Securing Nomads: The Case for Quarantine, Examination, and Decontamination

Kevin Eustice†
kfe@cs.ucla.edu

Dr. Leonard Kleinrock
lk@cs.ucla.edu

Shane Markstrum
smarkstr@cs.ucla.edu

Dr. Gerald Popek
popek@cs.ucla.edu

V. Ramakrishna
vrama@cs.ucla.edu

Dr. Peter Reiher
reiher@cs.ucla.edu

Laboratory for Advanced Systems Research
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90095

ABSTRACT

The rapid growth and increasing pervasiveness of wireless networks raises serious security concerns. Client devices will migrate between numerous diverse wireless environments, bringing with them software vulnerabilities and possibly malicious code. Techniques are needed to protect wireless client devices and the next generation wireless infrastructure. We propose QED, a new security model for wireless networks that enables wireless environments to quarantine devices and then analyze and potentially update or "decontaminate" client nodes. The QED paradigm is presented here, as well as the design of a practical prototype.

Categories and Subject Descriptors

C.2.0 **[Computer-Communication Networks]**: General – *Security and protection (e.g.firewalls), Data communications*

D.4.6 **[Operating Systems]**: Security and Protection – *Access controls, Invasive software (e.g., viruses, worms, Trojan horses)*

General Terms

Design, Human Factors, Security

Keywords

Decontamination, Examination, Mobile Computing, Nomadic Computing, Pervasive Computing, Quarantine, Security, Ubiquitous Computing, Wireless, Worm

† Kevin Eustice was partly supported by The Aerospace Corporation, El Segundo, CA.

New Security Paradigms Workshop 2003 Ascona Switzerland

1. INTRODUCTION

During the last few years, we have seen an increasing trend towards nomadic computing, or nomadicity. Users migrate with their mobile computers from network to network, accessing a wide array of services and networks. This new emerging paradigm has come about partly as a result of cheap, mobile computers and also due to the recent explosive growth in wireless computing. Unfortunately, systems security has not kept pace with this nomadic trend. As a result, our existing security infrastructure is ill-equipped to deal with emerging threats.

As millions of users migrate between home, office, coffee shop and bookstore, they take with them not only their computer, but also electronic hitchhikers they picked up in elsewhere. Continual migration from one access point to another with vulnerabilities threatens the integrity of the other environments, as well as that of other peers within the environments. A user may unwittingly bring in active threats such as viruses, worms, denial-of-service daemons, or even create a hole for a human intruder; alternately they may bring in more subtle threats such as vulnerable packages or poorly configured software. We must mitigate the impact and spread of these attack vectors.

Unfortunately, existing security paradigms do not address this problem. Current wireless security research and proposed standards [3, 2] seek to add better security—however the approaches are primarily focused on better authentication, routing, and stronger encryption. Such improvements are extremely desirable, but they do not address the core integrity issue. Authenticated but corrupted devices could still gain access to network resources and infect other networked devices.

This problem will only be exacerbated as wireless coverage expands and nomadic behavior becomes more and more common. This is a fundamental security threat that must be addressed. Environments must be able to quarantine potential clients, examine and evaluate clients for potential threats or vulnerabilities, and if desired, provide facilities to assist users with securing or cleaning their machine.

This paper proposes a new paradigm that we refer to as *QED—quarantine, examination,* and *decontamination*—to deal with these integrity concerns. We are currently designing and building a QED prototype in our laboratory at UCLA. Further adoption of these techniques will provide a much needed layer of security to protect mobile computing in the local infrastructure and Internet.

2. MOTIVATION

Wireless networks have been rapidly growing in popularity, both in consumer and commercial arenas. Businesses have adopted wireless technologies as an easy mechanism to keep employees connected wherever they go; others have adopted wireless as a new profitable service provided to the public. These services are being deployed in many different public arenas, and are quickly growing in popularity. To the user, access is as simple as inserting a wireless network card and connecting to the appropriate network. Some networks may require a username and a password or some other registration and payment of a fee to access the service. Once a session is instantiated, typical security measures include authentication and encryption of data.

Residential wireless networks are generally easier to access. Currently, there are thousands of residential access points, most with minimal or no security. Information regarding location, accessibility, network ID and deployed security for a great number of these access points is publicly available on the Internet. For the most part, these networks lack any reasonable access control and are thus extremely vulnerable to anyone who wishes to use them.

There is a more serious issue than simple theft of service. Trusting users place their laptops, PDAs and other Internet-capable devices into these insecure networks expecting unrestricted and safe access to both local network resources and the Internet. Unknown to the users, their machines may also play host to malicious agents acquired accidentally while visiting some other public forum or attached to software of dubious origin. If given full access to the network's resources, these infected users then represent a clear threat to the network in the form of a lurking Trojan horse, a worm, denial-of-service daemon, or a tunnel to an outside attacker or freeloader. Other local devices also make easy targets for further exploitation, and may in turn carry malicious code into other possibly more secure environments. In other words, people may place their exploited machines on your network and behind your firewall, and by doing so expose your machines to crackers or malicious code. Those devices may then unwittingly be taken into other networks; the epidemic spreads in this manner.

Widespread adoption of wireless technologies such as WiFi and Bluetooth exacerbates this problem by greatly increasing the nomadic population and the availability of wireless services. As corrupted machines move from network to network, they will be able to quickly spread offending code to network resources and users. Particularly resourceful worms could use nomadic trends to attack and quickly spread in dense urban centers, without resorting to the Internet. There is substantial evidence that recent worms, such as W32/Blaster-A, have been able to infect numerous networks protected by firewalls through infected users' laptops [9]. In a two week period from September 15th 2003 to September 29th 2003, more than 60+ Blaster and Blaster-variant infected laptops were introduced into the UCLA Computer Science network and attempted to spread the infection. Even more advanced worms may emerge that leverage nomadic trends to target normally disconnected or hard-to-reach installations. These types of worms could be extremely difficult to track or observe, as they would be able to avoid propagation through the Internet core.

Based on this observation, it seems imperative that the local infrastructure be capable of automatically isolating, identifying, and potentially repairing vulnerable and corrupt machines. We believe that a transition must be made to a new paradigm of security that allows active, network-based integrity analysis of client machines with minimal user or administrative overhead. This type of infrastructure would strongly encourage active and timely patching of vulnerable and exploited systems, increasing overall network security. It would benefit users by protecting their systems, as well as keeping them up to date, and benefit local providers by protecting their infrastructure and reducing theft of service. Deployment would also protect the Internet as a whole by slowing the spread of worms, viruses, and dramatically reducing the available population of denial-of-service daemons.

3. RELEVANT TECHNOLOGIES

The security model that we are proposing contains many of the characteristics of virus scanners, firewalls, and intrusion detection systems. In addition to maintaining secure environments, our model enables easy software maintenance and patching. Tools for these are available in one form or the other, but a unified integrity analysis and maintenance model has yet to emerge.

3.1 Virus Scanners

Malicious code such as viruses, Trojan horses and logic bombs pose a serious threat to all computer users. Virus scanners are used to counter this threat. These scanners usually work by matching code with known patterns, or signatures, which are stored in a database. They run continuously in the background, monitoring system activity—especially network traffic and downloaded files such as potentially harmful email attachments—and are updated frequently to handle any new threats that may appear. The major drawbacks to virus scanners are that typically they are signature-based, which limits detection to well-known viruses, and they are usually installed on a per-machine basis, which puts the onus of maintenance and retrieving updates on the user's machine, subject to the user's preferences.

In the QED model, virus scanning can be leveraged as part of the examination phase. Infrastructure-based security managers can keep themselves updated from online sources typically in a much timelier manner than mobile nodes, and then ensue that entering mobile nodes receive updated virus information as a condition of entrance. Benefits can be gained both in security and performance in the face of mobility.

3.2 Firewalls

Firewalls are systems that enforce boundaries between two or more networks. These systems are used primarily to filter out traffic from certain sources or those targeted at certain ports; this filtering is done usually on the basis of information stored in the packet IP header. Typically located at the network-centric entry point of a network, i.e., a gateway, they can also act as proxies for the machines within the network and perform various services on behalf of the local machines, such

as filtering out spam email. One capability in the QED model enables the local infrastructure to restrict outbound traffic to authorized hosts, preventing unauthorized local peers from communicating. This effectively serves as a personal firewall, operating at the device-centric entry points of the network, i.e., wireless access points, and other similar ingress points.

3.3 Intrusion Detection Systems

Intrusion detection systems (IDS) are used to detect attacks on a computer system or network based on traffic patterns, system logs, and periodic system integrity checks. Example systems include the Graph-based Intrusion Detection System [11], Emerald [5], Distributed Intrusion Detection System [10] and AAFID [1]. IDS techniques can also be used to defend against attacks generated by insiders [6]. The range of IDS responses to attacks varies from actively shutting down the attack to sending an alarm to the appropriate authority. The QED paradigm requires some IDS techniques to be used in the examination phase, to dynamically examine and perform integrity analysis of potential clients; additionally, IDS techniques are used to monitor active local clients.

3.4 Update and Patch Management Systems

Many commercial operating systems provide update management software that allows users or administrators to automatically download and apply system updates. For Microsoft Windows, this is done via both service packs and an automatic update tool that alerts users to new updates. Similar services are provided by the Ximian Red Carpet utility for Linux, and other UNIX and UNIX-like systems.

In general, these mechanisms are valuable and useful; however we believe they are insufficient for the quickly approaching wireless world. The current model provides little incentive to users to patch or update their system; additionally, downloading packages can be extremely time-consuming over slow links. The QED model requires users to maintain their software to receive connectivity, as well as offering infrastructure-based assistance with updates, such as locally cached packages.

4. QED: QUARANTINE, EXAMINATION, AND DECONTAMINATION

Devices operating within a shared environment must meet high integrity standards; this implies that mechanisms are needed with which to evaluate and ensure the integrity of all devices entering that environment. While a complete general solution to this problem may not yet be feasible, mitigating engineering approaches can be extremely helpful. Our proposed model increases the security and integrity of the network by providing a framework that allows proactive device examination and evaluation of device security characteristics. Tradeoffs may have to be made between desired privacy and required integrity. In some environments, safety must take precedence over privacy. If users are unwilling to compromise their privacy for this safety, they might choose to forgo interaction with the environment in question, or reveal limited information in exchange for limited access.

The model we are developing protects machines by logically isolating them, examining them for known vulnerabilities or malicious software, and then repairing, or otherwise mitigating, discovered problems. We refer to these processes as quarantine, examination, and decontamination. These processes are not necessarily mutually exclusive, and may overlap.

4.1 Quarantine

The goal of the quarantine stage is to isolate potential clients until it can be determined that they meet the local integrity standards. Ideally, we enforce two types of isolation. First, isolation from the outside world prevents possibly malicious code from spreading; additionally, it protects potentially vulnerable machines from outside attackers. In general, this type of isolation is fairly easy to enforce at the router level by employing routing rules that only forward packets for authorized machines. The second form of desired isolation is local isolation. Separating local peers requires the infrastructure to assign extremely restrictive network settings to clients. Such restrictive settings require that all communications go directly through the security manager. Additionally, well-behaved client software can be instructed to drop all packets not sent through the router. This ensures that cooperative clients can only talk to the router, and are not susceptible to attacks, scans, or probes from local peers. Compromised hosts or malicious users can attempt to configure their own network settings to talk to other devices—however, this communication would be limited to similar rogue machines; well-behaved and non-compromised clients would not participate.

Quarantine is not necessarily unilateral—the device itself can also quarantine the visited network, restricting access to local services according to device policy. In fact, the entire QED model may be applied from the perspective of the device—however we believe that typically there will be a resource asymmetry. Devices will be in need of services (such as connectivity) that are provided by environments, and thus somewhat subservient to the environment.

While quarantine is not a guaranteed protection, we believe that the model of providing an isolated network in which prospective client machines are examined is valuable. As trusted computing architectures such as TCPA [12] become more commonplace, it will be increasingly possible to make strong guarantees regarding machine cooperation in this, and other stages of QED.

4.2 Examination

The examination stage is where clients are analyzed and potential vulnerabilities and contaminants are identified. There are a large number of possible mechanisms that can be used to examine potential clients: traditional virus scanners, package management tools, network scanners, and configuration analysis tools such as SATAN [13].

Once a device enters an environment and is quarantined, it is subjected to analysis by the infrastructure. There are a large set of possible types of analyses that could be performed, included external scans and probes of offered services, internal package analysis, virus scans, or behavior monitoring. For instance, a simple type of examination might determine the versions of installed software and appropriate security patches, verifying checksums and signatures where applicable.

The paradigm can allow both passive and active examinations modes. In the active mode, the device would have to undergo such an examination at the point of entry into a network; in the passive mode, the network might accept a signed certificate

indicating that such an examination had been successfully passed in the last environment the device passed through.

The examination procedure would not have to stop after the wireless device entered the local environment. Using standard intrusion detection techniques, the local infrastructure could continuously examine network traffic to determine if any entity is trying to launch an attack or take over other machines.

4.3 Decontamination

The third stage of the QED process is decontamination. Once a client machine has been examined, and potential vulnerabilities or other problems have been found, the infrastructure can assist the user in updating vulnerable packages or cleaning up viruses or other potentially malicious code. Virus scanners can automatically remove or quarantine detected viruses. Package management tools could automatically apply new security patches, update software/firmware versions, or request that certain services be stopped. In the most extreme case, entire system images might be replaced from stored backups—obviously, this would be something not to be undertaken lightly.

Decontamination could be performed automatically, or in a user-assisted manner; in the latter, if vulnerabilities are found, the user is informed and given explicit instructions to clean up the device. The entities undergoing decontamination typically would remain quarantined until the infrastructure is able to verify that decontamination has completed. The degree of access allowed to the device would vary with the ability of the infrastructure to verify the success of the decontamination.

5. DESIGN OF A QED PROTOTYPE

We are designing and building a sample QED framework to provide secure service for Linux-based laptops and PDAs equipped with 802.11b wireless network cards. The framework is designed to provide wireless service and security updates to several dozen wireless clients. This is very simple QED system that is intended to illustrate the components. The prototype makes limiting assumptions, such as that participating devices are benign, and that there are no rogue QED nodes. This is reasonable within our lab, but in a real world environment, much more care and design work would be needed. The major components of our prototype are described below.

5.1 Quarantine

The quarantine phase of our prototype uses routing restrictions in the security manager, and local firewall rules on participating devices to restrict data flow from unknown and potential malicious devices.

A local Linux-based 802.11 gateway serves as the local security manager, as well as running a DNS and DHCP server. When a wireless device accesses the network, it issues a DHCP request for an address, enclosing its public key. The local DHCP server then hands an IP address to the wireless device. This process can be secured through the use of certificates for valid local DHCP servers.

The device sets up the local network settings as provided by the DHCP server. This includes a local IPtables [4] DENY rule that drops all incoming traffic not originating at the local gateway; this ensures that local devices are unable to initially communicate with each other without routing through the local gateway. Obviously, a malicious client will not drop this traffic, but well-behaved nodes will, providing some protection for themselves.

Meanwhile, the DHCP server has taken the client's public key and done a secure dynamic DNS update to insert the public key in the local DNS database entry associated with the assigned IP address. The client then can initialize an IPsec security association with the wireless gateway using the public key for the gateway. The gateway performs a reverse DNS lookup on the client's IP address and retrieves the client's public key from the local DNS database and uses it to create the security association on its end. A client application on the device then opens a connection to the security manager on the gateway and begins to negotiate for service.

The end result is that each client has established a private and secure link to the local gateway. IPsec-based encryption prevents eavesdropping, and firewall rules in the gateway and well-behaved clients ensure that the outside world and local, well-behaved devices are separated from the local quarantined devices.

5.2 Examination

The examination phase examines the incoming device to identify out-of-date packages, or possible vulnerabilities. There are essentially three subphases of examination: network profiling, package inspection, and scanning for viruses and worms.

Network profiling will be accomplished through the use of nmap [7]. Nmap allows users to examine open ports and available services on a remote host in a fairly nonintrusive manner. This analysis can identify anomalies and system vulnerabilities. For example, if nmap were run and determined that a normally unused port, e.g., TCP port 1337, was open on a scanned host, a flag would be set indicating that the machine had been potentially exploited. This violation can then be noted for clean-up during the decontamination phase. Nmap also provides some basic information about the overall system and software versions which could potentially be used by the package inspector or during the decontamination stage. Nmap can also be used to detect the presence of services that are unnecessary or undesirable in the given environment.

Package inspection is the most difficult phase of examination. The security manager would be required to query the device for package information, but in the absence of trusted architecture, there would be no guarantee that the returned package list was complete and had not been tampered with. However, we can make the assumption that benign devices and benevolent users will not intentionally deceive the infrastructure, while malicious nodes very well may attempt to deceive the infrastructure. We are currently investigating techniques to identify lying nodes by examining ongoing behavior to detect discrepancies. We will definitely use recurring examinations to help us detect possible discrepancies.

When a virus scan is requested, the device will be required to present proof, such as a certificate produced by running a virus scanner, that it has run a virus scan of the system within the last 24 hours, or since the last major virus alert, whichever is shorter. Requiring an immediate virus scan is the more secure option, but will add substantial overhead if required at every transition between networks. We are considering the use of

local trust relationships between access points to help optimize the efficiency of high overhead examinations—this is discussed in more detail below in section 6.

5.3 Decontamination

If vulnerabilities in the client are noted during examination, the local infrastructure will initiate decontamination. The results of the prior nmap examination are used to identify the vulnerable service[s]. If a known compromised or vulnerable application is found to be running, the infrastructure will attempt to update the application.

If the update is unavailable or the user is unwilling to accept the update or restriction, either the user must suspend the application, or other users must be prevented from accessing that service via the local firewall rules. Similarly, if the service is not vulnerable but is not permitted within a given environment, (e.g., a peer-to-peer file sharing application,) the device would be told to deactivate the service. Additionally, this could be enforced by the local router blocking all traffic to or from the service in question.

If device examination reveals that its virus scan is not up to date, ideally a virus scan is performed on the device and any viruses or worms are removed. This may not be feasible due to real-time constraints; it could take minutes to hours to scan a multi-gigabyte disk. Since a user would typically want to use only a few applications, the security manager will send a message to the user indicating that he should have those applications scanned. If the user accepts, the manager performs the necessary scan. This will be done by communicating a signed piece of anti-virus software to the client, which will be authenticated and executed. In theory, it would also be extremely valuable to be able to scan the memory space of running applications, looking for vulnerabilities there as well. It should be possible to leverage existing work in this area [8].

In our prototype, all application information is derived from the local RedHat Package Manager (RPM) database. If there are security alerts for any of the installed packages, the appropriate update must be applied to the vulnerable device. If the necessary updates are cached, they are immediately applied, again by communicating with the user of the client device. As our environment includes custom configurations, users are involved in these updates; automated updates without user input could very well break things, for example, patching a system library could break dependent applications.

6. CHALLENGES

There are several challenges that must be overcome as we explore this paradigm. We have identified three major challenge areas: trust, privacy, and performance.

6.1 Trust

There are substantial trust issues in each of the stages of QED that need to be addressed. Ideally, to be most effective, QED would be able to execute code on visiting devices, and trust the results of any execution. Similarly, the device itself would like to trust that the environment will not attempt to subvert it. However, given current operating systems, this is not yet possible.

We can categorize current operating environments into two major categories—public domains and private domains. A public domain is one in which a device is a transient visitor, lacking any long-term relationship; additionally the network has limited or no authority over visiting devices. Within such environments, QED must rely most heavily on external scans and perpetually maintain a limited form of quarantine to restrict undesired access to local and remote services. Within private domains such as one's home or office, it is possible to mandate much more stringent restrictions on the types of examinations that must be carried out. The environment and the device will have a pre-existing trust relationship, as well as some a priori knowledge of one another. Within such an environment users are likely to be much more willing to allow their devices to undergo thorough examinations. Privacy issues are still an issue however, and will be addressed in the next section.

Our prototype is designed to operate within the private domain of our laboratory. It relies extensively on client participation to successfully accomplish all of its goals. If our assumption that there are no rogue QED nodes does not hold, then malicious or compromised nodes may lie or mislead, nearly undetectably. Despite this limitation, our prototype increases security by requiring that client devices placed on the network be kept up to date, and provides a mechanism for assisting with that process. In the lab, QED is a proactive security measure that helps ensure that our wireless devices are free from vulnerabilities. In general, a similarly deployed infrastructure would help slow the spread of viruses and worms, and reduce the viable population of denial-of-service daemons by helping keep well-behaved machines patched and secure.

Future systems will greatly enhance the capabilities of QED. With a trusted computing architecture such as TCPA in place, it would be possible to strengthen all three phases. By running a verifiable trusted kernel on the device and security manager, both systems could verify the integrity of the other. Additionally, it would be possible to verify the scope of the examination, and the outcome of the examination and decontamination phases. Interoperation with TCPA trusted operating systems is a future piece of research for this project.

6.2 Privacy

Privacy is a second challenge area for QED. There is a fundamental tradeoff here between the ability to examine machines and the privacy desired by the users. A direct relationship exists between the degree of invasiveness of examination and the overall accuracy of the analysis.

Currently, if a device does not wish to be examined, it does not receive network connectivity; that will always be a choice. But it may be possible to offer a limited subset of services, or otherwise degraded service to a device that wishes to expose only limited personal information. We are actively investigating this issue in the context of our own prototype. One possibility that we are looking into is the use of verifiable examination modules that clients can analyze to determine the nature and extent of the desired examination.

6.3 Performance

Performance is a key issue that must be considered in the context of mobile systems. The model will not be adopted if machines with no vulnerabilities spend substantial time offline upon entering a new environment; nor will it be adopted if users suffer unpredictable delays when entering new environments. We believe that examination time is the

principal bottleneck in QED for most devices. A pertinent question is, therefore, how much time can be spent examining the device for out-of-date packages, viruses, or possible malicious code? For devices with no vulnerabilities, we wish to be able to quickly authorize them and get them onto the network. One possible optimization for wide-area deployment is the use of local trust between collaborating wireless access points. For instance, all of the access points in the local bookstore might establish reciprocal relationships allowing another access point in the store to vouch for the status of a given client. This would allow clients to easily move around within an administrative domain, without going through repeated quarantine and examination processes. On the other hand, an increase in size of the network of trust also increases difficulty in revocation, if necessary.

7. CONCLUSION

Future computing environments will allow computing and communications wherever we work, live, and play. We can easily foresee a future in which connectivity is ubiquitous, provided by businesses who gain profit or other benefit by offering such connectivity. But providers will not offer such services if the networks are perpetually corrupted by infected devices and users will not use these services if their devices will be continually attacked and compromised. This vision cannot be fulfilled unless it is safe to provide and safe to use.

QED offers a safety net to users and service providers. The service provider can use the model to ensure that infrastructure is safe from incautious or malicious users. The average user can rest assured that networks employing the paradigm are unlikely to corrupt machines, steal data, or abuse or deny services due to contamination. If QED had been available and deployed on laptops and other computers throughout the Internet in the last few years, there would have been a dramatic reduction in the spread of worms and other malicious code.

We are implementing a sample QED framework that displays the feasibility and promise of our approach. Adding further security services and leveraging the kinds of secure architectures beginning to emerge in the market will allow for more powerful and reliable QED systems in the future. This, in turn, will enable safe use of ubiquitous networking for everyone.

8. REFERENCES

[1] J. Balasubramaniyan, J. Garcia-Fernandez, E. Spafford, D. Zamboni. An Architecture for Intrusion Detection using Autonomous Agents., COAST Technical Report 98/05, 1998.

[2] Extensible Authentication Protocol – RFC 2284 - http://www.ietf.org/internet-drafts/draft-ietf-eap-rfc2284bis-01.txt

[3] Y. Hu, A. Perrig, D. Johnson Ariadne: A secure on-demand routing protocol for ad hoc networks. The 8th ACM International Conference on Mobile Computing and Networking, MobiCom 2002.

[4] Iptables : http://www.netfilter.org/

[5] Peter G. Neumann, Phillip A. Porras, Experience with EMERALD To Date. First USENIX Workshop on Intrusion Detection and Network Monitoring, April 1999.

[6] Nam Nguyen, Peter Reiher, Geoff Kuenning, Detecting Insider Threats by Monitoring System Call Activity. Submitted to 4th Annual IEEE Information Assurance, West Point, New York, Mar 2003.

[7] Nmap Network Mapper. http://www.insecure.org/nmap/

[8] The Open Group's Common Data SecurityArchitecture(CDSA).http://www.opengroup.org/security/l2-cdsa.htm

[9] Paul Roberts. Hackers find way to exploit latest Microsoft hole. IDG News Service, Sept. 16, 2003. http://www.infoworld.com/article/03/09/16/HNhackers_1.html

[10] Steven R. Snapp et al. DIDS (Distributed Intrusion Detection System) - Motivation, Architecture, and An Early Prototype. Proc. 14th National Computer Security Conference. Washington, DC, Oct. 1991, pp. 167176.

[11] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, D. Zerkle. GrIDS - A Graph Based Intrusion Detection System for Large Networks, in Proc. of the 19th National Information Systems Security Conference. Baltimore, MD, Oct. 1996, 361 - 370.

[12] The Trusted Computing Platform Alliance http://www.trustedpc.org

[13] W. Venema, W. and D. Farmer. Improving the Security of Your Site by Breaking Into It. 1993 Internet White paper. http://gd.tuwien.ac.at/infosys/security/wietse-archive/admin-guide-to-cracking.101.Z