WiFi Nomads and their Unprotected Devices: The Case for QED—Quarantine, Examination, and Decontamination

Kevin Eustice, Leonard Kleinrock, Shane Markstrum,
Gerald Popek, V. Ramakrishna, Peter Reiher
{kfe,lk,smarkstr,popek,vrama,reiher}@cs.ucla.edu

UCLA Laboratory for Advanced Systems Research
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.

1. Introduction

The explosive growth in wireless computing has been spurred on by new and cheaper hardware and increasingly better protocols and operating systems support, all accessing the ubiquitous Internet. The recent addition of IEEE 802.16a [802.16a], support for 2-11Ghz Metropolitan Area Networks (MAN), is only the latest in networking standards designed to facilitate widespread adoption of wireless technologies in communities. Additionally, commercial roll-out of wireless access points has commenced with initial deployment at popular locations such as university campuses, coffee shops, bookstores, and fast-food restaurants.

As millions of wireless users migrate between home, office, coffee shop and bookstore, they move from one wireless access point to the next. They take with them not only their computer, but also electronic hitchhikers they may have picked up in the local shopping mall and unpatched or poorly configured applications. Continual migration from one access point to another with these 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, Trojan Horses, denial-of-service daemons, or even create a hole for a human intruder; alternately they may bring in passive threats such as vulnerable packages or poorly configured software. We must mitigate the impact and spread of these attack vectors.

Unfortunately, the existing paradigm for wireless security does not address this problem. Wireless networking has been notorious for poor security implementations [Borisov2001, Arbaugh2001]. Current research and proposed standards [Hu2002, EAP] seek to add better security—however, the approach is primarily better authentication, and stronger encryption. Such improvements are extremely desirable and laudable, 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. Improved authentication and encryption would better ensure the identity of peers and the confidentiality of the data being transmitted over the network, but not be able to handle the hidden threats they may bring into the network.

The integrity and vulnerability problem will only be exacerbated as wireless coverage and participation continues to grow. We believe that 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 sample 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 service to provide to the public, usually for a small fee. 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 virus, 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 wireless network and behind your firewall, and expose your machines to crackers or malicious code; you may then unwittingly take one of those machines to your place of business and spread the epidemic.

Widespread adoption of wireless in the form of WiFi, Bluetooth and other technologies exacerbates this problem by greatly increasing the wireless population and the availability of wireless service. 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.

In a standard wired Internet environment, when a new user plugs his portable computer into a local network, a human system administrator can manually examine the machine and determine if it is sufficiently secure. This manual approach cannot work in the emerging wireless mobile world. Too many machines will move too frequently between too many administrative domains for any realistic number of human system administrators to keep up with them. Thus, an automated approach is required.

Based on this observation, it seems imperative that the local infrastructure be capable of isolating, identifying, and repairing vulnerable and corrupt machines. We believe that a transition must be made to a new paradigm of wireless 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 main drawback to virus scanners is that typically they are signature-based, which limits detection to well-known viruses. However, both Norton and McAfee constantly update their databases on host machines through the Internet.

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. 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 entry/exit point of a network, such as 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 with the outside world. Thus, firewalls help provide the quarantine phase of QED.

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 [Staniford1996], Emerald [Neumann1999], Distributed Intrusion Detection System [Snapp1991] and AAFID [Balasu1998]. IDS techniques can also be used to defend against attacks generated by insiders [Nguyen2003]. 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.

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 public 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 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 obvious privacy implications 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 not to interact 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 cleaning or patching the applications and libraries when appropriate or desired. 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 also protects possibly 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 router. 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.

While quarantine is not a guaranteed protection, we believe that the model of providing an isolated network in which wireless client machines are examined is valid and valuable. As trusted computing architectures such as TCPA [TCPA] 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 [SATAN].

Once a device enters an environment and is quarantined, the software and firmware that it carries can be subjected to analysis by the infrastructure. Analyzing the entire body of code on each individual entity may be infeasible for most devices; however the analysis can be performed on a small subset that is representative of various installed software packages. For instance, a simple type of examination might determine the versions of installed software and appropriate security patches, verifying checksums and signatures where applicable. Additional types of examinations might include either active or passive virus scans. An active scan might require the device to scan for viruses before being allowed entry, while a passive scan may just ask the device for some proof that it has completed a virus scan within a given timeframe.

The examination procedure would not have to stop after the wireless device entered the local environment. Using standard IDS 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 analyzed, and potential vulnerabilities or contaminants 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. Decontamination could be performed both automatically and with help from the user; in the latter, if vulnerabilities are found, the user is informed and given explicit instructions to clean up the device.

The entities undergoing decontamination should be quarantined until the infrastructure is able to verify that they are as clean as is necessary to allow communication to proceed. The degree of interaction allowed varies not only with the success of the decontamination, but also with the amount of unknown software on the devices.

Time is a serious constraint for decontamination. This process must be performed quickly to be feasible in real environments. Caching of software updates and system patches would likely be valuable; related work in web caching and content distribution networks can be leveraged. A profile of the local network could be maintained that indicates what type of devices and what software or applications are most likely to come into play within the local environment at any point of time. Based on this profile, active prefetching of service packs and other necessary software can be done. When the local cache does not have a necessary update, the required software will be retrieved from the Internet, imposing a slight performance penalty.

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, while keeping unknown, vulnerable, or malicious machines quarantined. The major components of our prototype are described below.

5.1 Quarantine

The goal of the quarantine stage is to isolate devices from the outside world, as well as from one another. In practice, the former is fairly easy to accomplish, but the latter is difficult. For our prototype implementation, we use a combination of techniques to achieve this effect.

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. The local DHCP server hands an address to the wireless device, and asks the device for its public key. This process can be secured through the use of pre-deployed certificates for valid local DHCP servers.

The device responds with its public key, and sets up the local network settings as provided by the custom DHCP server. This includes a local IPtables [IPtables] 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 a predeployed public key for the gateway stored on the device. 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 is separated from the local quarantined devices; thus, well-behaved local devices are isolated from malicious local devices.

5.2 Examination

The examination phase uses mostly publicly available software for the Linux platform. There are essentially three subphases of examination: network profiling, package inspection, and virus scanning.

Network profiling will be accomplished through the use of nmap [Nmap]. 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., UDP port 31337, 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. For instance, the local access point can ask nodes not to run a SMTP daemon, and instead use the local mail gateway.

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 well-behaved devices and 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 periodic nmap exams 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, the application will need to be cut off. Either the user must suspend the application, or other users must be prevented from accessing that service via the local firewall rules. An example of this would be *sendmail*, for which security alerts are issued frequently. If there is a local SMTP server, the client does not need to run *sendmail,* and therefore might be required to shut down the local daemon in certain environments. Similarly, other configuration holes (e.g., world-readable and –writable file shares) might require similar intervention.

If device examination reveals that virus checks are not up to date, the best possible method of decontamination would be to run a virus scanner on the entire contents of the device and remove viruses or Trojans, if any are found. 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 acquiesces, 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 our prototype, all applications 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. Users should be fully involved in these updates, as they best understand the contents of their devices and the interdependencies; automated updates without user input could very well break things, for example, patching a system library could break dependent applications. For our prototype, we assume users do understand their applications and the system dependencies fairly well; in most typical real world deployment scenarios this will usually not be true. The downloaded and cached updates are accompanied by MD5 checksums; this is a standard technique for authenticating downloaded code. This technique will not work for applications that have been built from their source, unless source RPMs were utilized. We could perform similar operations to update device firmware, but real-time constraints might prevent us from doing this on a wide scale.

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. In each stage, our prototype relies on client participation to successfully accomplish all of its goals. Without client participation, the system is less effective. The security manager requires the client to willingly partition itself off from other local nodes, execute a local application to provide data to be used in the examination phase, and accept updates or configuration changes in the decontamination phase. Malicious or compromised nodes may lie or mislead in these phases, nearly undetectably. On the other hand, if a malicious device masquerades as the security manager, it would mean disaster for client nodes. This faux manager would be able to hijack any device it chooses and run arbitrary code at any site.

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.

QED does have the potential to do much more. With a trusted computing architecture such as TCPA in place, it would be possible to strengthen all the three phases. In the quarantine stage, the security manager could force clients to only listen to packets that come from the security manager. This would prevent an authorized client from proxying for another device; this is obviously desirable to prevent theft of services in the context of public access points. In the examination phase, it would be ideal to ask clients to run trusted applications and have some assurance that the results are legitimately reported. Trusted architecture can ensure reported material by allowing trusted applications to be run. Similarly, in the decontamination phase, to trust that clients actually apply updates, repair applications, or fix configuration errors would be invaluable.

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. An inverse 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.

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. 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.

The decontamination process will also be a bottleneck for some devices. However we believe that with fairly widespread coverage of QED, most machines will already have been updated when they visit a new area and thus will not need to go through decontamination. As discussed earlier, caching and client profiling could significantly reduce decontamination time.

7. Conclusion

Wireless networking has the power and potential to allow computing and communications in places where we work and visit. We can easily foresee a future in which wireless connectivity exists almost everywhere, 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 clients, and users will not use these services if their devices can be easily compromised. This promising service cannot succeed in the long term unless it is safe to provide and safe to use.

QED offers the necessary new paradigm to allow safe use of widespread wireless service. The service provider can use the concept 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.

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 wireless networking for everyone.

References

[802.16a] IEEE Draft P802.16a - Draft Amendment to IEEE Standard for Local and Metropolitan Area Networks - Part 16: Air Interface for Fixed Broadband Wireless Access Systems - Medium Access Control Modifications and Additional Physical Layer Specifications for 2-11 GHz (amendment to IEEE Std 802.16) - 1 Apr 2003, ISBN # 0-7381-3566-6.

[Arbaugh2001] Arbaugh, W., Shankar, N., and Wan, Y.C. “Your 802.11 wireless network has no clothes.” Technical Report, Dept. of Computer Science, University of Maryland, March 2001.

[Balasu1998] Balasubramaniyan, J., Garcia-Fernandez, J., Spafford, E., Zamboni, D., "An Architecture for Intrusion Detection using Autonomous Agents", COAST Technical Report 98/05, 1998.

[Borisov2001] Borisov, N., Goldberg, I., and Wagner, D. "Intercepting mobile communications: The insecurity of 802.11." in MOBICOM, MOBICOM 2001.

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

[Hu2002] Y. Hu, A. Perrig, and D. Johnson, "Ariadne: A secure on-demand routing protocol for ad hoc networks," in The 8th ACM International Conference on Mobile Computing and Networking, MobiCom 2002.

[IPtables] http://www.netfilter.org/

[Neumann1999] Peter G. Neumann, Phillip A. Porras, "*Experience with EMERALD To Date*" First USENIX Workshop on Intrusion Detection and Network Monitoring, April 1999.

[Nguyen2003] 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.

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

[Shankar2002] Shankar N., Arbaugh W. “On Trust for Ubiquitous Computing.” Workshop on Security in Ubiquitous Computing, UBICOMP 2002, Göteborg Sweden.

[Snapp1991] 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.

[Staniford1996] Staniford-Chen, S.; Cheung, S.; Crawford, R.; Dilger, M.; Frank, J.; Hoagland, J.; Levitt, K.; Wee, C.; Yip, R.; Zerkle, D.: 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.

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

[Venema1993] Venema, W. and Farmer, D. “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