QED: Securing the Mobile Masses

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 growing trend of nomadic and mobile computing raises serious security concerns. Nomadicity has become common; users are in constant transit, migrating with their mobile devices through many diverse wireless environments. Some users are innocent—potentially vulnerable, but benign—while others are malicious or carry compromised devices. The mobile commons brings all of these users and devices together. Both clients and hosting environments must be protected. To that end, we are developing QED, a new security model for computer networks. QED enables environments to quarantine devices, examine them, and potentially update or "decontaminate" client nodes. We present the design of our QED prototype and evaluate its performance in our laboratory.

1. Introduction

In the last decade, the nomadic computer-user has become commonplace. We take computers from our home network to our office network, to school, and to local cafés. The increasing pervasiveness of wireless networks in the past few years has made this type of network migration extremely easy. In fact, many wireless devices are by default configured to access any local network that will answer their request for service. This means it is now extremely simple to acquire connectivity—an obvious boon. Unfortunately, this flexibility and easy access to local networks and the Internet can also be a security nightmare. For example, as users and their mobile devices migrate from their home to office, they take with them vulnerable applications and services and, possibly worse, malicious code.

Existing security mechanisms are incompatible with this degree of device mobility and this scale of dynamism. Typical security mechanisms are unaware of transient members; the traditional focus has been on protecting the "inside" network from the "outside" network, where there are well-defined boundaries. However, these boundaries become indistinct as users freely move devices back and forth with little thought to security.

An immediate consequence of mobility is that the impact of vulnerabilities and malicious code is amplified many times over. Vulnerabilities are exposed to many potential attackers, and malicious worms and viruses have access to many more local networks in which they can spread. The sudden infection and spread of worms, such as Blaster during August 2003, illustrate this problem. Numerous site-wide infections were caused by users bringing their infected laptops into a supposedly secure zone, allowing the worm to freely propagate [Hilley2003].

Unfortunately, this situation will only continue to escalate as more and more mobile devices and services emerge. These new consumer devices and appliances will gain first-class status as networked computers—both wired and wireless. Simultaneously, network services will be offered virtually everywhere, allowing handheld and mobile devices to acquire connectivity wherever they may be taken. Public wireless commons will bring together numerous mutually unknown and potentially dangerous devices.

This emerging trend requires new security solutions to address its unique demands. Hosting networks need to ensure their local integrity, and ensure that new clients adhere to local policy restrictions. The level of protection may differ with the type of environment and relationship between the environment and member devices.

Home and business environments may proactively keep occupant and employee devices up-to-date with the latest security patches and virus definitions. In public areas where intrusive mechanisms are not feasible, QED can mediate interactions and prevent casual infection within the environment.

QED is a new security model designed to protect nomadic users and the networks that host them. In three stages, devices are *quarantined, examined,* and if necessary, *decontaminated* upon entering a new network. The quarantine stage isolates newly joined devices—potential clients—from the entire network, establishing a secure session between the visiting device and a security manager. This serves to protect the incoming device and devices within the network from each other. The device is then examined and evaluated by the security manager, which determines whether the device meets local policy in terms of application and operating system patch level, offered services, or any other specified metric. If the device does not meet local policy, it may be offered restricted service or possibly none at all. Alternatively, the hosting network may offer decontamination assistance, for instance, it might provide appropriate signed software updates or suggest services to disable in order to receive service.

QED is a modular framework, designed to be extensible and support whatever form of examination or decontamination that may be dictated by the network environment's policy. The contribution of this research is an extensible framework which strives to enhance the security and integrity of mobile computers, local infrastructure, and the Internet in as unobtrusive a manner as possible. QED-enabled networks will provide a much needed layer of proactive protection, ensuring that devices are maintained in a timely fashion according to the policy demands of their environment.

We have designed and implemented a sample QED prototype within our laboratory. In this paper we present the implementation details of our prototype, as well a performance evaluation. The time overhead required to join a QED-enabled network is fairly minimal, typically requiring on the order of 6 to 7 seconds upon discovering a new network until becoming a full member of that network. This can be done automatically upon sensing an appropriate network, and may not even be noticed by the user.

While an industry consortium led by Cisco has announced that they intend to tackle some aspects of this problem in the near future [Cisco2003], their exact plans are not yet known, and they have no working systems available. This paper describes work that precedes the announcement of the Cisco effort, and outlines an already-working system that goes beyond the announced industry plans in several significant ways. This will be further discussed in the related work section.

2. Motivation

In the past decade, we have experienced a dramatic shift in the way computers are used. The rapidly shrinking size of personal computers, coupled with ever-improving wireless technology and network autoconfiguration points at a future where most, if not all, consumer devices are network-aware. Our environments will host numerous services geared toward these mobile devices—different environments will need to be able to impose varying policy requirements on member devices.

This vision is rapidly becoming reality. The increasing trend toward nomadicity has already created critical security challenges that must be addressed immediately. Consider the case of Bob, a typical company employee. Bob owns a laptop and is reasonably technically savvy, but not a security expert. He takes his 802.11-enabled laptop to and from work daily. He also frequently interacts with other public wireless networks to check his e-mail and browse the web.

Before work one morning, Bob stops by his local coffee shop to buy a latté and read the news on his laptop. Unfortunately for Bob, the coffee shop is not a particularly secure environment. A new quick-spreading worm is out in the wild, and another patron has brought into the coffee shop a laptop that is infected with such a worm (see Figure 1), e.g., W32.Blaster.Worm [Blaster].

In short order, Bob's laptop is attacked and infected. Oblivious to the infection, Bob takes his laptop with him to work, and connects to his corporate network, behind his company's firewall. Other machines on Bob's local LAN are vulnerable and infected as the worm spreads quickly. If any of the newly infected machines are also mobile, they can be taken out of the office and will continue to spread the infection.

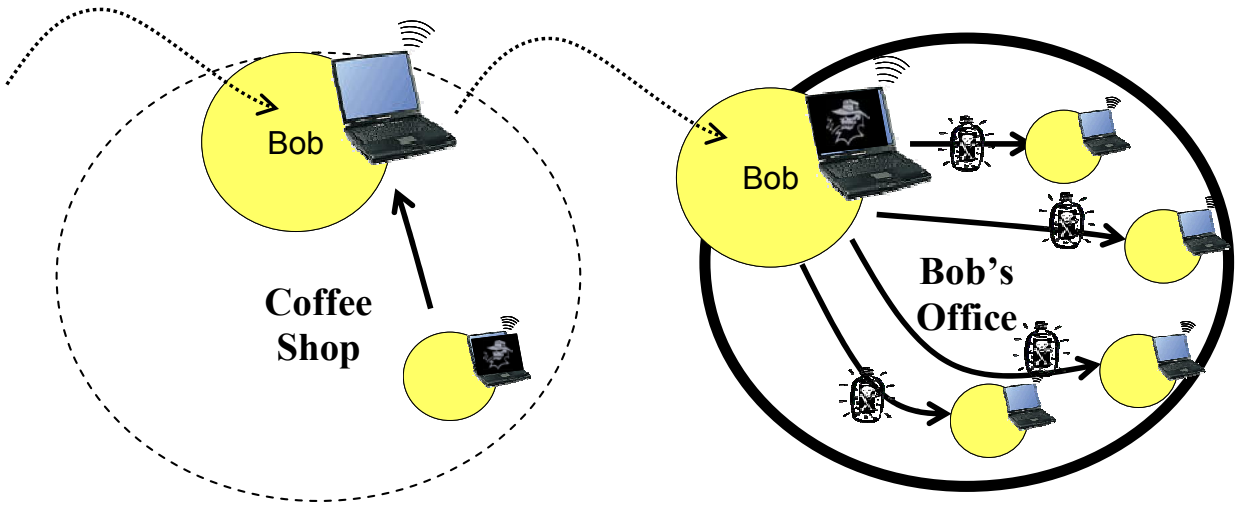

Figure 1. Bob visits his local coffee shop. While there, a compromised device exploits an unpatched vulnerability on Bob's laptop. Bob later goes back to his office, taking his laptop with him into his office, past the company firewall. Bob's laptop proceeds to infect other devices in the office.

This simple scenario illustrates the threat inherent in relying on traditional security mechanisms to deal with nomadic users and devices. Unfortunately, this scenario is far from fiction. Occurrences very similar to this led to the infection and subsequent re-infection of the <*anonymized*> Computer Science Department network during the late summer and fall of 2003. Similar experiences have been common elsewhere.

Currently, tens or hundreds of thousands of residential and commercial wireless networks may be acting as breeding grounds for an unknown number of malicious agents. Clearly, a new security paradigm must be adopted to identify vulnerable and compromised devices, and quarantine them to prevent infection or exploitation. Unless there is a direct need for local peer communication, local devices should be isolated from one another, communicating through a secure gateway. Additionally, networks could provide resources to devices to allow repair and update of software packages, or instead, simply alert users to current problems. Malicious code spreads most commonly by exploiting known software vulnerabilities. Updating software in a timely manner would go a long way in slowing down this spread. As the number of mobile devices quickly grows, proactive maintenance will be absolutely essential.

In a QED system, Bob's experience would be much different, as illustrated in Figure 2. Bob's office is QED-enabled, and all devices go through the QED process when they join the network. When Bob comes to work in the morning, his laptop is quarantined and communication is only allowed with a local security manager. The security manager quickly examines Bob's laptop and determines that he needs the newly released patch. Bob's laptop examines the patch and cryptographically verifies its integrity. The patch is installed, and data is allowed to be routed through the security manager.

Later, when Bob goes to get a coffee and browse the web at his local coffee shop, an exploited laptop attempts to attack Bob's laptop. However, Bob is no longer vulnerable, and the attack fails. Bob's laptop meanwhile, is also aware that the coffee shop is not QED-enabled, and the laptop is by default denying all traffic not

originating from the coffee shop gateway. Additionally, if Bob chooses, his laptop can automatically create an IPsec-based VPN tunnel back to a QED-enabled network and deny all traffic that originates outside the tunnel. Bob could then remotely receive updates from his office, as available.

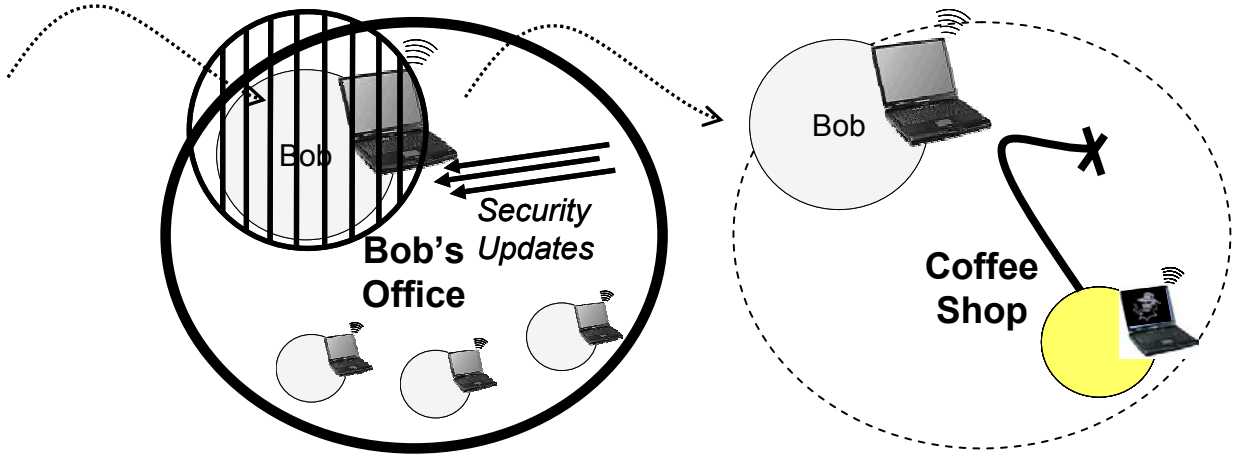

Figure 2. A vulnerability alert and corresponding system patch are issued. At the office, when Bob brings his laptop in, it is automatically quarantined to protect it from local exploitation. The security update is applied to Bob's laptop. Later, when Bob visits a local coffee shop, which is not QED-enabled, Bob's laptop successfully deflects an attack on the just-patched piece of software.

Another possible outcome is that Bob is away from his office for several days. In the meantime, his laptop is compromised by a worm for which his office was unable to proactively patch him. In this case, Bob unknowingly brings the exploit into his office as in Figure 1. However, the office is QED-enabled. When Bob enters, his laptop is quarantined by the office network and prevented from communicating with other machines. The local security manager has the necessary patches and already updated local machines. Examination of Bob's laptop's behavior in quarantine indicates the presence of the new exploit. If possible, suitable anti-worm software is run on Bob's laptop, and a system administrator is notified about the problem. Bob's laptop is denied access to critical resources until it has been automatically decontaminated, or has potentially been deemed safe by an administrator. The exact requirements are network-dependent, and will be partially be dictated by the degree of trust the network places in client devices.

The type of infrastructure described in this scenario is a strong proactive measure designed to encourage active and timely patching of vulnerable systems, increasing overall network security. It will benefit users by protecting their systems from unknown and potentially corrupted devices, as well as keeping these systems up to date, and benefit local providers by protecting their infrastructure.. Deployment will also protect the Internet as a whole by slowing the spread of worms and viruses, and dramatically reducing the available population of denial-of-service daemons.

This scenario explores just one possible use of the infrastructure we are currently building. In its more general form, QED can be used to implement location- or network-specific policy regarding system patches, antiviral updates, allowed services or software, valid operating systems, cryptographic schemes, etc.

3. Relevant Technologies

The security model that we are proposing in this paper moves away from the traditional approach taken to solve these kinds of problems. Our aim is to provide a comprehensive security solution that ensures safety of devices and environments by actively controlling access to networks as well as continual network monitoring. We would not only like to be able to fix security holes as they occur, but also to anticipate such problems and take preventive measures as appropriate. The QED security model contains many characteristics of virus scan-

ners, firewalls, and intrusion detection systems. Each of these tools is specialized to solve one or more security problems, but a unified integrity analysis and maintenance infrastructure has yet to emerge.

Current security frameworks are built in a variety of ways. Some use access-control lists—these tend to be static and not particularly scalable—unsuitable for most mobile computing applications. Others use role-based access control or delegations—both useful techniques that add flexibility, but they are not as useful when dealing with many unfamiliar devices whose demands cannot all be easily classified. The QED framework attempts to extend upon these with idea that some trust and privileges can be earned through compliant behavior and particular actions taken by client devices. With trusted hardware and an appropriate model of trusted client characteristics, this can be a very strong tool for determining device compliance with stated network policy. Without trusted hardware, QED still serves to proactively maintain devices, provide appropriate network-specific configuration, and limit casual exploitation and infection.

A hardware-based intelligent gateway has been developed by Lockwood et al. [Lockwood2003]. Their main concern is to prevent or slow the spread of worms and viruses, such as Code Red, Nimda, Slammer and Blaster, through the Internet. A platform has been implemented that actively scans Internet traffic at multi-Gigabit/second rates using the Field Programmable Port Extender. Programmable hardware (FPGA) based modules scan packet headers for a set of given signatures. Detected viruses can be handled both actively and passively. The FPGA-based logic can be dynamically reconfigured to scan for new signatures.

Though this promises to be a very useful tool, it suffers from the drawback of being hardware-based, and is more difficult to deploy and maintain than a purely software-based system like QED. Though packet scans can be extremely fast, processing every packet could negatively impact normal operation. How effective this system is in reducing virus and worm spread remains to be seen.

The Cisco Self-Defending Network Initiative is a recently announced project that aims at improving the ability of networks to identify, prevent and adapt to security threats. Cisco leads an industry-wide collaboration in this project, which includes major anti-virus software companies such as Symantec. The first program announced under this initiative is the Cisco Network Admission Control (NAC) [Cisco2003], which leverages the network infrastructure to limit damage from viruses and worms. With this framework in place, networks will be able to maintain security policy and control device access; devices that are trusted or comply with the policy will be allowed access and the others will be regulated to quarantined environments with limited or no network access. A device will be considered to be compliant if it has the latest patches as recognized by a network policy server, which then makes the appropriate admission control decision—permit, deny, quarantine, or restrict. Routers enforce access control (prevent non-compliant devices from communicating with the rest of the network) using access control lists.

The Cisco approach resembles one aspect of QED's functionality, with certain differences. The NAC will quarantine and examine devices, but no framework for automatic decontamination is specified. System examination is limited to checking for the latest anti-virus and operating system patches, unlike the open and modular approach we have taken with QED. Few technical details are available as of yet, and only high-level architectural information is available. It is likely that this security framework will be integrated with Cisco's hardware products, whereas QED will be completely software-based, open-source, and not require specialized hardware. Additionally, a working prototype of QED currently exists.

Other related work is ongoing within Fortune 500 IT departments and academic institutions—automatic update facilities are being put into place primarily as a response to the threat posed by recent worms. These can be implemented in the form of login scripts; however details tend to be proprietary. Systems such as these provide one piece of the functionality QED offers—however, QED is much more. QED is an open and extensible framework for implementing security policy in a network and location sensitive manner. A fundamental difference between QED and other security frameworks is QEDs ability to offer customized handling based on policy, device, and individual.

4. The QED Model

The QED model is based on the observation that when a mobile device senses a network, the device and the network must participate in a process to determine if and how the device is going to join a network, and what

the device's role will be in the new network. This must be done as securely and safely as possible, without interference from external entities. This leads to the extrapolation of three distinct components of this process: 1) a quarantine component to protect the device from external interference and to protect existing network denizens from potentially hostile devices, 2) an examination component that allows the device and environment to each determine if the other is acceptable, and 3) a decontamination component that assists with any recommended or required configuration changes. When actualized, the implementation of these components may not be quite so distinct and segmented, e.g., there may be blurring of boundaries between examination and decontamination. However, there is a necessary time ordering—quarantine must begin first, followed by examination, and then followed by decontamination to ensure safety. Each of these components is discussed in detail below.

4.1 Quarantine

Quarantine refers to connection establishment and restricting network communication to a necessary minimum. From a device perspective, this means denying incoming messages from network devices other than an authorized security manager for the network the device is attempting to join. From the network perspective, this means isolating an incoming device from other devices on the local network and the Internet.

Every environment must have a security manager, ideally located at the gateway of the network, where it can enforce two types of isolation. The first type of isolation involves protecting local machines from the outside world. All communication goes through this manager. This type of isolation is provided today to varying degrees by firewalls. The second type of isolation protects local peers within the network from the entering device. This capability is at least as important as isolation from the outside world, and rarely provided by existing security systems.

Quarantine requires establishing a secure channel between the network security manager and the device. This typically will require establishment of trust either directly, via pre-deployed public keys on the mobile device for all networks that the device may visit, or more likely indirectly, via cached certificates of trusted signing authorities. Once the candidate network presents a signed certificate, the device can assess the certificate and determine if the network is one it wishes to visit. Similarly, the network examines the device's credentials and determines if the device should be allowed to continue to join. Device credentials may be signed by a vouching institution, such as one's company, or by a financial institution indicating that the user can and will pay for access.

While this secure channel is being created, the potential client is denying all incoming traffic not associated with the creation of this channel, e.g., if using an IPsec-based VPN with X.509 certificates, we only allow IPsec packets related to the SA creation and the certificates to be exchanged. After the secure channel to the security manager has been created, the client denies all traffic not part of the channel. This prevents attacks from other devices in the network. This is a voluntary procedure; however it is wholly beneficial to cooperative client devices. Uncooperative devices are addressed in section 7.1.

A potential client will remain in quarantine during the entire QED process. It may in fact be desirable to maintain a partial quarantine indefinitely, depending on the degree of trust between client and network. There are many ways in which an entering device can be isolated from the rest of the network. It can be explicitly prohibited from communicating with other devices within the network; likewise, pre-existing devices could be prohibited from communicating with any device that is not formally a part of the local network. Isolation from peers could also be enforced by requiring all communication to be routed through the secure tunnel established with the security manager. In the same way that all wired traffic traverses a local gateway , and wireless traffic must traverse an access point, scalability would require numerous "security gateways" to offer this capability at an enterprise level. The security gateways can then determine which connections to allow, and which to disallow based on policy and privilege granted to individual devices. In many organizations, only minimal peer interaction is actually needed, with most devices communicating directly over local link with a small and typically static subset of their peers.

One other ongoing quarantine task is identifying when devices leave and rejoin the network. Depending on network policy, the rejoining device may go through the same process as others, or a more limited form of

QED based on length of absence, etc. Group membership management can be performed through beacons, keep-alive messages, or another similar technique.

4.2 Examination

Once the prospective client device and network have established a secure channel and quarantine, examination begins. During examination, the network assesses the incoming device and determines if the device is fit for entry. Similarly, the device can potentially examine the network and determine if it meets its needs in terms of offered services, capabilities, etc. In general, we focus this discussion on networks examining devices.

In general, many different mechanisms could be used for examination, including virus scanners, package management tools, network scanners, and configuration analysis tools. Full examination of a device is potentially an immense task. Depending on the size, it would be generally infeasible to examine all of the contents of a device. Instead, a small subset of the various installed packages could be selected for examination. File sizes, checksum values, and version information are some examples of what a package management exam module could check. Obviously, this type of examination will consume time and resources, making the examination phase a performance bottleneck for the QED model. Trust relationships between networks reduce examination time by a great deal. If a device moves from network A to network B, and network B trusts network A's examination process, then network B may not require the same examination. This type of trust relationship works well in an environment where devices may move between numerous different physical networks, but still remain within one administrative domain.

There are multiple modes of examination. Two different examination variables include whether the exam is external or internal, and immediate or windowed. An *external* exam refers to external probing or monitoring to analyze the device. This may include traffic monitoring, port scanning, service fingerprinting, etc. This is less invasive, and results are generated inside the security manager.

Internal examinations refer to exams in which the device cooperates by executing appropriate software. These can include virus scanners, package examination, configuration examination, and so on. Participation in these scans by the device would typically be voluntary. These results would then be generated on the device, and may be suspect. Trust and device-based examination is discussed further in section 7.1.

Since an internal exam requires the device to run code possibly provided by the security manager, safeguards must be used to ensure the integrity of the examination code and to protect the device from malicious examination attempts. Cryptographically signed examination modules, signed either by a trusted third party or by the network in question, can be verified for authenticity. Once authenticated, the device must consult its own policy to determine whether or not to execute the module. Additional techniques can be used to certify module functionality and safety, including proof-carrying code and the use of virtual machines.

Immediate exams and *windowed* exams differ in their time requirement. Immediate exams must be based on results generated during the current QED session. A windowed exam refers to an exam that accepts results generated earlier. An example would be a network requiring a full virus scan daily. If a device already went through a full virus scan and had a signed certificate attesting to that, it can forgo further scans that day.

Exams are not fool-proof and are limited in scope. External exams cannot be perfect, and without advanced hardware support, internal scans can be evaded by a clever piece of code. To that end, examination results cannot be guaranteed and incoming devices have to be dealt with accordingly. Hence, the QED model advocates maintaining partial quarantine by restricting communication to that absolutely necessary, as well as continued monitoring of all peers with feedback to the examination component. If problems are detected later, the device can be placed back into full quarantine and further examination required. Additionally, as new security alerts become available, e.g., a CERT advisory, or a new system update, devices may be required to resubmit to the QED process.

A characteristic of nomadic computing environments is the large degree of heterogeneity in operating system configurations and applications, both in the client devices and the environments themselves. A given environment may not possess sufficient knowledge of a potential client to perform certain types of examinations, such as package scans, etc. Certain basic types of examinations will always be applicable—port scans, service fin-

gerprints, external traffic monitoring, etc. Typically though, invasive operating system scans are generally needed only in areas where they are already easy to perform. Internal scans such as virus scans and package examination would usually be performed in environments such as the home or the office where the device and the network have an existing trust relationship, and also knowledge of the applicable operating system requirements. Within one's home or one's office, it is quite appropriate to require a much more stringent examination and decontamination. It would be perfectly reasonable for a business to require that their employee's devices undergo a rigorous scan before being allowed onto the network; similarly, within one's own home, the network and the device have a pre-established trust relationship. The network would know the recent state of the device, so detecting changes to the system is fairly straightforward.

In public environments, such as the coffee shop Bob visits in our motivating examples, it is not reasonable to expect the user's device to accept intrusive examination or decontamination. External scans, monitoring, and basic quarantine may be all that are acceptable to the discriminating user. However, this is sufficient to prevent the casual spread of many types of malicious code.

4.3 Decontamination

The decontamination component of QED begins after the initial examination is complete. Decontamination attempts to bring the target in-line with appropriate local policy. From a network perspective, this means updating a device's packages, disabling or disallowing local access to some services, requiring configuration changes, or removing malicious code. From the device perspective, it may include asking the network to run a needed service or alter the configuration of an existing service. Again, as our focus is primarily a network-oriented approach, this will be examined more closely in the future.

There are many different ways that a network might decontaminate a device. Virus scanners can automatically destroy or quarantine viruses. Package management tools can apply new security patches and update software/firmware code. The security manager can instruct the client to stop certain services. Again, any software locally executed on the client must be cryptographically signed by a trusted authority before the client device will allow it to run, and techniques used to verify the functionality of mobile code can be used here to offer further protection.

When decontamination is complete, the client is given privileges to communicate and interact as appropriate—the degree of interaction allowed still depends on the determined role of the device, identity of the device, and level of trust in the device's integrity.

Time is a serious constraint in decontamination. For this process to be feasible in real environments, it must be performed as quickly as possible. This requires that the security infrastructure actively and aggressively cache the latest anti-virus and software updates. Work done in web caching and content distribution can be leveraged for this purpose. Profiles of what devices use the network at a particular time and what applications and services are likely to run could be maintained, to assist with the update caching process. If a particular update is not found in the cache, it must be fetched from the appropriate source through the Internet, adding additional overhead to the process. Lastly, decontamination does not have to strictly follow examination. These processes may overlap or interleave to a large extent since their functions are so closely related.

5. The QED Prototype

We have designed and built 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 or vulnerable machines in a restricted quarantine.

The center of our QED framework is the security manager. This is a Linux-based desktop machine that acts as a wireless access point and router for the QED subnet. Additionally, the security manager offers authenticated DHCP and IPsec VPN services.

5.1 Discovery and Quarantine

The QED client software configures the client's wireless interface to monitor all wireless traffic, listening for QED beacons. The security manager for each network sends out beacon messages every 300 ms that advertise the presence of the QED-enabled network. These messages include a description of the network, the network's public key, and the IP address of the local security manager. When a client overhears a QED beacon, it must decide if it wishes to learn more about the network and possibly attempt to join. Currently, laptops operating in the environment have pre-deployed certificates for the appropriate Computer Science networks, making it simple to determine if the network in question is one that the device can use. This technique is inherently non-scalable and is used only for prototype purposes.

We are actively extending this to support a X.509 certificate-based scheme where networks locally publish their certificate, signed by well-known signatories; a prospective client in range of the network will be able to easily acquire the network's X.509 certificate. Additionally, networks can determine whether clients are allowed to request access by examining the client's X.509 certificate. Requiring the appropriate signature or chain of signatures will assist networks in determining whether a given client can join. Signatories might include a user's company, school, or even credit card companies.

Once a client has decided which network it wishes to join, it uses the Dynamic Host Configuration Protocol (DHCP) to obtain all the network configuration information that is necessary in order to become a part of the network and to be able to communicate with the other members. The DHCP protocol [RFC2131] is primarily used to assign a dynamic IP address to a machine that is expected to be a temporary member of a network in an automated fashion. The protocol relies on communication between a DHCP server hosted by the network and the DHCP client on any device that wishes to connect to that network.

For our QED prototype, we use DHCP as a means of assigning an IP address to a client, as well as disseminating other configuration information. We have implemented a modified version of DHCP that allows clients and servers to mutually authenticate themselves. In a DHCP packet, all configuration information is stored in options [RFC2132], one of which is an authentication option [RFC3118]. RFC 3118 gives general guidelines for transmission of authentication information.

Our modified DHCP achieves three things: mutual authentication of client and server, establishment of DHCP session keys, and registration of the client's RSA public key, used for IPsec. The protocol works as follows:

- The client first broadcasts a DHCPDISCOVER message. Included in the message is an authentication option that includes a RSA public key created for this session. The message and authentication option are signed using the client's private key.
- The server verifies the message and the session key. It also caches the client's public session key to later use to authenticate all other messages sent by the client during the current session.
- The server replies with a DHCPOFFER message that offers the client a specified local IP address. The authentication option contains a public key that the server has previously generated, and one that it will use to sign all future messages during that session.
- The client caches the session key. The client sends out a DHCPREQUEST message, signed with its session key. This message requests the IP address offered by the client.
- The protocol finishes when the server answers with a DHCPACK message, again signed with the appropriate session keys. In our prototype, we used the SHA1 algorithm for signing and verification, with 1024-bit keys.

Additionally, when the DHCP server registers the RSA public key of the client, it performs a secure DNS update to publish the acquired key into the local DNS database. The key is mapped to the newly allocated IP address given to the client. We are using the WAVEsec patches to the DHCP daemon that enable this functionality [WAVEsec].

After successfully acquiring an IP address via DHCP, the client initiates an IPsec tunnel to the security manager, using the RSA public key for the security manager acquired from the network beacon. The IPsec implementation on the security manager uses the client key, accessed via local DNS, and the tunnel is established.

The client routes all traffic through the tunnel, denying all incoming traffic that does not originate from within the tunnel. The potential for attack between DHCP address setup and IPsec VPN establishment can be reduced by only accepting inbound IPsec SA establishment packets from the security manager, as identified by the network beacon. Absent vulnerabilities in the IPsec SA setup mechanism, this should prevent attacks issued during this window.

Meanwhile, the security manager is aware of the new potential client. However, no outbound communication is allowed, nor is any other local device allowed to address packets to the quarantined client. The client must successfully complete examination and decontamination before any further interaction is allowed.

5.2 Examination and Decontamination

Our QED prototype supports a framework for examination and decontamination of client devices. Since the particular mechanisms for these processes are network and policy dependent, the implementation takes a modular approach. The framework controls the flow of examination and decontamination, whereas the actual work is carried out by a collection of modules that can be plugged in as deemed necessary. Both the framework and the modules are written in Java; the various modules currently implemented rely on Linux system tools including *rpm*, *nmap*, and a few custom scripts. In our prototype, the examination and decontamination phases largely overlap due to the modular nature of the implementation.

5.2.1 Framework

The framework is divided into two parts: the code that runs on the security manager—the examiner—and the code that runs on a client—the target. The framework code on the examiner reads a configuration file that specifies which modules to run in what order for each supported Linux distribution. The configuration file can also specify arguments to pass to modules. When the IPsec VPN with the potential client is established, the security manager initiates examination.

Once examination is initiated, the client device's operating system version is determined by querying the examination daemon running on the device. Once the examiner determines which modules need to be executed, the information is sent to the target, which checks its local module cache. If a module already exists in the target's cache, its MD5 checksum is compared to that of the same module on the examiner. If the MD5 checksum matches, the cached copy is loaded and executed; if it doesn't match, the cached copy of the module is overwritten by an updated version transferred from the examiner. Examination modules can additionally be signed by the network, or by an accepted third party signatory.

In our environment, the user is given complete discretion regarding what code is executed on his device. Therefore, the framework allows the target system to query locally cached modules for information describing their functionality. Using this information, the user can determine if it is acceptable for the modules to run. The user can also take into account the time for each module to run and the amount of privacy maintained by the module. Depending on the user's decision, the framework then runs the module or skips to the next one. This process can also run in an expedited mode where approved modules execute automatically.

5.2.2 Modules

The modules, like the framework, can be separated into two categories: code that runs on the examiner and code that runs on the target. The two parts communicate with each other to accomplish the various examination and decontamination tasks. The modules can be written to be generic or specific to a operating system distribution. An individual module can also be written to handle both examination and decontamination.

Before a module runs, it must provide information about its functionality to the user through the framework. After a module terminates execution, it returns its status to the framework. Using this status, the module can control the flow of the examination and decontamination session by specifying other modules to run. For example, a module can be designed to check for a very specific virus. If the target is infected with that virus, the module can instruct the framework to load a separate module that will remove the virus. Once this decontamination module removes the virus, the framework loads the next module as indicated in the configuration file.

We have implemented the following modules:

RPM Check

This module examines the target machine to see if any of the installed packages are out of date. It uses a local archive of update packages that are automatically synchronized with the appropriate update directories by the security manager. Currently our prototype supports RedHat 7.3, 9.0, Fedora Core 1 and all packages are stored as RedHat Package Manager files (RPMs). Work is ongoing to add support for Debian Linux.

The examiner sends a list of package names to the target. The target module collects the relevant information from its local RPM database and sends it back to the examiner. The examiner compares this information against the packages stored in the local archive. If any of the target's packages are out of date, the examiner will indicate this to the target and then instruct the framework to run the RPM Update module.

RPM Update

This module is only run if the RPM Check module determines that there are packages on the target that require security updates. It expects a list of packages to update from the RPM Check module. The new RPMs are transferred to the target and the *rpm* tool is used to update the target. We rely on rpm's dependency checking to manage and avoid conflicts.

In general, this is satisfactory; however systems that mix rpm-installed applications with hand-installed applications may run into problems with conflict management. Unfortunately, the use of a system-wide cross-package database to record and manage installed applications and services, similar to the Microsoft Windows registry, is not yet widespread in Linux.

One concern here is that updates may be required extremely frequently and impose undue overhead. In general, updates required to fix security vulnerabilities are needed regularly, but not extremely frequently. For example, security updates for RedHat Linux 9 were issued on the order of one update every four days and with an average size of 271 kilobytes.

Port Scan and Service Blocking

This module relies on the *nmap* tool [Nmap] to scan the target to identify which services are running. The examiner initiates the scan and compares the results to a pre-specified policy to determine whether the target is running any services that are disallowed by local policy. Using this information, the examiner instructs the target module to block all incoming traffic on unacceptable ports using iptables firewall rules [IPtables]. Additionally, the examiner can block routing for the given ports, not allowing any other computers to access the blocked ports. The examiner then rescans the target to check that the rules have been put into place.

File Property Check

This module is essentially an integrity check on the most commonly used executables to check for signs of tampering or the presence of Trojan horses. It uses the information RPMs carry about the files they install; examples are file location, file permissions, and MD5 checksums. The examiner requests information for specific files from the target. It then extracts the same information from the local archive of RPMs and compares it to this. If inconsistencies are found, they can be dealt with in different ways, depending on policy and file type. It is important to note, that this type of an examination is less useful to those who frequently rebuild their systems and apply custom patches. In general though, for most user systems, binary executables change infrequently, and inconsistencies may be a cause to alert a system administrator. Temporary fixes are possible, such as archiving the altered files and using the RPM Update module to overwrite the altered file(s). It is important to be careful and avoid inadvertently destroying user data.

This module could also compare current system state to a recorded baseline state, and identify recent changes. This would be very useful for one's home or office environment which can maintain logs of the contents of one's machine when it last visited the environment, and easily detect unauthorized changes to the system contents.

The examination and decontamination framework is open and extensible; it can easily support other modules such as network-specific reconfiguration, etc. It can also support modules relevant to other operating systems.

5.3 Network Management

The QED network is managed via periodic heartbeats from the gateway and clients. The heartbeat must be properly signed with the gateway's private key in order to prevent false messages, tampering, and replay. The QED heartbeat serves two purposes: announcement and revocation of membership.

If the QED security manager does not hear a heartbeat within a configurable time period, the IPsec SA is torn down, and no further traffic is accepted. The client must then reestablish the IPsec SA, and potentially be subjected to QED again, depending on local requirements.

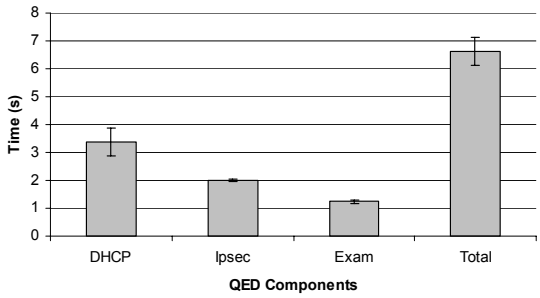

Figure 3. Time spent completing QED for an up-to-date machine.

6. Experiences with QED

Our QED prototype has been deployed within our research group. We have constructed several experiments to measure the impact upon connectivity and the user experience. Results indicate that the added overhead is fairly small, and in general does not affect the user experience significantly.

6.1 Experimental Results

We measured the various stages involved in joining and becoming an active member in a QED-enabled network. In the LASR network, QED is principally used to determine the current version of installed packages and identify externally offered services. If packages are found to be out of date, updates are provided to the device. Similarly, if an undesirable network-based service is running, the device is instructed to firewall off the appropriate port. In the most common case, when a machine is determined to be up to date, the total overhead involved in joining the wireless network is 6 to 7 seconds.

Experiments were conducted to measure the incurred overhead when system updates were required. These experiments involved two nodes. The first node consisted of a 1.3 GHz AMD Athlon equipped with 1.5 GB of RAM. This node was the QED client, operating on top of RedHat Linux 9.0, with kernel version 2.4.20. QED executed under the Sun JDK 1.4.2. The second node was the security manager, and consisted of a 2.53 GHz Intel Pentium 4 equipped with 512 MB of RAM and running RedHat Linux 7.3 and kernel version 2.4.18. The implementation of IPsec used was Linux FreeS/WAN U2.04. The experiment was designed to mimic the behavior of a device joining the 802.11b wireless network. The DHCP implementation used in these measurements was ISC DHCP v3.0.1rc11, and the RPM implementation was RPM v4.2.

When the client device saw a beacon announcing the QED-enabled wireless network, it automatically joined the network, getting a DHCP address and initiating an IPsec security association (SA) to the security manager. It is important to note that the DHCP caches are cleaned out after each experimental trial, so the existing leases are destroyed. This has the effect of forcing the DHCP client to go through the full four-way handshake. The security manager then initiates the examination process across the SA. Decontamination, if necessary, is performed after examination, and then the client is given full network access. Tests were run multiple times, and all results are reported with 99% confidence intervals.

To observe the range of possible delays introduced by the QED package examiner and decontamination modules, we tested five different cases. We tested the null case (where no updates were necessary) and individual updates of four different representative RPM files, each with very different characteristics. The null case is the typical user experience when his or her machine is up to date. QED in this case can be broken down into three parts—the time to acquire an IP address via DHCP, the time to establish an IPsec SA, and the time

	Exam Only	Exam and install pam_smb-1.1.6	Exam and install gdm-2.4.1.3	Exam and install foomatic-2.0.2	Exam and install perl-5.8.0-88.3
Elapsed time to examine and install (in seconds)	1.24	1.88	13.14	23.90	91.14
Confidence (99%)	0.01	0.01	0.73	0.41	3.66
RPM size: (in Kbytes)		32	1,466	1,379	14,143
# of files:		8	122	1875	2560
Package Density (files/Kbyte)		0.25	.08	1.36	0.18
Uncompressed file size (in Kbytes)		68	3,962	19,373	34,123
Compression ratio		2.1:1	2.7:1	14:1	2.4:1

Figure 4. Results from examining one machine, and a detailed breakdown of the four packages updated, and overhead incurred from applying the update.

to transfer the current installed package list to the security manager and examine it. Figure 3 presents the results from this experiment. Absent QED, devices would typically experience only the DHCP overhead when joining a typical DHCP-enabled network. Establishing the SA and the package examination incurs ~3 seconds of additional delay before the device receives network access.

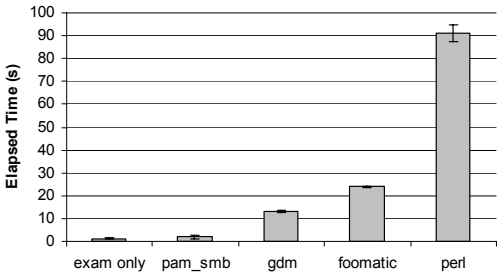

Figure 5. Time spent examining the client package database and installing individual packages.

If an available update is found, our QED prototype initiates a transfer of the appropriate RPM to the client device. To simulate this, we forced downgrades of four different packages and then measured the time to perform QED, allowing the security manager to update the packages in question. Figure 4 shows the packages selected, and presents a breakdown of the time to install each package and individual package information. The packages chosen are fairly representative of typical RPMs that might be updated. Both ends of the size spectrum were represented. Another factor we considered is *package density*—this refers to the number of included files within a package divided by the file size. We believe that package density should affect installation time and thus QED overhead, with those packages with greater files per kilobyte requiring more time to unpackage and install. Figure 5 graphically depicts the relative installation times for the different packages. As expected, the size of the file and file density both greatly impacted the time to install and update the given package. Overall, the overhead seems quite reasonable. At the extreme end, updating *perl* required approximately 1.5 minutes, but this is an extreme case.

Additional examination benchmarks include nmap scans of incoming client devices. Measurements demonstrate that the security manager can perform a brief port scan of 200+ commonly used ports in 1.15±0.3 seconds. This can be performed in parallel with other active examination modules.

6.2 Discussion of Results

These experiments have shown that QED is effective at implementing local policy and maintaining our user machines, while typically adding very little overhead. Typical security updates of 1 to 2 Mb could be downloaded and installed in 10 to 20 seconds. Very dense packages, such as *foomatic-2.0.2* require more time at the client side to install, compared to files of comparable size and lower file density; as a point of comparison, the average file density of all packages in the RedHat 9 updates is 0.18; *foomatic* is unusual with its density of 1.36. We feel that overall, these times are quite reasonable. The typical case for most users will be that their machine is up to date, and they experience only ~3 seconds of added delay. This is acceptable for most user demands, and may even go unnoticed. Longer delays are possible; however they should be infrequent.

7. Future Directions

QED currently works in a wireless context, and was designed with wireless, ubiquitous computing environments in mind. However, the fundamental concept of QED is equally applicable to a wired environment. One obvious direction for future work is to create a wired version of QED.

The QED security framework is envisioned as part of a much larger framework for nomadic and pervasive computing, which we call *Panoply*. The aim of the *Panoply* project is to enable devices to interoperate seamlessly and access services, all within a secure framework. In this envisioned framework, devices need to negotiate policy with the environment and obtain access to desired resources. Many important issues not yet addressed within the QED framework (such as security policy issues and the security of composing particular services in a ubiquitous environment) will be addressed later in the *Panoply* project. Further, some aspects of QED may well need enhancement to support a ubiquitous environment. For example, different QED environments that are physically adjacent or logically related might need to have some degree of cooperation in their handling of devices on the move between them. However, just within the QED model, there are many remaining challenges that must be addressed.

7.1 Trust and Device Participation

There are substantial trust issues that must be dealt with in QED. The most obvious issue involves requiring potential clients to execute examination and decontamination modules provided by a network's security manager. Modules may potentially be malicious. Alternately, devices or malicious software may lie or subvert the results of exams. The problem of ensuring the integrity of an operation is an old one. It has been investigated in the context of virus scanners, anti-piracy mechanisms, and digital rights management (DRM).

We believe that the situation is not as dire as it sounds. Given currently available systems and technology, QED can be a valuable tool. For example, had QED been available at the times of their release, the existing model would have protected against Code Red, Code Red II, Slammer, Blaster and the vast majority of known viruses that depend on exploiting well-known vulnerabilities. In its ongoing examination mode, QED would detect and be able to stop the denial-of-service activities of many popular DDOS tools in their most common modes of operation. Given the natural limitations of examination and decontamination, it is still possible to greatly improve system security through proactive device management and through the use of aggressive quarantine. To that end, one view of QED is as an intrusion tolerance tool that increases the overall preparation and resilience of user devices, reducing the likelihood of infection or compromise.

QED has the potential to be much more powerful, as trusted computing hardware becomes more widespread. The use of a trusted computing platform, such as TCPA [TCPA], beneath QED would allow numerous enhancements. The security manager could be certain of the fact that the client is not making unauthorized communication. Trusted modules and operating systems could not lie or otherwise subvert examination results or the decontamination process. Simply deploying widely deploying TCPA is not the sufficient, unfortunately. A great deal of work is still required to build secure examination components, and the underlying operating system components upon which we would rely.

7.2 Privacy

There is a fundamental tradeoff between the ability to examine machines and the privacy desired by users. The more invasive the examination procedure is, the better the guarantee of security. The tradeoff itself could provide a solution to the problem. Networks could define various levels of access. Devices could choose limited sets of examinations they are willing to undergo in exchange for limited access. The network will always have the right to deny access to a device if necessary, and the user will always have the right to decline to submit to a particular investigation, at the cost of not receiving access.

With TCPA-based hardware, we foresee a solution to the privacy problem. A device could provide a cryptographic guarantee of the absence of vulnerabilities or malicious code that the security manager could trust, thereby not compromising security and also preserving the privacy of the device's contents.

7.3 Scale

With increasing number of devices entering the market and supporting heterogeneous platforms, the number of modes of attack is going to increase dramatically. Security infrastructures must be prepared to allow all kinds of devices to join their networks. Much as we would like to have individual security managers store all the information about all possible platforms, it is infeasible for a practical system due to scale; the storage space at the local site is limited. Instead, security managers could maintain profiles of the more widely used platforms, especially those that are encountered frequently in the local network, and store relevant application patches and anti-virus software. If an unknown device comes along, the security manager must procure the necessary information through the Internet. We will be investigating the scaling issue in the context of our prototype.

7.4 Other Examination Approaches

Our existing examination services demonstrate that useful forms of examination are feasible and perform well in the QED framework. As our experience with QED develops, we anticipate building other examination modules. For example, as other researchers discover innovative ways to detect worm spreading behavior, we are likely to incorporate some of those in a QED examination module.

The inherent modularity of QED's examination component will make such additions easier. However, we will need to exercise some care in adding new modules to the examination suite. While our experiments show that reasonable examinations can be done in acceptably short times, if the set of examination modules keeps increasing, so will the time required to perform examination. Either we must be careful about which modules to add or we must put in more sophisticated mechanisms to prioritize different examination modules and limit the total amount of time spent on examination to a reasonable level.

8. Conclusion

Mobility has become commonplace. Nomadic users take their devices from network to network, thinking little of the dangers to which they are being exposed. Unfortunately, those networks are ill-equipped to defend mobile devices, or even alert them to potential threats. In the home and office, networks need to be able to impose policy upon these devices and limit participation based on trust metrics. As existing tools are unable to provide this facility, new techniques are needed to insulate devices from one another, dynamically apply locale and network-specific policy, and provide update and configuration services to this new breed of highly mobile and dynamic users.

This paper presents a working approach to solving this problem. QED strikes directly at the core of the problem by allowing potential client devices to establish a relationship with a new network in a safe and secure manner, inside a highly restricted quarantine. The new network may examine visiting devices and impose constraints upon the device's behavior and configuration to offer extended protections, allowing for secure device update and configuration and simplifying many normally tedious administrative tasks. Additionally, QED gives power to the nomadic user by allowing them to control the degree to which they are willing to conform to a network's demands. QED is highly extensible and the current prototype can be used to provide security

updates, deploy configuration files, implement policy, as well as isolate devices from potential attackers. In the future, QED will be used to build powerful security mechanisms for ubiquitous computing environments.

References

[Blaster] Security Response for W32.Blaster.Worm; http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.worm.html

[Cisco2003] White Paper - Cisco NAC: The Development of the Self-Defending Network. http://www.cisco.com/en/US/netsol/ns340/ns394/ns171/ns413/ networking_solutions_white_paper09186a00801e0032.shtml

[Hilley2003] Sarah Hilley. “MSBlaster could have been MUCH worse,” CompSec Online, Aug 13, 2003. http://www.compseconline.com/analysis/ 030813msblaster.html

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

[Lockwood2003] J. W. Lockwood, J. Moscola, M. Kulig, D. Reddick, and T. Brooks. "Internet worm and virus protection in dynamically reconfigurable hardware," *Military and Aerospace Programmable Logic Device (MAPLD), p. E10, Sept. 2003.*

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

[RFC2131] RFC 2131 - Dynamic Host Configuration Protocol. http://www.faqs.org/rfcs/rfc2131.html

[RFC2132] RFC 2132 - DHCP Options and BOOTP Vendor Extensions. http://www.faqs.org/rfcs/rfc2132.html

[RFC3118] RFC 3118 - Authentication for DHCP Messages. http://www.faqs.org/rfcs/rfc3118.html

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

[WAVEsec] WAVElan SECurity using IPsec http://www.wavesec.org