policy will be distributed, giving a distributed intrusion detection architecture (IDA) [HKJ99]. It is clear that the IDS' ability to protect its detection policy is highly dependent on the intrusion detection architecture.
An IDS is distributed when the different components of the detection system are distributed in some respect. Figure 1 illustrates a distributed intrusion detection architecture where components are located in different domains.

In a strictly centralized system, the detection policy is known only to a small part of the system, while in a distributed intrusion detection architecture, the policy is distributed to all of the decision functions that participate in the system (as described by Figure 1). In addition, the end-systems are not dedicated to detecting intrusions (i.e. many other applications execute on the systems), which alters the threat model for the IDS. An end-system such as an office PC is often under the control of the user, which makes it more difficult to enforce the security policy in practice. Thus, the detection policy may be known to a possibly large number of entities. This implies that the security of the policy is dependent on the security of all the entities that have knowledge of the policy and of the security of the distribution channel. Deploying IDS capabilities to ubiquitous environments may

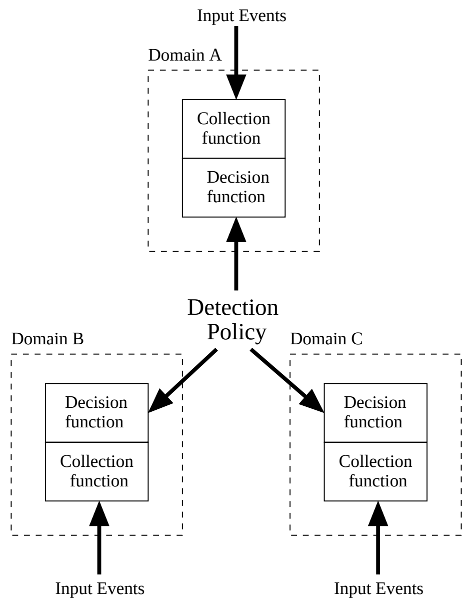

Figure 1: A distributed intrusion detection architecture (IDA)

result in a vast number of entities having knowledge about the policy. Hence, we need security mechanisms to allow the distribution of policies without risk of compromising its confidentiality.

4 Protecting a policy using one-way functions

One way of solving the problem of how to protect the policy may be to use strong one-way functions. In this section we discuss this concept and how they can be used to protect stateless policies.

4.1 The concept of one-way functions

One-way functions are one of the fundamental building blocks of cryptography. Many security mechanisms providing such security services as authentication, integrity protection and confidentiality depend on the unique properties provided by one-way functions. Informally, a one-way function [Gon95] is a function $f : S \rightarrow T$, where S and T are any two sets, such that:

1. for any $x \in S, f(x)$ is ”easy” to compute,
2. given the information that $f(x) = y$, there is no ”feasible” way of computing x for any reasonably large proportion of the y belonging to T,

3. for any $x, z \in S, f(x) \neq f(z)$. This property states that the function must be collision free in the sense that no two values of S may result in the same y belonging to T.

Assuming that set is sufficiently large, an exhaustive search will be computationally infeasible. The UNIX password protection scheme [MT79][FK90] is an example of a security mechanism making use of one-way functions. It provides confidentiality of the users' passwords, thus preventing disclosure of the passwords even though the password file itself is disclosed. Several candidate one-way functions have been proposed. Factoring and the discrete logarithm problem are two well-known ”hard” mathematical problems that are often used to create one-way functions.

4.2 Protection principle

Informally, the policy classifies an incoming event (or sets of events) into predefined categories such as *legal events, intrusion attempts, intrusions* etc. In its simplest form a security policy states what patterns or signatures of events are authorized / unauthorized. A *default accept* policy would search for events having a certain signature and classify them as unauthorized whereas a *default deny* policy makes the assumption that all events are unauthorized except those that explicitly matches a defined signature. Most rule-based IDSs have taken a default accept standpoint due to the difficulty of defining all authorized events, while most firewalls have a default deny policy, only letting through the events matching the policy. The following simple example show how signatures can be used to detect policy violations in a default accept policy. Consider a set of input events, $x_1, x_2, \ldots, x_n \in X$, all of which are represented by *k*-bit binary strings. Further, the set $u_1, u_2, \ldots, u_m \in U$ is the set of ”signature strings” that identifies an unauthorized event. Whenever $x_i = u_j$, where $i \leq n, j \leq m$, the event being analyzed matches a detection signature and an alarm is raised. The detection scheme is fairly simple as it only involves comparing events over X with all strings in U searching for identical pairs. Now consider the set $u'_1, u'_2, \ldots, u'_m = f(u_1), f(u_2), \ldots, f(u_m) \in U'$, where f is any cryptographically strong one-way function. A hash is calculated and stored for each signature. Because of the inherent properties of the one-way function, it is hard to deduce any $u \in U$ given $f(u) \in U'$. Thus, U' can be made publicly available without compromising the secrecy of U. The computational effort to successfully deduce $u \in U$ is equal on average to an exhaustive search of $1/2$ of the domain of U. Thus, assuming $u_1, u_2, \ldots, u_m \in U$, where $|u|, u \in U$ has a binary length of *k* bits. The computational effort to find $u \in U$ given $f(u) \in U'$ would (in average) require 2^{k-1} operations.

Detecting policy violation in the default accept case is a straightforward process of applying the same one-way function to all input events $x \in X$ and compare the resulting values to the stored values of U'. If a match is found, the input event is an unauthorized event and an alarm is raised. Many intrusion detection sys-

tems utilizes simple string matching to find unauthorized patterns in the input data. For example, an IDS could search a UNIX syslog looking for strings containing the pattern ”su root failed” which would indicate a failed attempt to gain administrative privileges on the system. If one-way functions are used to hide the string patterns, it is very hard for an intruder to identify the detection policy of the system.
Policy violations in the default deny case can be handled in a similar manner as discussed above. The only difference in this case is that U, and thus U', will contain permitted events and actions are taken if $f(x), x \in X$ is not a member of U'.

5 Handling variabilities

In a normal case one input value will lead to a unique output value from the one-way function, meaning that even a small change in the input value will generate a completely different result. This is a desired property in the traditional use of one-way functions but undesirable in the policy case. This section discusses reasons for handling variabilities in the stateless policy case and offers ideas about how this may be achieved by different methods and elaborates on what we believe to be the shortcomings and merits of the different methods.

5.1 Why do we need to handle variabilities?

When describing rules in a policy it is useful to be able to handle variables and express intervals. This means that the protection scheme for the policy must also be able to handle this variability in some way. For example, it may be desirable to state in the policy that certain actions are forbidden to all users or that a given user is not allowed to login to certain network addresses. If we divide the information into fields, there will be fields containing variables in both the cases above. In the first case the value in the forbidden action field will be constant while the value in the user field is variable `(FIX<Action>, VAR<User>)`. In the second case the value in the user field is constant but the value in the network field is variable `(FIX<User>, VAR<IP Address>)`. The variability in the latter case could be defined as the the range of network addresses that are permitted or not permitted.

5.2 A first approach

One naive approach toward handling variability may be to apply the one-way function on every possible combination of field contents, although this would lead to a very large collection of values that need to be compared. This might be tolerable in small systems, but would be unacceptable in large systems. The next approach would be to skip the variable fields and simply apply the one-way function on the fixed fields. This solves the problem in the first case, where we do not care who the user is, i.e all possible values of the user field are considered illegal, but does not solve the problem in the second case above since not all of the possible values in the network field might be considered illegal. It also has a serious side effect. By applying the function about selected fields only, we are giving away information on which fields we are interested in, which in turn could point an intruder towards the kinds of attacks we are looking for.

5.3 Using fuzzy commitment to handle intervals

Fuzzy commitment is a a way of doing commitment suggested by Ari Juels and Martin Wattenberg [JW99]. It is essentially a method that accepts a certain amount of fuzziness (i.e variability) in the witness[1] used. Its main use is in the area of authentication by means of biometrics (e.g. fingerprints etc.), where it is almost impossible to get exactly the same result from two consecutive scans of the same object. For example, a thumb is never placed in the same way twice on a thumb scanner. Juels et al. show that using a combination of error correction methods and one-way functions one can create a commitment scheme that will accept different witnesses as long as they differ only up to a controllable threshold. In essence the method works as follows: Assume that we have a witness x and an error correction function f that corrects codewords from the set C. Now, choose a $c \in C$ and calculate $d = x - c$. Commit c by using a strong one-way function and store d. To decommit c a user has to give a witness x' that is sufficiently close to x so that the correction function can correct $x' - d$ to c. The amount of fuzziness accepted is thus dependent on the number of errors accepted by the correction function and on the value d.
Fuzzy commitment can be used to handle intervals in the following way. The basic idea is to let all the values in an interval hash to the same cryptographic hash-sum. In this case we do not really need a proper error correction function but merely a function that groups values together. Such functions can be used in fuzzy commitment, e.g. Juels et al. [JW99] use a lattice rounding function rather than a proper error correcting function as an example of how the scheme works. The lattice rounding function in their example is a function that rounds points in the plane to the nearest multiple of 100x100. If we interpret the values in the variable fields as integers, we can use a function that truncates values downwards to the nearest multiple of the selectable integer i, e.g. if $i = 10$ then $0 \ldots 9 \rightarrow 0, 10 \ldots 19 \rightarrow 10$ and so on. We call this function g. Such a function can easily be generalized by making i a parameter of the function (e.g. $g(i, x) = i\lfloor x/i \rfloor$). Let witness x represent the lower end of the interval and let i be the width of the interval. Randomly choose $c \in C$ where $C = \{\forall c \in C : ix\}$, where x is an integer. Calculate all other parameters according to the rules above. To test later whether a witness x' is within the interval the calculations described above for decommitment are used exchanging f with g.

[1] The witness is the value that is to be committed or the value that is compared with a committed value after it has been transformed with a strong one-way function

5.3.1 Example (commitment).

This simple example shows how integer intervals can be committed. The commitment scheme can be applied to any type of variability, however, using a suitable encoding of the input fields.
Assume that we want to commit the integer interval $234 \ldots 243$. In this case $i = 10$ since the width of the interval is 10. Set C will be all integer multiples of 10 and we randomly choose $c = 410$ from this set. Since the lower end of the interval is $234, x = 234$ and $d = x - c = 234 - 410 = -176$. c is then committed using a strong one-way function.

5.3.2 Example (decommitment).

Assume that the commitment in the example above has been made. Further assume that we want to find out whether 240 is within the interval i.e $x' = 240$. We first calculate $c' = x' - d = 240 - (-176) = 416$. By applying function g on 416, the result is 410. We then commit 410 using the same strong one-way function as in the example above and compare the results.

5.4 The shortcomings of using fuzzy commitment

The great disadvantage of using fuzzy commitment is that the error correction function will give the codeword in the clear as output. With this value and the known value of variable d, it is very easy to calculate the interval. The implications of this is that one lucky guess will reveal the whole interval. This is a small problem if the set of possible values is large as compared to the values within the interval range. In this case, the probability of guessing a correct value is small, and an exhaustive search is highly time consuming. However, in many ”real world” cases of specifying intervals for detection purposes, the set of possible values will be small and / or the intervals will be very wide. The probability of a lucky guess is thus high and it is relatively easy to make an exhaustive search. In this light, fuzzy commitment by itself is not a perfect solution to the interval problem. In the next section we discuss a method to make it more difficult to make this deduction.

5.5 Making value deduction harder

The possible input values for the one-way function are usually few enough to permit an exhaustive search. This is basically because we are dividing the input stream into fields and separately applying the one-way function to each field but also because we in essence are dividing the possible values of the fields into equivalence classes, thereby making it easier to find one input value that maps to a valid output value. Of course, one could divide the data in such a way that the input fields would be large enough. We believe, however, that there are natural divisions in most applications based on the format of log entries and other data structures in the system. The individual fields in these structures usually have a small domain of possible input values, but the combined structure in itself has a much larger domain. By applying the function repeatedly in a tree-like manner it is possible to use the input domain of the structure as the input domain of the resulting one-way function and still be able to handle variabilities as previously discussed. The method works as follows:

1. Calculate the individual fields of the structure by applying an appropriate method, i.e a one-way function on the fixed fields and a method for handling variability on the variable fields. Note that the method for handling variablility includes the application of a one-way function.

2. Concatenate the resulting values pair-wise and apply a one-way function to each of the pairs

3. Repeat step 2 until only one value remains. This is the value that is compared with the policy.

The process above is graphically described in Figure 2. It can be generalized by always applying the method for variability and restricting the variability for the fixed fields, e.g setting the interval length to 1 if the fuzzy commitment approach is used.

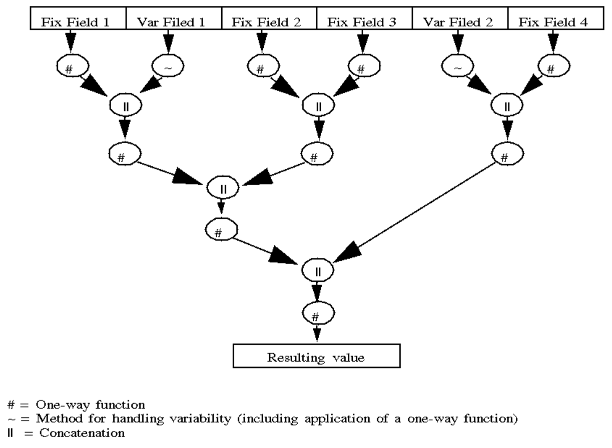

Figure 2: Method one for handling variabilities

If all the fields in the structure are variable or if the structure is small the domain may still be too small. However, it is significantly larger than the input domain of the individual fields.
An alternative to the method described above is the following:

1. Calculate the individual fields of the structure by applying an appropriate method (i.e. a one-way function on the fixed fields and a method for handling variability on the variable fields).

2. Concatenate all the resulting values and apply a one-way function to the concatenated value.

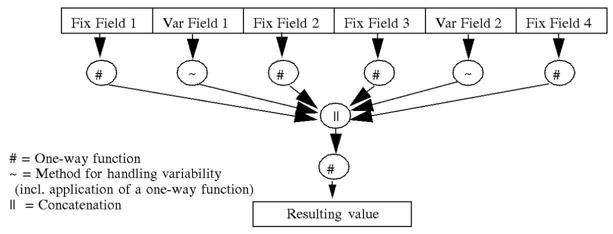

Figure 3: Method two for handling variabilities

This process is graphically described in Figure 3. The big difference between the first approach and the second approach, besides a reduction in the use of one way functions, is that the former can be used to implement a hierarchical matching scheme. Thus, matches can be found for individual fields. This may be useful if it is necessary to adapt the matching path based on intermediate hash values in the tree (e.g applying different field structures for different categories of attacks).

6 Examples of the proposed protection scheme

This section gives two simple examples of how the method presented in here can be used. The first illustrates how the audit logs in Table 1 on page 165 can be used for detection and how variability is achieved. A second example shows how a simple snort [Roe99] rule can be protected using our scheme. Snort is an open source network intrusion detection system capable of performing real-time traffic analysis and packet logging on IP networks.

6.1 Remote login policy violations

In the first example, a template is constructed that will be used for filling in information contained in the logs (Table 1 on page 165).

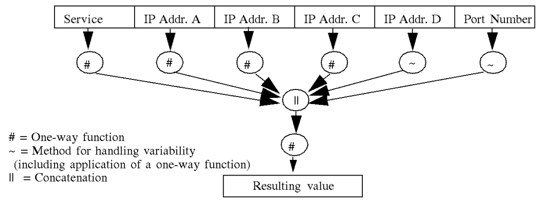

Figure 4: Template for detecting remote login policy violations

In Figure 4 four fixed fields and two variable fields are used to encode an event for remote login attempts. The service field contains the name of the service ”sshd”. The first three IP address fields (A-C) contain fixed values for the subnet in which the IDS resides (e.g ”192”, ”168”, ”0”). The variable IP address field (D) contains the range of hosts for which remote login is a policy violation (e.g. a value between 0 and 255). The last variable field, ”Port Number”, enables us to disallow login attempts from certain port numbers such as non-privileged ports (above 1024) on UNIX systems. During the detection phase, the input events are encoded one by one using the template and compared to the set of stored hash values indicating a policy violation.

6.2 Buffer overflow attacks

The second example shows how we can protect a simple snort rule with the purpose of detecting a buffer overflow attack. The rule base of snort (i.e. the detection policy) consists of rules matching bit patterns in the header / payload, ranges of IP numbers and port numbers, header bits etc. The following simple rule (Table 2) aims to detect a buffer overflow attack in an application X. The signature of the attack is characterized by the hexadecimal value ”94B5 C1AF FFCB 16EF” followed by the string ” / bin / sh”. The rule also defines the vulnerable targets to have an IP address in the range 192.168.1.0-255. Any source address originating from any TCP port will trigger an alert as long as the attack is targeted at port 7890.

Table 2: Snort rule detecting a buffer overflow attack.

```
Alert tcp any any -> 192.168.1.0/24 7890
(content: "|94B5 C1AF FFCB 16EF|/bin/csh";offset: 3;
depth: 22; msg:="Buffer overflow attack in Application X!";)
```

During detection, a template is used that is similar to that in the first example.

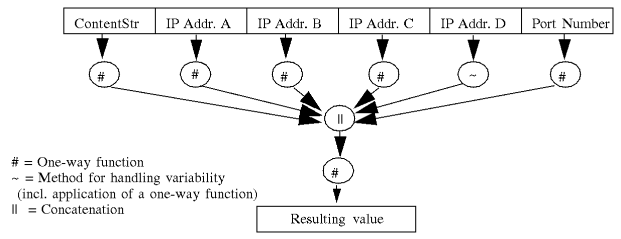

Figure 5: Template for detecting buffer overflow attacks

Note that the template can be reused for all rules having similar parameter values. To gain flexibility, the snort’s rule description language may be extended

to describe the templates needed for each rule. The offset and depth parameters can guide the process of filling in the correct bit string in the first field of Figure 5.
To reverse engineer the above rule, an exhaustive search over all possible policies would be necessary. The 72-bit *ContentStr* (64-bit hex value + 8 bit text string) alone would theoretically require 2^{72} operations. By eliminating illegal combinations of machine code and arguments, the search space can probably be reduced. However, we have not further studied the entropy of machine code for various platforms nor have we found any studies discussing the topic. The entropy would yield a rough estimate of the effective search space. In addition, the port number adds another 16-bits of complexity. Note that all fields must contain correct values for the resulting value to match a valid signature. This prevents a divide-and-conquer attack in which the fields are broken one by one.

7 Discussion and future work

There are a number of open questions regarding the protection of policies using one-way functions. Below we discuss some of them and give indications for future work.
First, we have only discussed the protection of stateless policies, i.e. policies that take only the current event into consideration. We are fully aware that this poses a certain limitation, but the concept is nevertheless useful and applicable as our examples show. A more elaborated protection scheme for stateful policies could be used to describe more complex threats and attacks and we are currently working to develop such a scheme.
Second, we have discussed the need to handle variability and suggested solutions as to how to handle intervals although with some limitations. One limitation is the disclosure of the variable range once a single value is found. In order to solve this problem we would need a strong cryptographic one-way hash function that generically and controllably would hash different values to one hash value. Basically, this function must divide the possible values of the variable into equivalence classes based on the values in which we are interested and then hash the individual members of each equivalence class into one hash value. In order for this to be generic, the function must be able to take the relation that defines the equivalence classes as a parameter and it must be possible to define arbitrary relations without disclosing the individual values. We do not know of such a function, nor whether it is possible to construct it. Collisionful one-way hash-functions are described in the literature e.g. [Gon95][BSNP97][BSNP96], but they do not solve the problem discussed above. Moreover, even if the interval disclosure problem were solved, a range could almost as easily be found using linear search once a single value within the range has been found.
Finally, there is the question of performance. Some of the systems, e.g. IDSs in large computer systems or firewalls in high capacity networks, must be able to handle large amounts of information in a limited time. The performance of an

IDS described in this paper is likely to be significantly lower than a traditional signature-based intrusion detection system. We do not claim this scheme to be efficient but instead focus on confidentiality properties and the feasibility of protecting the policy from unauthorized disclosure. In a real-life system, traditional signature detection techniques can be used to detect well-known attacks and be complemented with this technique for attacks that are target-specific. This would limit the decrease in performance to a minimum.

8 Conclusion

This paper discusses the problem of protecting the security policy of a security mechanism, such as an intrusion detection system or a firewall operating in distributed or ubiquitous environments. We have noted that the policy contains sensitive information that can be misused by an attacker in order to avoid detection or to render the detection system useless. Still worse, it could provide information that would facilitate intrusions into the target system or even extend to logically connected systems within or outside of the actual network. Thus the policy is crucial for the function of the security mechanism and the system it is supposed to protect.
We have suggested that one-way functions may be used as a means to protect the policy. This approach has certain shortcomings, however. Foremost of these is that normal one-way function schemes can only be used on constant values, so that even small variations in input values give completely different output values. This is a desirable property for the ordinary use of one-way functions. In our case, however, we would like a complete equivalence class of data to be hashed into one specific hash value and have suggested a clustering method based on fuzzy commitment that would accomplish this for some, but not all, types of variability in the input data. The drawback of clustering is that it increases the probability of guessing a correct match or carrying out an exhaustive search. It is also a fact that the very nature of intrusive events puts a bound on the possible cases, thereby making it easier to make a good guess. To counter this we have suggested a method that expands the possible domain by grouping values together and repeatedly applying one-way functions in a tree-like manner, thereby making guessing and exhaustive searches more difficult. Finally, we present two concrete examples of how to apply our protection scheme in real-life situations.

References

[BSNP96] S. Bakhtiari, R. Safavi-Naini, and J. Pieprzyk. On Selectable Collisionful Hash Functions. In *Proceedings of the Australian Conference on Information Security and Privacy*, pages 287–292. LNCS No. 1172, 1996.

[BSNP97] S. Bakhtiari, R. Safavi-Naini, and J. Pieprzyk. On the Weakness of Gong's Collisionful Hash Function. *Journal of Universal Computer Science*, pages 185–196, 1997.

[CB94] W.R. Cheswick and S.M. Bellovin. *Firewalls and Internet Security: Repelling the Wily Hacker*. Addison-Wesley, 1994.

[CZ95] D.B. Chapman and E.D. Zwicky. *Building Internet Firewalls*. O'Reilly & Associates, Inc. September, 1995.

[FK90] D.C. Fieldmeier and P.R. Karn. UNIX password security - ten years later. In *Proceedings of Advances in Cryptology CRYPTO 89, LNCS 0302-9743*, pages 44–63. Springer Verlag, 1990.

[Gon95] L. Gong. Collisionful keyed hash functions with selectable collisions. *Information Processing Letters 55*, pages 167–170, 1995.

[HKJ99] H. Hedbom, H. Kvarnström, and E. Jonsson. Security Implications of Distributed Intrusion Detection Architectures. In *Proceedings of the 4th Nordic Workshop on Secure IT systems - Nordsec 99*, pages 225–243, Stockholm, Sweden, 1999.

[HLJ99] Hans Hedbom, Stefan Lindskog, and Erland Jonsson. Risks and Dangers of Security Extensions. In *Proceedings of the Security and Control of IT in society-II (IFIP SCITS-II)*, pages 231–248, Bratislava, Slovakia, June 15-16 1999.

[JW99] A. Juels and M. Wattenberg. A Fuzzy Commitment Scheme. In *Proceedings of the Second ACM Conferens on Computer and Communication Security CCS'99*, Singapore, 1999.

[Lab95] SRI Computer Science Laboratory. Next-generation Intrusion Detection Expert System (NIDES) - A Summary. Technical report, 1995.

[MT79] R. Morris and K. Thompson. Password security: A case history. *Communications of the ACM*, 22(11):594–597, 1979.

[Neu95] P. G. Neumann. Architectures and formal representations for secure systems. Technical report, Final Report; SRI Project 6401; Deliverable A002, 1995.

[Nor99] S. Northcutt. *Network Intrusion Detection : An Analyst's Handbook*. New Riders, 1999.

[PN98] Thomas H. Ptacek and Timothy N. Newsham. Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection. Technical report, Suite 330, 1201 5th Street S.W, Calgary, Alberta, Canada, T2R-0Y6, 1998.

[Roe99] M. Roesch. Snort - Lightweight Intrusion Detection for Networks. In *Proceedings of the USENIX LISA '99 Conference*, November 1999.

[SvS98] A. Shamir and Nico van Someren. Playing hide and seek with stored keys, 1998. Weizmann Institute of Science, Israel; nCipher Corporation Limited, England.

Paper G

Protecting Stateful Security Policies using One-way functions

Reprinted from

Proceedings of the Asia Pacific Information Technology Security Conference (AusCERT 2004)
Gold Coast, Australia
May 23-27, 2004

Protecting Stateful Security Policies using One-way functions

Håkan Kvarnström[*] Hans Hedbom[†] Erland Jonsson

{*hkv,hansh,erland.jonsson*}*@ce.chalmers.se*
hakan.kvarnstrom@teliasonera.com
Department of Computer Engineering
Chalmers University of Technology
412 96 Göteborg, Sweden

Abstract

This paper addresses the problem of protecting security-related information, such as the detection policy of an Intrusion Detection System, in distributed computer systems. Unauthorized disclosure of such information, possibly stored in a large number of nodes, can reveal the fundamental principles and methods for the protection of the whole network domain, thereby introducing new risks and vulnerabilities. To counter this, we suggest a protection scheme. The scheme extends previous research on protection of stateless policies to stateful policies. A stateful policy can deal with temporal occurrences of events and not only a single event as in the stateless case. It can be described by a finite state machine corresponding to regular languages. The protection scheme uses one-way functions and can be seen as a way to provide obfuscation and thus prevent reverse engineering. We provide a complexity analysis of its ability to resist attacks. An example is given to show its applicability.

Keywords: intrusion detection systems, detection policy, protection schemes, one-way functions, finite state machines

1 Introduction

The protection of computers and information systems is of vital importance for virtually every enterprise, in particular when using a distributed system architecture with a large number of computers. Protection is achieved by a range of

[*]Håkan Kvarnström is also with TeliaSonera AB, SE-123 86 Farsta, Sweden [†]Hans Hedbom is also with Karlstad University, SE-651 88 Karlstad, Sweden

security-enhancing mechanisms, but even so full protection will not result. Thus, as a second line of defense, intrusion detection systems (IDS) [Nor99] are used. Such systems deploy techniques that attempt to detect unauthorized activities and suspicious behavior, i.e. events that violate the effective security policy. A problem that is often neglected is that the security of the intrusion detection system itself is an important issue. This is so for several reasons. First, the functionality of the IDS and its ability to operate as expected, depends strongly on its security, i.e. its ability to resist attacks. If an intruder succeeds in mounting an attack against the IDS, either by taking it over, corrupting its input data or its detection policy, or in other ways duping the IDS, it may no longer generate alarms for attacks on the target system [PN98]. Further, the information contained in the IDS may be misused by an intruder to gain knowledge about the target system that would indeed facilitate attacks. Examples of such sensitive information are: security properties, lists of trusted hosts and information about unprotected vulnerabilities and dubious system configurations. Finally, the need for security is even more emphasized if the IDS architecture is distributed. In a centralized system, the detection policy is known only to a small part of the system, while in a distributed intrusion detection architecture, the policy is distributed and known to a possibly large number of entities. This implies that the security of the policy is dependent on the security of all the entities that have knowledge of the policy and of the security of the distribution channel. Hence, we need security mechanisms to allow the distribution of policies without risk of compromising its confidentiality.

Thus, traditional IDS designs are vulnerable. They may introduce new security risks [BSNP96][HLJ99] or even impose a threat to the targets they are supposed to protect, when operating in certain environments. As a remedy, we suggest a policy protection scheme using one-way functions. It extends previous work on the protection of stateless security policies [KHJ03] by introducing a transformation mechanism that allows the protection of more complex (stateful) policies. Our scheme protects policies that can be described by regular languages and is not applicable to policies of arbitrary complexity. However, many real-world policy-enforcement systems use regular languages, e.g. intrusion detection systems with string matching using regular expressions [VRKK03].

2 The need for policy protection

A primary issue in an IDS is how to conceal its policy from unauthorized parties. This can be achieved by using encryption, as discussed by Neumann for the NIDES system [Neu95][Lab95]. His method requires storing keys on the local host, which means that virtually anyone that gains access to the host also gains access to the keys and thereby the policy. Attempts to hide the key do not always help as Shamir and van Someren demonstrate in [SvS98]. Besides, as intrusion detection systems extend, from having been a mainly centralized component to a distributed computing platform, key distribution and their protection become issues. A high degree of distribution also increases the risk of the policy being disclosed as a result of increased exposure [HKJ99].

2.1 Policies

IDSs and firewalls, are equipped with a decision function which decides when the IDS should send an alarm or when a firewall should block or allow network traffic. To make these decisions the systems use some form of rule set (or rule base) as the basis for the decision. Depending on the domain in question, this rule set can be called *detection policy* (IDS) or *filtering policy* (firewall). When there is no need to separate the two we will collectively refer to them as the *(security) policy*.

The policy is usually expressed in a description language describing the *event* or *combination of events* that is considered inappropriate (or, in the firewall case, usually appropriate). In this paper, we define an event as the occurence of a single activity which is registered and stored in an audit log. The mechanisms presented in this paper are not limited to events of a certain type or having certain characteristics, as long as they can be coded or described as a (binary) string. A typical event would be an audit log record describing user activity (e.g. accessing a web page, logging into a service etc.). Table 1 shows five events generated as a result of remote logins to a server host.

Table 1: Five events generated by remote logins to the host ”jellybean”

Jun 24 18:19:42:jellybean sshd[1084049]: log: Connection from 192.168.0.1 port 56722
Jun 24 18:21:12:jellybean sshd[3472342]: log: Connection from 192.168.0.244 port 16239
Jun 24 19:29:14:jellybean sshd[1265421]: log: Connection from 192.168.0.123 port 54346
Jun 24 20:19:01:jellybean sshd[9934742]: log: Connection from 192.168.0.220 port 16222
Jun 24 21:45:41:jellybean sshd[1124234]: log: Connection from 192.168.0.11 port 201

For example, a rule in a filtering policy could state that connections between the internet and the intranet are allowed only if the traffic originates from the intranet. A rule in an IDS could state that a logfile entry containing the string ”login successful” and having a time stamp between 11pm and 4am indicate possible intrusions.

A rule base of a typical intrusion detection system can be divided into two parts, the first part consisting of a set of well-known (standard) attack signatures, often provided by the supplier of the IDS. This part is updated regularly, similarly to virus scanners. The second part of the rule base consists of site-specific threats and vulnerabilities, which may be unique to the target system that it protects. For example, the target system could be configured to run old versions of software for compatibility with legacy systems or to use applications and protocols known to be vulnerable to attacks such as NFS and NIS.

2.2 Confidentiality and integrity of the policy

An attacker may break into an intrusion detection system and disable the detection mechanism so that target system attacks can be launched without being disclosed. However, when the attack is detected, the system can be taken offline and brought back to operation with its integrity restored. The damage is limited to the loss of detection capability over a period of time. In contrast, if the rule base is disclosed during the attack, the attacker can learn about inherent vulnerabilities of the target system, knowledge that would remain by the attacker. The inherent vulnerabilities of the target system may not be easy to fix and, hence, permanent damage may have occurred. Thus, the main reason for our concern about the confidentiality of the rule base is that a breach of confidentiality is irreversible.

Our experience in applying intrusion detection systems is that the rule base containing target-specific attack signatures grows rapidly as the complexity (e.g. the number of hosts and services) of the target systems grows. This is different from the standard attack rule base, which is not affected by the complexity of the target systems. However, vendors may also like to prevent disclosure of their rule base as it may provide a competitive advantage not to reveal the workings of attacks not known to competitors. Also, by keeping the rule-base confidential, software vendors could include detection of novel attacks (i.e attacks not yet known to the hacker community) without the risk of someone reverse-engineering the policy for the purpose of writing malicious code to exploit the vulnerability.

2.3 Related work on software protection and fault tolerance

The problem of protecting policies is related to the problem of protecting software/code from the influence of malicious environments. Different types of attacks can threaten software applications, such as *piracy*, where an application is copied and executed in a non authorized environment. *Software tampering* deals with the issue of protecting an application against unauthorized manipulation. Many software developers also need to prevent *reverse engineering* of a program to protect a valuable piece of code or data structure from disclosure. The issue of protecting security policies against unauthorized disclosure is closely related to the reverse engineering problem. Several different techniques have been proposed to tackle the reverse engineering problem and an overview is given in [CT00]. Attempts to use encryption to hide the execution [HP87] is cumbersome as the encryption/decryption process needs to take place in hardware, which limits the applicability. Encrypted evaluation of functions was demonstrated by Sander and Tschudin [ST98a][ST98b]. They demonstrated how polynomial functions could be computed using a solution implemented as software only in a hostile environment. The limitation of polynomial functions was later extended to all functions computable by circuits of logarithmic depth [SYY99]. Obscuring the code through *obfuscation* to prevent attack is another means of protection. Program obfuscation is a semantics-preserving transformation technique, which aims to make the target application ”unreadable”. Extensive research has been conducted in this area and an overview is given in [CT00]. Later, Barak et al. proved the impossibility of achieving program obfuscation under certain conditions. They showed that a *software virtual black box generator* that can prevent reverse engineering of program code, except from that of its input and output, does not exist [BGI$^+$01]. In practice, this is of less concern as applications for practical obfuscation schemes still would be useful in many situations and under less strict conditions [vO03].

In the MAFTIA project, fault-tolerant techniques were used to build dependable systems. They showed how an intrusion-tolerant intrusion detection system could be designed that could provide a secure service, despite the presence of malicious faults [(Ed02]. However, their work was focused on the system and architectural level rather than on mechanisms and protocols.

Research in the field of secure multi-party computation has addressed related problems. Canetti and Halevi [CH98] showed how to maintain authenticated communication even in the presence of break-ins. Their method allows a certain degree of resilience against attackers to protect the *integrity* of authenticated communication. Goldreich et.al. [GMW87] demonstrated how ”mental games” could be played between parties without leaking any information about the actual computation. Neither of these techniques address the same problems presented in this paper, although they all share the same goal to keep computations and information private.

3 Techniques for Policy Protection

3.1 Protecting a security policy using one-way functions

Informally, an IDS uses its detection policy to classify incoming events into predefined categories such as *legal events*, *intrusion attempts*, *intrusions* etc. In its simplest form a security policy states what patterns or signatures of events are authorized / unauthorized. A *default accept* policy would search for events having a certain signature and classify them as unauthorized whereas a *default deny* policy makes the assumption that all events are unauthorized except those that explicitly matches a defined signature. Most rule-based IDSs have taken a default accept standpoint due to the difficulty of defining all authorized events, while most firewalls have a default deny policy, only letting through the events matching the policy.

The following simple example shows how signatures can be used to detect policy violations in a default accept policy. Consider a finite set of input events, $x_1, x_2, \ldots, x_n \in X$, all of which are represented by k-bit binary strings. Further, the set $u_1, u_2, \ldots, u_m \in U$ is the set of ”signature strings” that identify an unauthorized event. Whenever, $x_i = u_j$, where, $i \leq n, j \leq m$, the event being analyzed matches a detection signature and an alarm is raised. The detection scheme is fairly simple as it only involves comparing events over X with all strings in U searching for identical pairs. Now consider $u'_1, u'_2, \ldots, u'_m = f(u_1), f(u_2), \ldots, f(u_m) \in U'$, where f is any cryptographically strong one-way function. Because of the inherent properties of the one-way function, it is computationally infeasible to find any $u \in U$ given $f(u) \in U'$. Thus, U' can be made publicly available without compromising the secrecy of U. The computational effort to successfully deduce $u \in U$ is equal on average to an exhaustive search of $1/2$ of the domain of U. Thus, assuming $u_1, u_2, \ldots, u_m \in U$, where $|u_i|$ has a binary length of k bits, it would (in average) require 2^{k-1} operations.

Detecting policy violations in the default accept case is a straightforward process of applying the same one-way function to all input events, $x_i \in X$, and compare the resulting values to the stored values of U. If a match is found, the input event is an unauthorized event and an alarm is raised. Many intrusion detection systems use simple string matching to find unauthorized patterns in the input data. For example, an IDS could search a UNIX syslog looking for strings containing the pattern ”su root failed” which would indicate a failed attempt to gain administrative privileges on the system. If one-way functions are used to hide the string patterns, it is very hard for an intruder to identify the detection policy of the system.

3.2 Protecting Stateless Security Policies Using One-Way Functions

In [KHJ03], a simple and straightforward scheme for protecting stateless policies is presented. A stateless policy is one that only takes the current event, and not the preceding chain of events, into consideration. The scheme has certain shortcomings. Foremost of these is the fact that normal one-way function schemes can only be used on constant values, so that even small variations in input values give totally different output values. This is a desirable property for the ordinary use of one-way functions, but less desirable for attack signatures. The paper suggested a clustering method based on fuzzy commitment [JW99] that would allow variability in the input data, even if with some restrictions.

4 Protecting Stateful Policies

In this paper, we deploy deterministic finite automatas (DFA) to describe stateful policies. Informally, stateful policies can deal with temporal sequences of events in addition to the occurrence of a single event. Finite automatas (e.g. Finite-state machines) have been used in some IDS [IKP95][Ilg93][Ko96], so the idea of using them for intrusion detection is not new. However, in order to be useful in the context of secure policies they need to be protected from reverse engineering (i.e. to prevent disclosure of the detection policy).

4.1 The deterministic finite state machine

Formally a deterministic finite-state machine M can be described as a five-tuple $M = (\Sigma, Q, q_0, \delta, A)$. In this case $\Sigma = \{x_0, x_1, \ldots, x_n\}$ is the alphabet of the machine or the set of possible input events to the machine, $Q = \{q_0, q_1, \ldots, q_m\}$ is the set of states, $q_0 \in Q$ is the start state, $\delta : Q \times E \rightarrow Q$ is the transition function and $A \subseteq Q$ is the set of final states (or accepting states).

The transition function δ can be realized as a $|Q| \times |\Sigma|$ matrix. In this matrix every state in Q is associated with a column and every event in Σ is associated with a row. The intersection between a row and a column holds the value of the new state to transfer to should this event happen in this state.

The DFA is not a general computation device, i.e. it does not have the computational power of a Turing machine. However, despite its restricted capacity, it can be shown [Mar91] that the languages that can be recognized by a DFA are precisely the same as regular languages. Regular languages are those that can be expressed using regular expressions.

4.2 Protecting finite-state machines using one-way functions

In the following we will describe a protection scheme for a finite-state machine that in effect hides one or more valid finite-state machines within a large set of invalid finite-state machines. We will also describe a method for traversing this large set of finite-state machines without knowing if a valid machine is traversed until a predetermined number of states or an accepting state has been reached. Basically, the machine M will execute under encryption giving no intermediate indication of its state.

In our scheme there is a valid transition for every combination of events and states. This means in effect that any traversal of the protected finite-state machine will either go on forever or stop when an accepting state is reached. The idea of a never ending traversal may work in theory, but is in practice, an impossible traversal method (this will be apparent in section 4.4). Therefore, a traversal that has not reached an accepting state or some other predetermined state within predetermined time will be terminated. In the following we will call these predetermined states synchronization states (ST) and the time limit will be counted in number of states traversed and denoted by L.

In order to protect a deterministic finite-state machine we propose the following:

Assume a one-way function $f : \{0,1\}^u \rightarrow \{0,1\}^v$ which maps bit strings of length u to bit strings of length v and a deterministic finite-state machine $M = (\Sigma, Q, q_0, \delta, A)$. Note that Σ contains all possible events of the type we are looking for and not just the specific events that will lead to an accept state. Thus, if we are looking for specific text strings then Σ would be all the alphanumeric characters.

Define $f(\Sigma) = \{f(x_0), f(x_1), \ldots, f(x_n)\}$.
Create $C(M)$, the obfuscated equivalent of M, by the following procedure:

1. Given a sufficiently large integer k and $n = 2^k$, define a new set $Q' = \{q'_0, \ldots, q'_{n-1}\}$. This set defines the state space in which M will traverse (i.e. the number of possible states). It is important that the size of the set is a power of two in order for the xor operation to work correctly during traversal. The elements of Q could easily be represented by bit strings of size $\log_2(n)$.
2. Choose L in such a way that it is at least as large as the number of states in the longest path in M.
3. Traverse every simple path in M and choose synchronization states on this path in such a way that there is a maximum of L states between two consecutive synchronization states. Make every synchronization state a member of the set S. The number of synchronization states should be kept at a minimum to make an exhaustive search harder. If L is as large as the longest simple path in M between q_0 and any $a \in A$ then no synchronization states are needed. Synchronization states give indications whether the traversal is ”on track” and should continue or be terminated.
4. Define the set $Q_g = \{q'_i | \forall i, 0 < i < n, \exists j \text{ s.t. } q'_i = q_j \text{ where } q_j \in Q\}$ by randomly relabelling and uniquely mapping the states in Q in such a way that $Q_g \subseteq Q'$. Use the same mapping on $A \subseteq Q$ and $S \subseteq Q$ yielding $A' \subseteq Q'$ and $S' \subseteq Q'$.
5. Define the recursive sum $e_{q'_j} = f(e_{q'_{j-1}} || f(x_j)), e_{q'_0} = 0$. Thus, $e_{q'_j}$ is recursively calculated from all events leading to state q'_j.
6. Calculate the state-matrix $K[f(\Sigma), Q']$ where:

$$\begin{cases} k_{f(x_m),q'_j} = \begin{cases} f(q'_j||e_{q'_j}||f(x_m))\ mod\ |Q'| \bigoplus q'_i \text{ for } q'_j, q'i \in Q_g, x_m \in \Sigma \\ \text{if } (q'_j \times x_m) \rightarrow q'_i \in \delta \end{cases} \\ k_{f(x_m),q'_j} = q'_r \text{ if } (q'_j \times x_m) \rightarrow q'_i \notin \delta, \text{ where } q'_r \\ \qquad \text{is a randomly chosen element of } Q' \end{cases}$$

In this equation q'_j is the current state in M, x_m is the current event and q'_i is the next state in M given x_m and q'_j. $e_{q'_j}$ is the recursive sum of all events leading to state q'_j. The modulo operator prevents traversal to state values beyond the range of the elements of Q'. Each element of the matrix are bit strings of size $\log_2(n)$.

The result of the steps above defines the six-tuple $C(M) = (f(\Sigma), Q', q'_0, \delta_Q, A', S')$, the protected (obfuscated) finite-state machine of M. $C(M)$ is a state machine that is hard to (correctly) traverse, backwards or forward, without knowledge of the correct x_m for each transition.

4.3 Analysis of the protection scheme

If q'_0 is the start state then the probability of finding one simple path from q'_0 to $a \in A$ or $s \in S$ by guessing is, $\prod_{i=1}^{L} \frac{o_i}{|f(\Sigma)|}$, where o_i is the number of valid transitions out of the i:th state. Literally, this is the probability of finding *any* path leading to an accept or synchronizing state. If the number of states n in the shortest simple path from q'_0 to $a \in A$ is $n < L$ then the probability p of finding that particular path is $\prod_{i=1}^{n} \frac{1}{|f(\Sigma)|}$.

It is important that the parameters used in our schemes are chosen wisely. The domain of possible events must be sufficiently large to prevent an exhaustive search. In addition, the state space of $C(M)$ must be sufficiently large to further limit the feasibility of an exhaustive search.

If we assume an alphabet Σ consisting of 16-bit binary strings, then $|\Sigma| = 2^{16}$. Further, if the path from start to accept requires five steps, then the probability of finding a unique path is $\prod_{i=1}^{5} (\frac{1}{2^{16}})^5 \approx 0.8 \times 10^{-24}$. i.e. an average of 6×10^{23} attempts before an accept is found.

The number of elements in Q' are also important. The probability of a false alarm is proportional to the number of elements as it defines the state-space in which the machine executes. A machine running havoc in state-space will traverse to any element of Q' with equal likelihood due to the random properties of the one-way function. Thus, given the maximum path length, the number of accept states and the size of the state-space, the probability of false accept is $\frac{L|A|}{|Q'|}$.

4.4 Traversal of the protected finite-state machine

In the following we will describe a traversal method that could be used in order to use the machine for detection purposes. In order to do so we will assume that every traversal of $C(M)$ has associated with it an ordered set V. The elements of V are two-tuples of previously visited states (q'_i) and their corresponding recursive sum ($e_{q'_i}$). The set is chronologically ordered and can hold a maximum of L elements. At the beginning of a new traversal V is emptied. A traversal will continue until it is terminated or reaches an accept state.

The pseudo code for traversal is straightforward:

Let q'_0 be the start state, q'_j be the current state and q'_i be the next state of $C(M)$. x_m is the current input event. All other variables and equations used are defined as above or in section 4.2.

1. Let $q'_i = f(q'_j||e_{q'_j}||f(x_m))mod|Q'| \bigoplus k_{f(x_m),q'_j}$. This is the transition function that determines the next state on input x_m.
2. If $q'_i \in A'$ then raise an alarm and terminate traversal. An accept state has been reached.
3. If $q'_i \in S'$ then $V = \emptyset$.
4. If q'_i is in a tuple of V, the state was previously visited and we delete all the states in V visited after the first occurrence of the tuple containing q'_i from V else add $(q'_i, e_{q'_i})$ to V.
5. Repeat 1-4 until V is full.
6. Terminate.

In general this scheme will have L simultaneous traversals executing, each associated with an ordered set V. Thus L will control the number of simultaneous traversals, Q' and $f(\Sigma)$ will control the memory space needed for the state matrix.

4.5 A simple example

To illustrate the detection of unauthorized events using a protected finite-state machine, a simple example is given. The example does not aim to illustrate any meaningful policy. It simply focuses on showing the principle of the protection scheme.

4.5.1 Finding a substring

Suppose our intrusion detection system analyses the keystrokes from a user, looking for certain substrings. The set of all possible keystrokes is our alphabet Σ. Let us assume that there are $2^3 = 8$ possible keystrokes, thus the input events can be considered as bit strings of length 3 and hence $\Sigma = \{A, B, \ldots, H\}$, $|\Sigma| = 8$. Each keystroke is regarded as an independent event $e_i \in E$, where E is the set of all keystrokes generated by a user over some period of time. Note that the input alphabet does not need to be defined as single equal sized characters or bit-strings. An alphabet could potentially be defined by the set of all possible log strings generated by an application or service such as the log strings generated by SSH in Table 1, on page 185. In this example, the finite state machine M, seeks to detect *sequences of events* containing the substring "ABBA". To be more formal, the machine M recognizes strings of the language $L(M) = \{x \in \Sigma^* |\text{ABBA is a substring of x}\}$. Figure 1 shows a finite state machine recognizing strings in L. M has five states including a start- and an accept state. Thus, $Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$, $S = \{q_0\}$ and $A = \{q_4\}$. The unspecified transitions represent state changes caused by input other that explicitly specified. Our machine M will not traverse these as specified but will traverse randomly into the state space for all input not explicitly specified. This results in a side-effect that we need to start a new traversal on every input event. In this example, this would require four machines running simultaneously. Once M finds the substring, the traversal is aborted (in q_4).

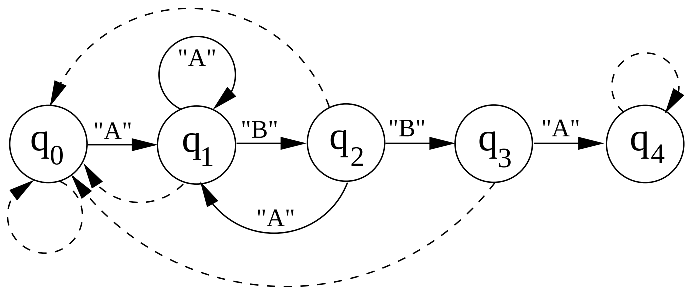

Figure 1: A finite-state machine recognizing strings containing the substring "ABBA".

4.5.2 Protection of M

In this section we will protect M by applying the procedure as described in section 4.2:

A hash function f is chosen. We selected MD5 for that purpose but used only the high 8-bits (out of 128) as output to increase the readability of the example.

1. M has five states. For simplicity we choose a small state space of Q'. Let $n = 16$. The states of Q' are encoded using integer values $0, \ldots, 15$.

2. The longest path of M has a length of five ($|P_{q_4}| = 5$). Since the length is small, we choose L to be equal to the longest path of M i.e. $L = 5$.

3. Since $L \geq |P_{q_4}$, no synchronization states are needed.

4. The states of Q are randomly mapped to elements of Q'. The following mapping is used:

Table 2: Mapping of Q onto Q'

State values of Q':	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Q:		q_1	q_0		q_3				q_2				q_4			

From Table 2 we can see that the state value for $q_4 \in A$ is 12 and the start state value is 2.

5. The state-matrix $K[f(\Sigma), Q']$ is calculated and illustrated below (Table 3).

Table 3: The state-matrix $K[f(\Sigma), Q']$ for M

	State values of Q'															
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
$f(F) = 18$	8	11	1	4	9	2	13	5	3	10	14	1	6	7	2	6
$f(A) = 41$	0	***7***	***14***	8	***7***	9	11	14	***11***	6	9	12	1	8	4	9
$f(C) = 87$	5	14	10	8	2	2	3	6	14	0	7	1	10	4	1	2
$f(B) = 113$	2	***13***	15	0	8	1	14	7	***6***	6	10	4	13	15	4	12
$f(H) = 150$	8	4	10	5	12	5	12	12	11	1	3	11	6	4	1	10
$f(D) = 181$	7	8	2	4	10	6	10	13	5	15	3	4	2	10	11	5
$f(G) = 191$	4	2	10	9	5	7	12	13	7	14	5	11	14	11	13	3
$f(E) = 218$	10	5	13	2	6	9	0	7	11	14	15	3	12	4	1	7

M has six valid transitions resulting in the following equations:

$e_{q_0'} = 0$

$k_{f(A),q_0'} = f(q_0'||e_{q_0'}||f(A))\, mod\, 16 \bigoplus q_1' = f(2||0||f(A))\, mod\, 16 \bigoplus 1 =$ ***14***

$e_{q_1'} = 41$

$k_{f(B),q_1'} = f(q_1'||e_{q_1'}||f(B))\, mod\, 16 \bigoplus q_2' = f(1||41||f(B))\, mod\, 16 \bigoplus 8 =$ ***13***

$e_{q_2'} = 114$

$k_{f(A),q_1'} = f(q_1'||e_{q_1'}||f(A))\, mod\, 16 \bigoplus q_1' = f(1||41||f(A))\, mod\, 16 \bigoplus 1 =$ ***7***

$k_{f(B),q_2'} = f(q_2'||e_{q_2'}||f(B))\, mod\, 16 \bigoplus q_3' = f(8||114||f(B))\, mod\, 16 \bigoplus 4 =$ ***6***

$e_{q_3'} = 47$

$k_{f(A),q_2'} = f(q_2'||e_{q_2'}||f(A))\, mod\, 16 \bigoplus q_1' = f(8||114||f(A))\, mod\, 16 \bigoplus 1 =$ ***11***

$k_{f(A),q_3'} = f(q_3'||e_{q_3'}||f(A))\, mod\, 16 \bigoplus q_4' = f(4||47||f(A))\, mod\, 16 \bigoplus 12 =$ ***7***

The k values for the valid transitions are highlighted in Table 3 for illustration. Those are the only values that will result in a valid traversal leading towards accept. All other values are chosen randomly from Q'. The result of the steps above is $C(M) = (f(\Sigma), Q', q_0', \delta_Q, A', S')$.

4.5.3 Traversing $C(M)$

Traversing $C(M)$ is straightforward. Assume the following sequence of input events: $\{A, B, A, B, B, A\}$. It ends in ”ABBA” which should result in M accepting the string. q_0' is the start state.

The transition function for finding the next state,

$$q'_i = f(q'_j||e_{q'_j}||f(x_m))\ mod\ |Q'| \bigoplus k_{f(x_m),q'_j}$$

(see section 4.4) yield the following six steps (one step for each input):

$$q'_i = f(q'_0||e_{q'_0}||f(A))\,mod\,16 \bigoplus k_{f(A),q'_0} = f(2||0||f(A))\,mod\,16 \bigoplus 14 = 1\,(q'_1)$$

$$q'_i = f(q'_1||e_{q'_1}||f(B))\,mod\,16 \bigoplus k_{f(B),q'_1} = f(1||41||f(B))\,mod\,16 \bigoplus 13 = 8\,(q'_2)$$

$$q'_i = f(q'_2||e_{q'_2}||f(A))\,mod\,16 \bigoplus k_{f(A),q'_2} = f(8||114||f(A))\,mod\,16 \bigoplus 11 = 1\,(q'_1)$$

$$q'_i = f(q'_1||e_{q'_1}||f(B))\,mod\,16 \bigoplus k_{f(B),q'_1} = f(1||41||f(B))\,mod\,16 \bigoplus 13 = 8\,(q'_2)$$

$$q'_i = f(q'_2||e_{q'_2}||f(B))\,mod\,16 \bigoplus k_{f(B),q'_2} = f(8||114||f(B))\,mod\,16 \bigoplus 6 = 4\,(q'_3)$$

$$q'_i = f(q'_3||e_{q'_3}||f(A))\,mod\,16 \bigoplus k_{f(A),q'_3} = f(4||47||f(A))\,mod\,16 \bigoplus 7 = 12\,(q'_4 \in A')$$

(Accept)

Hence, M accepts the string and halts.

4.6 Drawbacks

There are some drawbacks with the scheme that we have identified.

1. The matrix K needs to be stored. This would require substantial amounts of memory (several MB) depending on the size of the state machine but also on length of the input alphabet. An alternative idea would be to represent K as a pseudo-random function but we have not further investigated that possibility.

2. The maximum path length L, leaks information about the policy. An attacker might be able to draw conclusions about the complexity of the policy based on the knowledge of its maximum size.

3. Using our current state-machine model, the policies are limited to those equal to regular languages.

4. Performance is likely to be an issue. Iterative invocations of one-way functions will perform considerably worse than traditional detection schemes. However, the purpose of our work was to explore the *possibility* to protect confidentiality of policies, without paying too much attention to the computational overhead. If one were to deploy our method in practice, a combination of detection techniques can be applied to find a balance between performance and confidentiality.

5 Conclusion and future work

This paper aims to show the applicability of one-way functions for protecting stateful policies used in security components such as intrusion detection systems and firewalls. Specifically, we present a method for preventing reverse-engineering of finite-state machines capable of expressing policies equal to strings corresponding to regular languages. Hence it allows an IDS to maintain confidentiality of its policy even if an intruder manages to study the representation of the policy. Lacking the knowledge of the sequence of input events leading to an accept state, the intruder must perform an exhaustive search of all possible input to find the detection policy. We give a brief complexity analysis of a system having a reasonably sized alphabet which gives support for our claim that it would be computationally hard for an attacker to extract the policy.

Our scheme's ability to limit false alarms is dependent on the state-space of the state machine as hash collisions can occur. Future improvements could redefine an accept state to include some history of input data. This would lower the probability of a false alarm due to a limited state-space. Another means to increase the size of the state-space matrix would be to utilize pseudo-random functions. However, we have not explored this idea further.

Finally, exploring the possibilities of applying our scheme to more generalized computing devices such as those equal to Turing machines would be useful. This would allow us to define policies having a complexity beyond those of regular languages.

References

[BGI$^+$01] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (Im)possibility of Obfuscating Programs. *Lecture Notes in Computer Science*, 2139, 2001.

[BSNP96] S. Bakhtiari, R. Safavi-Naini, and J. Pieprzyk. On Selectable Collisionful Hash Functions. In *Proceedings of the Australian Conference on Information Security and Privacy*, pages 287–292. LNCS No. 1172, 1996.

[CH98] R. Canetti and S. Halevi. Maintaining Authenticated Communication in the Presence of Break-ins. *Journal of Cryptology: the journal of the international Association for Cryptologic Research*, 1998.

[CT00] Christian Collberg and Clack Thomborson. Watermarking, Tamper-Proofing, and Obfuscation - Tools for Software Protection. 2000.

[(Ed02] M. Dacier (Editor). Design of an Intrusion-Tolerant Intrusion Detection System, Deliverable D10. MAFTIA European Project IST-1999-11583, IBM Zurich Research Laboratory, 2002.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play ANY mental game. In *Proceedings of the 19th Annual ACM conference on Theory of computing*, pages 218–229. ACM Press, 1987.

[HKJ99] H. Hedbom, H. Kvarnström, and E. Jonsson. Security Implications of Distributed Intrusion Detection Architectures. In *Proceedings of the 4th Nordic Workshop on Secure IT systems - Nordsec 99*, pages 225–243, Stockholm, Sweden, 1999.

[HLJ99] Hans Hedbom, Stefan Lindskog, and Erland Jonsson. Risks and Dangers of Security Extensions. In *Proceedings of the Security and Control of IT in society-II (IFIP SCITS-II)*, pages 231–248, Bratislava, Slovakia, June 15-16 1999.

[HP87] Amir Herzberg and Shlomit S. Pinter. Public protection of software. *ACM Transactions on Computer Systems, 5(4)*, pages 371–393, 1987.

[IKP95] K. Ilgun, R. A. Kemmerer, and P. A. Porras. State transition analysis: A rule-based intrusion detection approach. *IEEE Transaction on Software Engineering*, 21(3):181–199, March 1995.

[Ilg93] K. Ilgun. USTAT: A real-time intrusion detection system for UNIX. In *Proceedings of the 1993 IEEE Symposium on Security and Privacy*, pages 16–38. IEEE Computer Society Press, 1993.

[JW99] A. Juels and M. Wattenberg. A Fuzzy Commitment Scheme. In *Proceedings of the Second ACM Conferens on Computer and Communication Security CCS'99*, Singapore, 1999.

[KHJ03] H. Kvarnström, H. Hedbom, and E. Jonsson. Protecting Security Policies in Ubiquitous Environments Using One-Way Functions. In *Proceedings of the 1st International Conference on Security in Pervasive Computing*, volume LNCS 2802, pages 71–85, 2003.

[Ko96] C. Ko. *Execution Monitoring of Security-critical Programs in a distributed System: A Specification-based Approach*. PhD thesis, Department of Computer Science, University of California at Davis, USA, 1996.

[Lab95] SRI Computer Science Laboratory. Next-generation Intrusion Detection Expert System (NIDES) - A Summary. Technical report, 1995.

[Mar91] J.M. Martin. *Introduction to languages and the theory of computation*. McGraw-Hill, Inc., 1991.

[Neu95] P. G. Neumann. Architectures and formal representations for secure systems. Technical report, Final Report; SRI Project 6401; Deliverable A002, 1995.

[Nor99] S. Northcutt. *Network Intrusion Detection : An Analyst's Handbook*. New Riders, 1999.

[PN98] Thomas H. Ptacek and Timothy N. Newsham. Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection. Technical report, Suite 330, 1201 5th Street S.W, Calgary, Alberta, Canada, T2R-0Y6, 1998.

[ST98a] Tomas Sander and Christian F. Tschudin. On Software Protection via Function Hiding. *Lecture Notes in Computer Science*, 1525:111–123, 1998.

[ST98b] Tomas Sander and Christian F. Tschudin. Protecting Mobile Agents Against Malicious Hosts. *Lecture Notes in Computer Science*, 1419:44–??, 1998.

[SvS98] A. Shamir and Nico van Someren. Playing hide and seek with stored keys, 1998. Weizmann Institute of Science, Israel; nCipher Corporation Limited, England.

[SYY99] Tomas Sander, Adam Young, and Moti Yung. Non-Interactive CryptoComputing For NC 1. In *IEEE Symposium on Foundations of Computer Science*, pages 554–567, 1999.

[vO03] P. C. van Oorschot. Revisiting Software Protection. *6th International Conference on Information Security*, 2003.

[VRKK03] G. Vigna, W. Robertson, Vishal Kher, and R.A. Kemmerer. A Stateful Intrusion Detection System for World-Wide Web Servers. In *Proceedings of the 19th Annual Computer Security Applications Conference, Las Vegas, Nevada, USA*, pages 34–43. IEEE Computer Press, December 8-12 2003.