Quarantine Net: design and application

Matthijs Bomhoff[1], Casper Joost Eyckelhof[1], Remco van de Meent[2], Aiko Pras[2]

[1]Quarantainenet v.o.f.
Witbreuksweg 377-215
7522 ZA Enschede
The Netherlands
{matthijs,casper}@quarantainenet.nl

[2]University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands
{r.vandemeent,a.pras}@utwente.nl

Abstract. Lately the world has seen an alarming increase in both the amount of new computer viruses and worms, as well as the speed at which they spread. Dealing with such outbreaks puts a heavy burden on end-users as well as operators of (enterprise) networks. In this paper we present an overview of, as well as operational experiences with *Quarantine Net* (QNET), a system that has been developed at the University of Twente. The goal of QNET is to limit the impact of computer viruses and worms outbreaks by quarantining infected systems in an automatic way. QNET not only reduces the speed at which viruses and worms spread, but also helps infected users to clean their systems and restore connectivity in a fast and easy way. Deployment at the University of Twente's network has shown that QNET is an effective system to deal with outbreaks of computer viruses and worms, and sometimes achieves earlier results than popular antivirus tools.

Keywords: security management, network monitoring, computer viruses and worms.

Introduction

- Motivation
  - Increasing number of viruses and worms
  - Viruses and worms spread faster
  - Big impact on (large scale) networks
- Context
  - University network, over 10000 workstations, multiple access technologies
- Objectives
  - Reduce workload of the abuse department
  - Increase awareness of users about security issues
  - Quarantine infected systems to prevent spreading
  - Improve response times in disabling and re-enabling connections

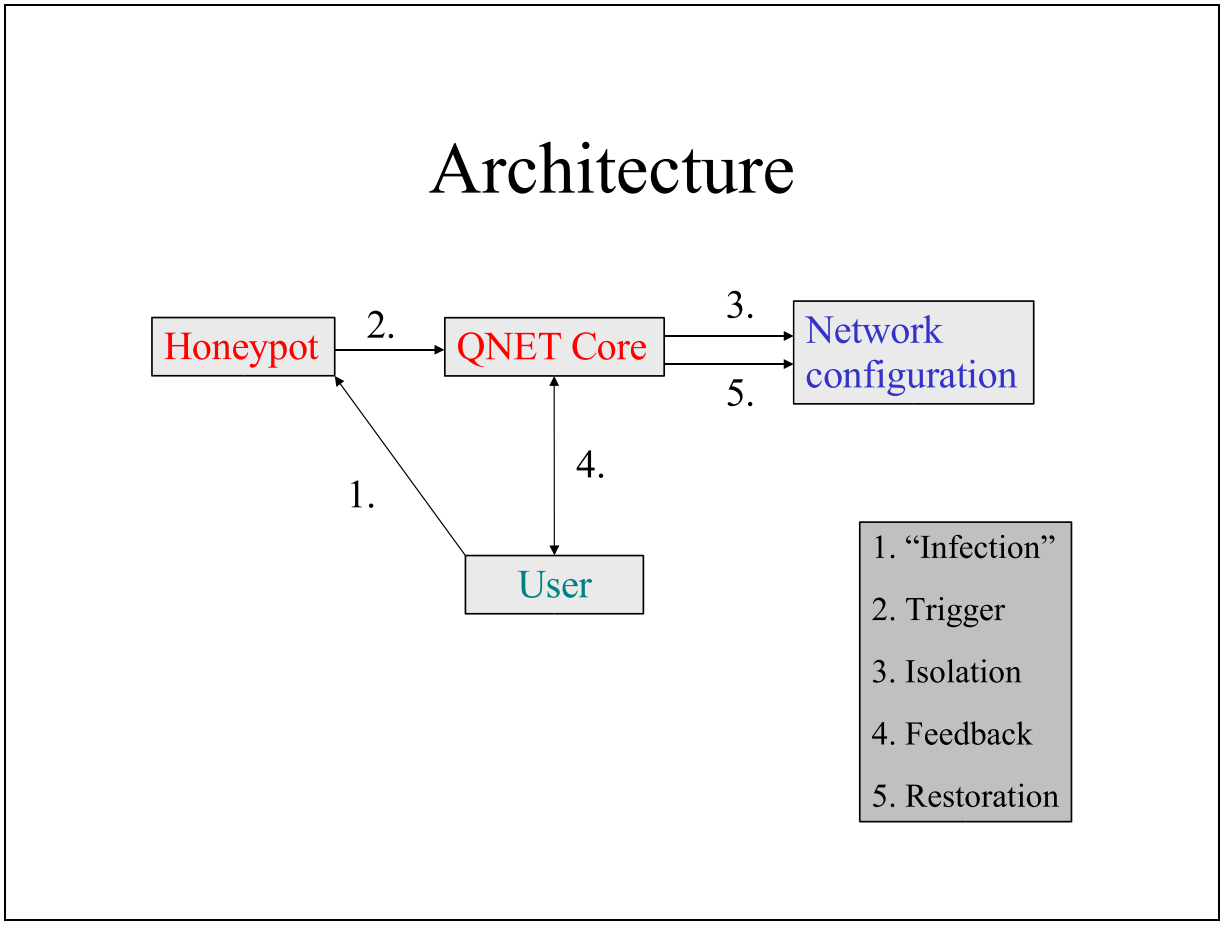
Motivation & Context. The University of Twente (UT), the Netherlands, provides Internet access to its employees and students, both on- and off-campus. Access technologies vary from cable modem and ADSL, to 100 Mbps and 1 Gbps Ethernet. In total, the UT's network comprises about 10000 network access points.

Just like other (large) organizations, the UT has to deal with the negative side-effects of such a large and high-speed network: network abuse, computer viruses, computer worms, denial of service attacks, etc. The UT employs *abuse handling staff* at helpdesks, to assist users dealing with these matters, handle complaints and, importantly, actively reduce the effects of computer infections.

In the past, the typical way to deal with a virus- or worm-infected machine, was to (i) deactivate the corresponding network access point manually as soon as infection was identified (usually via a third party complaint), (ii) contact the user, (iii) have the user "somehow" clean his system (which could be difficult, since the machine was no longer connected to the Internet), and (iv) finally re-enable the access point. The number of hours spent by our helpdesks to resolve abuse related affairs increased steadily throughout the years, mainly because the number of such incidents grew.

The world-wide outbreak of the *Blaster* worm in Summer 2003 had a significant impact on the UT's network operations, and overloaded the abuse handling staff. This outbreak triggered the development of *QNET* (which stands for *Quarantine Net*), a semi-automatic system to reduce the crippling effects of computer viruses and worms.

Objectives. The goals of QNET are to (i) reduce the workload of the abuse handling staff; (ii) increase awareness of network users of security related issues; (iii) place infected systems in quarantine, to reduce further spreading of the virus or worm; and (iv) improve response times in disabling network access points in case of an infection, and restoration when the problem has been resolved.



Quarantining. The general idea behind QNET is that computer systems are *put in quarantine*, i.e., isolated from other systems, when they are found to be infected with a virus or worm. Furthermore, QNET encourages and helps users to clean their system.

Design. QNET consists of several parts: (i) detection, which is implemented as a *honeypot* that "attracts" and detects viruses and worms, (ii) isolation, which is realized by changing the network configuration, (iii) supervision, which is implemented by the *QNET Core*. The entire "QNET process" comprises five steps, as described below. In the next slides, these steps will be further detailed.

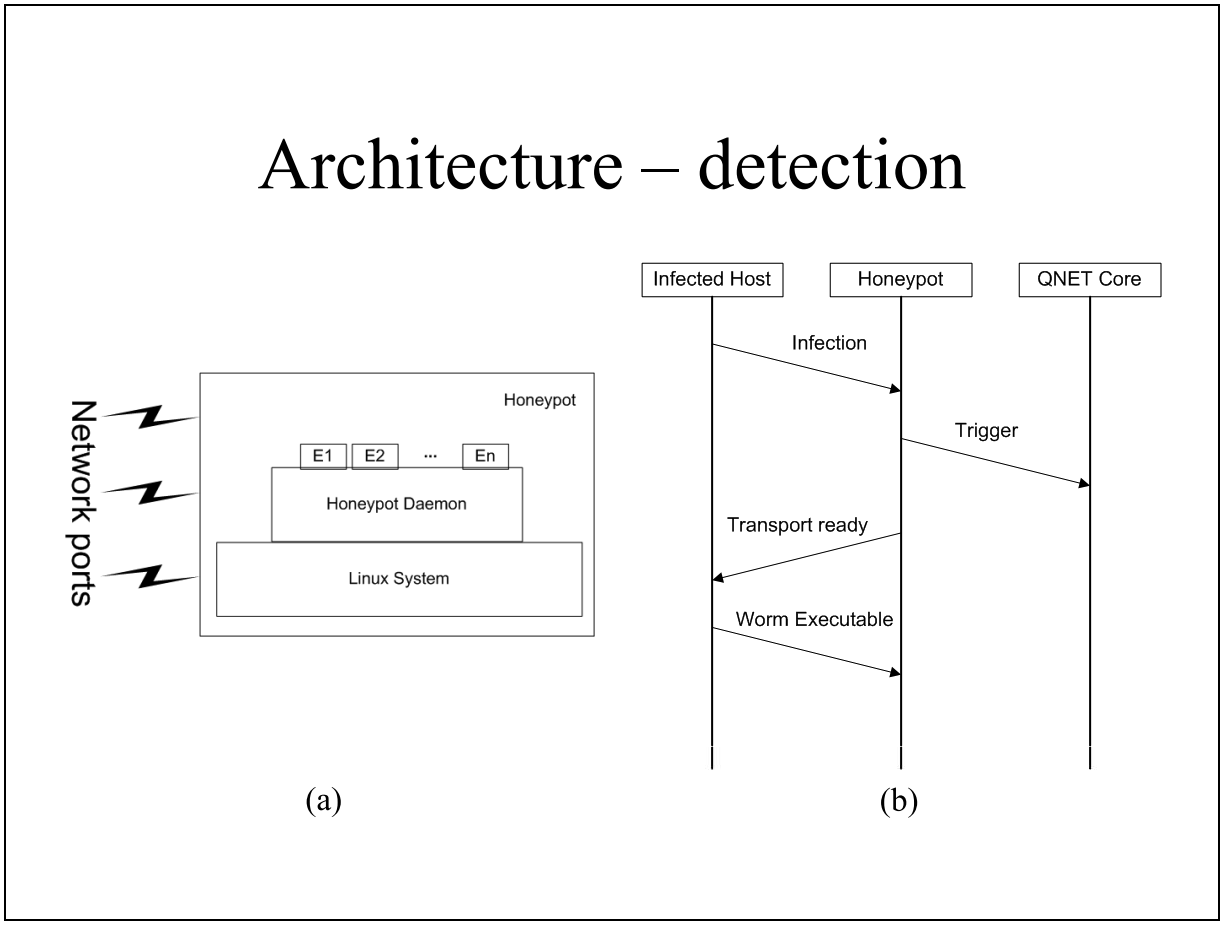
1. When a user's system is infected (how this might happen is outside the scope of this paper), the virus or worm may try to spread itself by contacting and exploiting vulnerabilities in other systems on the network. One of these other systems is a *honeypot*, which acts like a vulnerable system by "mimicking" the behavior of systems that are really vulnerable.

2. After the honeypot has detected that a specific system spreads a virus or worm, it identifies the specific threat (e.g., a variant of the worms exploiting the infamous LSASS vulnerability) and informs the *QNET Core* with information about the threat and infected system.

3. The QNET Core quarantines the infected system, by modifying the configuration data of various network elements, e.g., routing tables, DHCP servers, etc.

4. The user of the infected and now quarantined system is informed of the infection, via email and a webpage informing him of the threat. He is also given instructions on how to deal with the specific virus or worm in order to clean his system. Apart from traffic to websites of, for instance, Windows Update and antivirus companies, all the user's traffic to the Internet is redirected to a QNET server, to limit further spreading of the virus or worm.

5. After the user has cleaned his system, normal networking connectivity may be restored by undoing the previously made changes to the network configuration.



Detection. Viruses and worms can be detected in several ways. One way is to take advantage of virus-scanning software that is usually running on mail servers. Another way is to install an Intrusion Detection System, such as Bro or SNORT [1, 2]. In this paper we focus on the idea of a honeypot, which is a special system that is designed to attract "hostile activity" by acting like being vulnerable to all possible security threats.

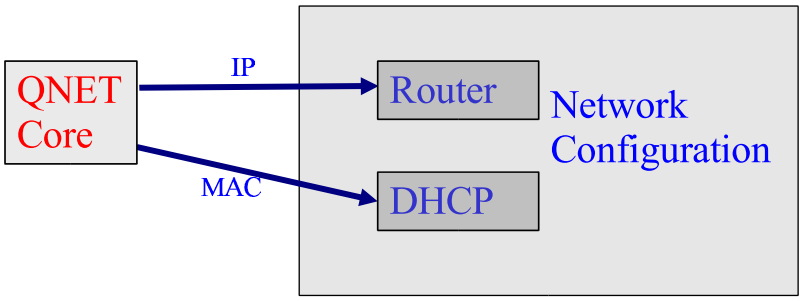
Reactive honeypot. For a variety of reasons, most computer worms spread by exploiting vulnerabilities in Windows network services. Therefore the honeypot *appears to run* the same services as a normal Windows computer. Moreover, it *identically reacts* to exploitation as (ordinary) Windows hosts. In reality, however, the honeypot runs the GNU/Linux operating system (figure a), and includes a *honeypot daemon* and various *detection modules (E1 ... En)*, that together listen to various TCP/UDP ports and behave as a vulnerable Windows system. This reactive approach has as advantage that it reduces the risk of false positives (i.e., systems that would be regarded as being infected, while actually being clean), and allows the collection of exploit code. The time-sequence diagram (figure b) illustrates how such code collection works. In fact, current worms infect a vulnerable system in a number of steps. The first step is to prepare a simple transport mechanism (e.g., TFTP). The second step is to use this transport channel to transfer the real worm executable to the victim. Subsequent execution of the real worm completes the infection process. Because the UT honeypot also completes the second step (obviously, it does not execute the worm itself), the worm executable can be saved by the honeypot for further analysis – more than once we've found worm executables not known to common antivirus programs at that time.

The honeypot currently reacts to some 20 different exploit codes, including the infamous DCOM exploits (responsible for the Blaster outbreak) and LSASS exploits (used by worms like Sasser in 2004).

Architecture – isolation

Two main categories

- Fixed: isolation based on IP address
- Mobile users: isolation based on MAC address or user credentials



Isolation techniques. After a system has been identified as being infected with a virus or worm, it should be isolated to prevent further spreading: quarantining. Various approaches to achieve quarantining are possible, e.g., physical disconnection from the network, putting the host in a designated "quarantine VLAN", and MAC or IP address based blocking. The first two options were not feasible at the UT. Physical disconnection is too labor-intensive, and prevents that infected systems download new virus definitions or software updates. VLANs require support at every network switch, as well as central network management. As people may connect (and manage) their own switches, use of VLAN technology was not desirable. Thus, for QNET we have chosen to implement isolation based on blocking (actually: redirection, as detailed below) of MAC and IP addresses.

Access technologies. The UT's network facilitates both *fixed* (Ethernet, ADSL, cable modem) as well as *mobile* (WLAN, roaming clients) access. All computer systems are assigned an IP address (and other configuration) through DHCP. Mobile users are required to authenticate before they can use the network, via Radius authentication and/or registration of their system's MAC address.

Blocking. When an infected system is put in quarantine, its Internet traffic (except Windows Update, etc.) is not entirely blocked, but instead *redirected* to a QNET server. This is achieved via *policy based routing* (i.c., routing based on source IP address) at a central IP router. Thus, the router needs to known which systems (actually, their IP addresses) are put in quarantine:

- In case of fixed network access, the router is informed by the QNET Core of the corresponding IP address, to update a "quarantined" *access control list* on the router.
- In case of mobile access, the QNET Core updates the DHCP server configuration such that the IP address of the infected system is changed (via DHCP) to an address from a specific "quarantined" IP range.

Note that this fault isolation procedure also allows to put "unknown" (e.g., new) systems in quarantine, informing the user to register, and encourage him to update his system with the latest security patches, before "full connectivity" is given.

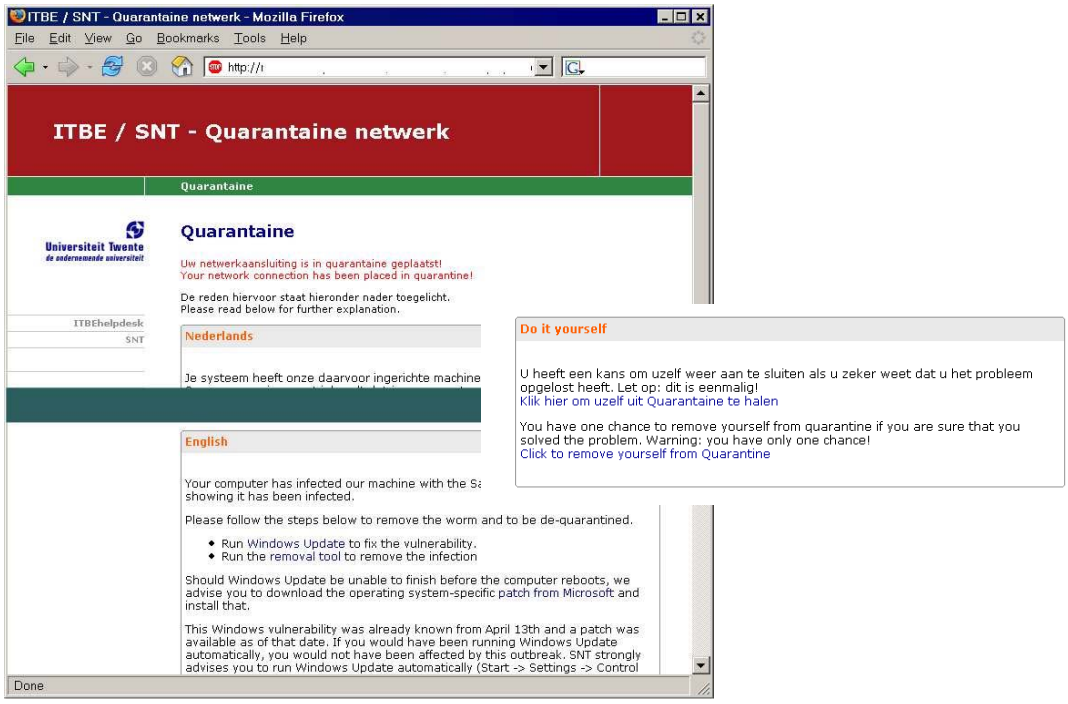
Architecture – user feedback

- Email notification on isolation
- Personalized web page
- Restricted internet access through proxy
  - MS Windows Update
  - Antivirus Software updates
- Contact helpdesk using web form or use OneStrike

While in quarantine. When a user's system is quarantined, this user has only very limited networking possibilities: the user is able to access his email, and the user may access various websites that may assist him in cleaning his system, such as Windows Update and sites of well-known antivirus companies, through a proxy run by QNET. When the user tries to access any other webpage, he is redirected to a *QNET web-server*, and the user is shown a webpage indicating that his system is placed in quarantine. Importantly, this (personalized) QNET webpage also shows *why* the user's system was quarantined, and *how* he can clean his system and restore full network access again. This personalized information is also sent to the user via email, upon quarantining.

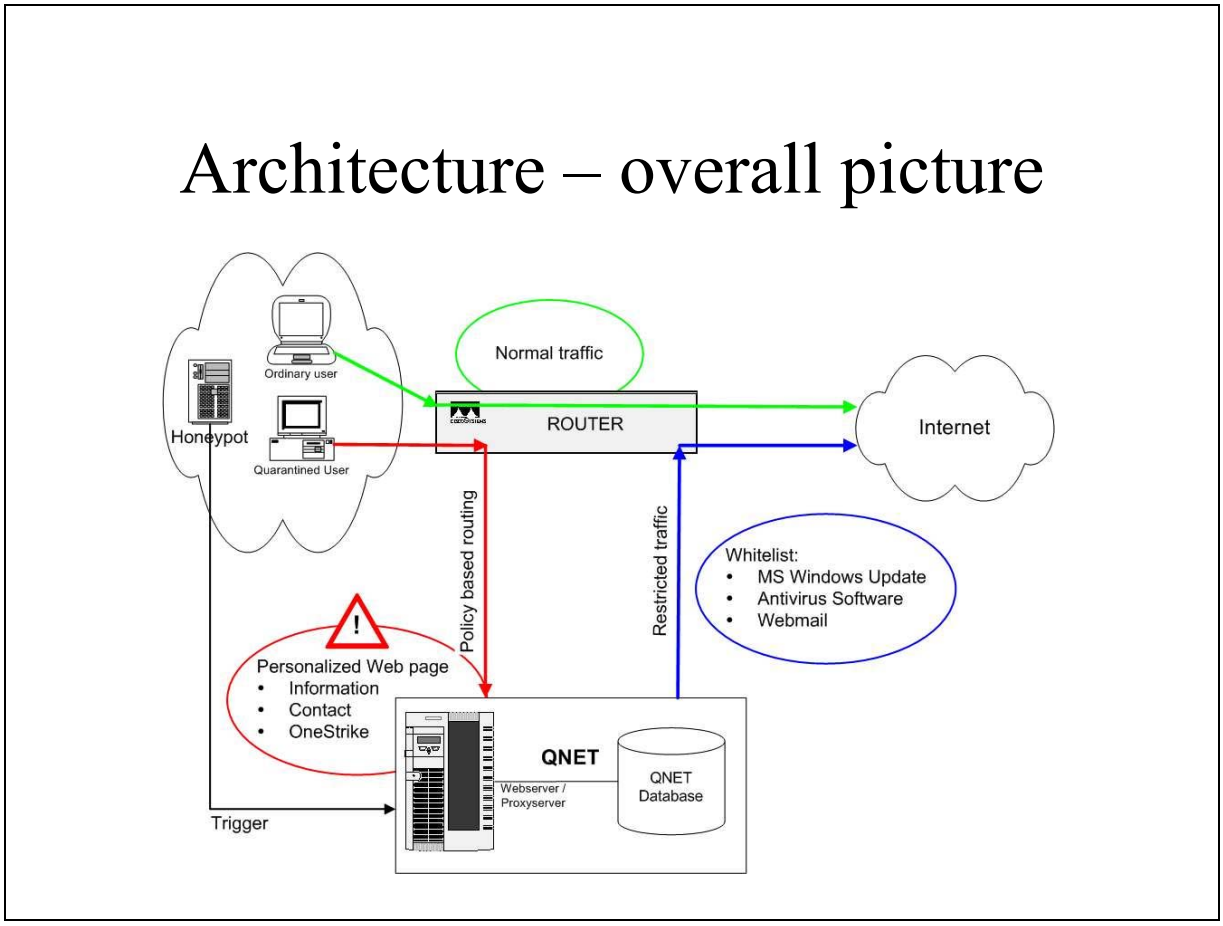
De-quarantining. After the user cleaned his system, he informs the QNET server of this. The user may be given a *OneStrike* option: the user (re)confirms his system is clean of viruses and worms, and full network access is automatically restored. Alternatively, the user informs the helpdesk, and after abuse handling staff manually *de-quarantines* the user's system, the QNET Core makes the necessary changes to the network configuration.

User feedback - example



Personalized QNET webpage. The screenshot above shows the (personalized) webpage that is shown to a user who's system is put in quarantine. The webpage lists the reason why this system is quarantined (i.c., infection with the Sasser worm), and outlines the steps that can be followed to clean the system.

Also (see inset), there is a *OneStrike* option to trigger "de-quarantining". The user is also given the possibility to contact the helpdesk, through a web-form. When the user does so, the QNET server automatically includes all relevant information in the resulting email to the helpdesk.



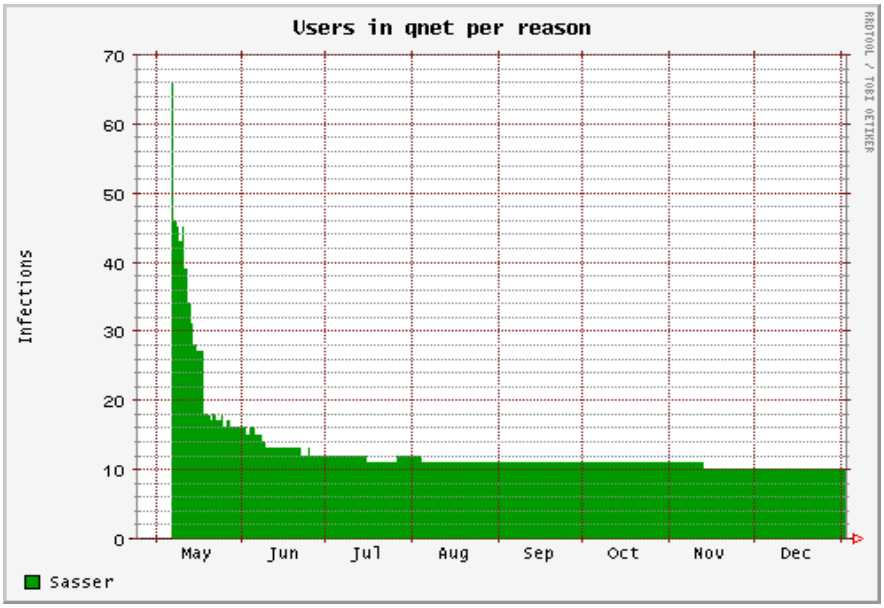
The picture above gives a schematic overview of the entire QNET system:

Ordinary users. Users of systems that are not quarantined (thus, the big majority of users) are not hindered in their Internet access; their traffic is not impacted by QNET in any way.

Honeypot. The honeypot resides in and listens to the same network as the users. Compared to "a honeypot far away", this approach increases the probability to catch a newly infected system as soon as possible after infection. In the UT case, this means that the honeypot is connected to all subnets. When the honeypot notices a new infection, it triggers the QNET Core, which takes care of isolating the infected system.

Quarantined users. Network traffic from systems that are quarantined is redirected by the router (which was configured by QNET to do so) to a QNET server. QNET makes sure that a quarantined system is still able to connect to several "whitelisted" systems as indicated in the picture, by "proxying" the corresponding network traffic. These whitelisted systems include antivirus sites, Windows Update, as well as some "mission critical" systems, e.g., class schedules and exam results in the UT case. When a user tries to visit any other website, a personalized QNET webpage is shown to the user instead.

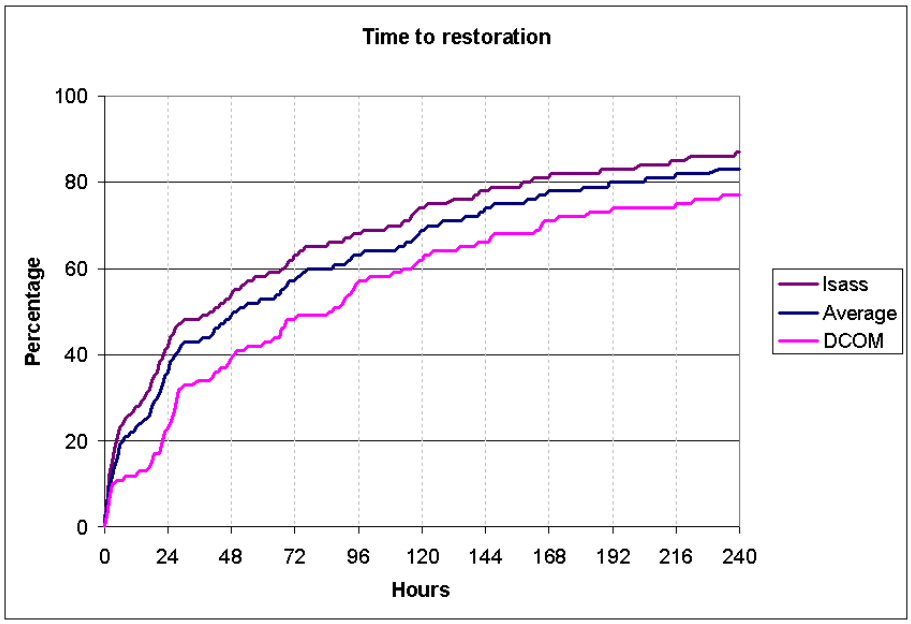
Operational results – example: Sasser



Sasser worm. During the year 2004, the Sasser worms, exploiting the LSASS vulnerability in Windows, caused widespread infections, and also on the UT's network a lot of systems were affected. The above graph shows the amount of systems quarantined over time, on the grounds of infection by Sasser.

Clearly, the number of quarantined systems reduced drastically soon after the initial outbreak. Such decay is the logical consequence of many systems being infected during the outbreak, and de-quarantined after cleaning in the hours and days to follow. In total, 136 infected systems were identified - 10 of them are still in quarantine at time of writing. Possibly these are systems that are no longer active at the UT's network.

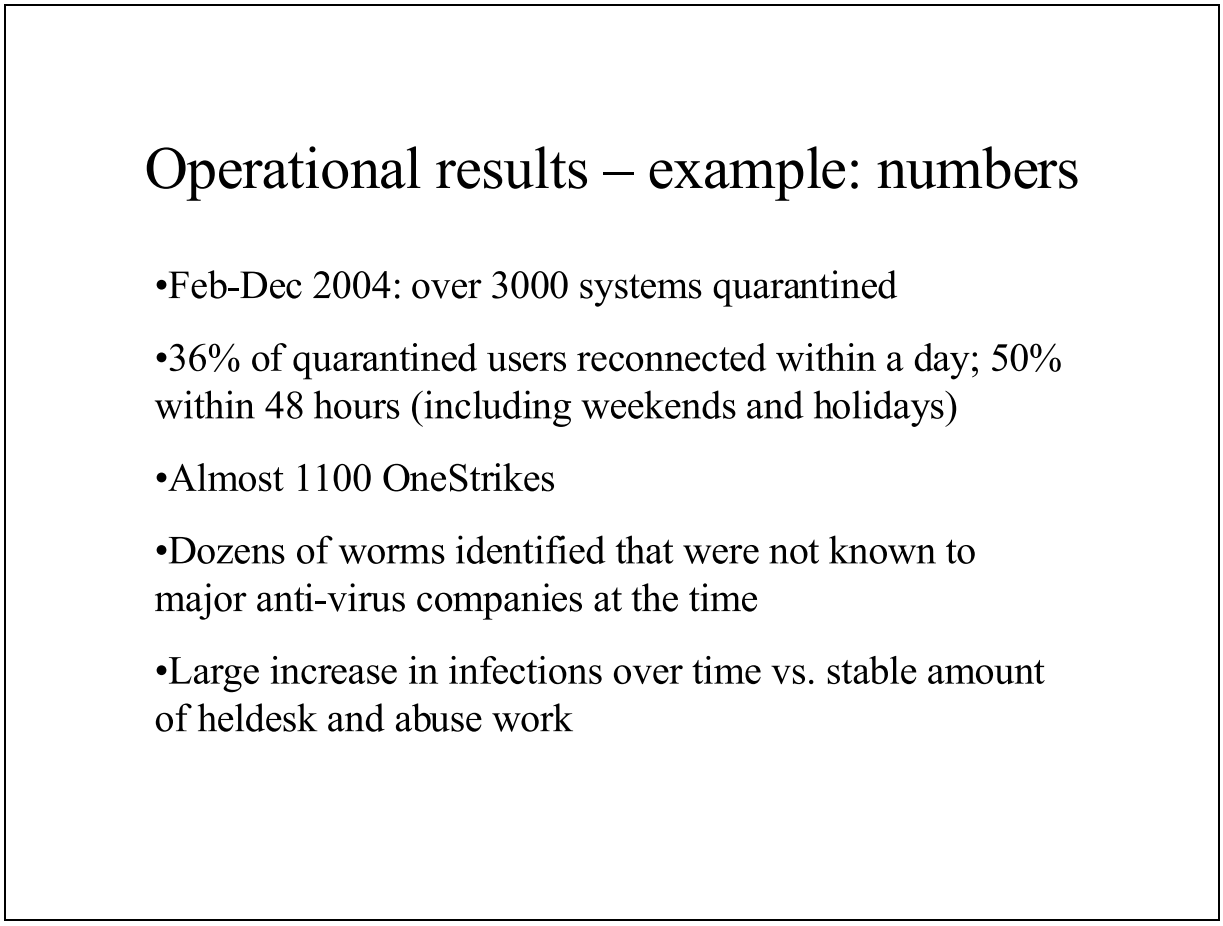
Operational results – time to restoration



Response times. Before QNET was deployed at the UT's network, infected systems were disconnected (usually after complaints) manually, by either disabling the specific network interface on the switch the infected system was connected to, or even physically removing the connection. This procedure led to "high" response times (up to 24 hours, assuming weekdays). Also, restoration of the network connection required manual intervention, which may take considerable time (could be up to several days, especially when somebody have to go "on-site" to put a plug back in). All in all, users who got infected by a virus or worm were not able to use their network connection for a significant period of time. QNET has reduced this time, as illustrated by the graph above. Also, the primary response time (detection of infection until isolation), has been reduced to about 20 minutes on average (up to about 2 hours, worst case). Clearly, fast isolation of an infected system helps containment of a computer worm [3], and also reduces the potential threat that an infected system is abused by an attacker with ill intent (e.g., to "help" in a distributed denial of service attack to a third host; also see [4]).

Time spent in quarantine. The graph shows a cumulative distribution function of all (i.e., over 1300) computer systems that have been quarantined in the period May-December 2004. It shows that on average, 36% of the isolated systems is de-quarantined within 24 hours, and 50% within 48 hours (including weekends, holidays, etc.). Note that the time to restoration appears to have decreased, since the time of the widespread DCOM exploitations (bottom line in graph), compared to the more recent LSASS plague (top line in graph). This may be caused by users getting more and more used to QNET, and automatic de-quarantining (OneStrike) in particular.

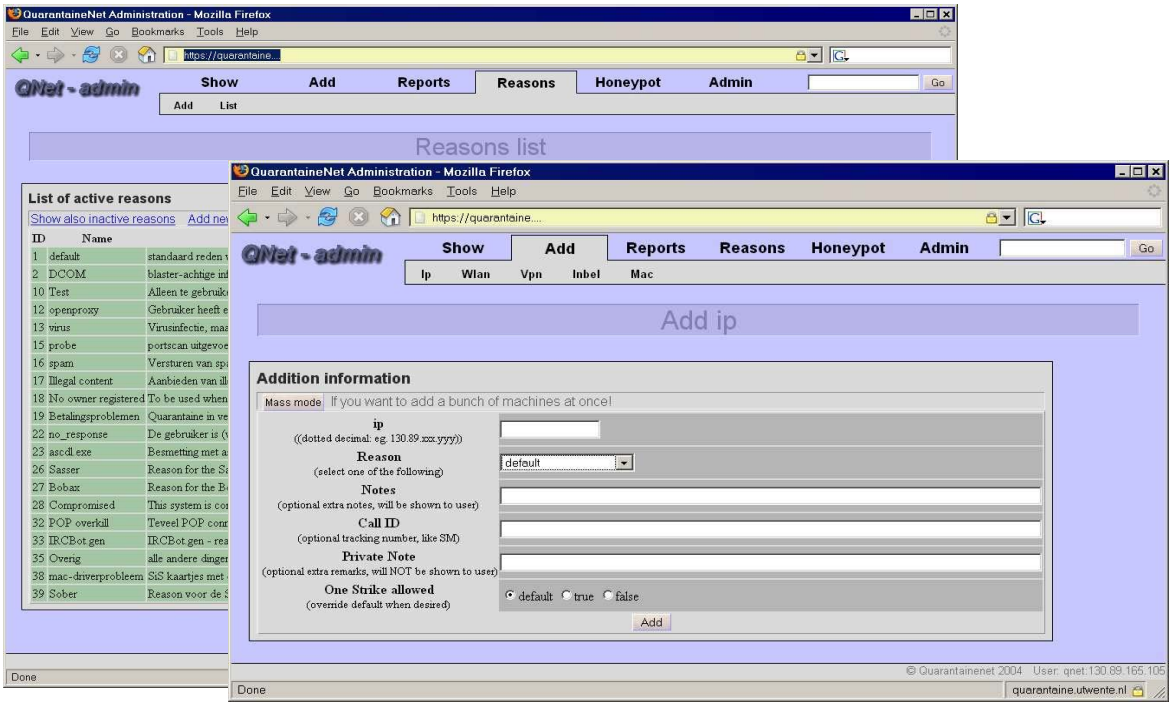
In quarantine "forever". Note that some systems that are put in quarantine, seem to never leave this isolation. Various reasons contribute to this phenomenon (which may look odd at first sight), e.g., people moving away from university, people buy a new network after they find their system quarantined, users who instead of registering a new MAC address, overwrite it with an (old) address that is already registered for them. Although these reasons may appear somewhat speculative, they all actually occurred.



The following observations can be made, since deploying QNET at the UT's network in February 2004:

- Over 3000 infections have been identified and led to quarantining. Note that this does not imply that over 30% of the approximately 10000 connected systems have been infected over the last 8 months: it is not uncommon that a single system is quarantined more than once through time, because of new infections.
- As indicated before, 36% of isolated systems were de-quarantined in the subsequent 24 hours, and 50% within 2 days.
- In almost 1100 cases, QNET has automatically de-quarantined a user, i.e., without intervention from or interaction with abuse staff, after the user has confirmed his system is clean again. One may have the policy to limit this *OneStrike* opportunity; in the UT's case, a user is allowed to de-quarantine himself without abuse staff interaction once every 6 months. Such policy discourages users to have connectivity restored before their system is cleaned from infections.
- As mentioned earlier, it is not uncommon for the honeypot to find new variants of known worms earlier than major antivirus companies are able to identify such new threats.
- Regardless of the increase in number of infections by new viruses or worms, as shown on previous visuals, the work load for the helpdesks has not increased since QNET has been deployed. Moreover, outbreaks of new threats do no longer cause problems and overtime for the abuse handling staff, since QNET takes care of quarantining infected systems, and in many cases also de-quarantining.

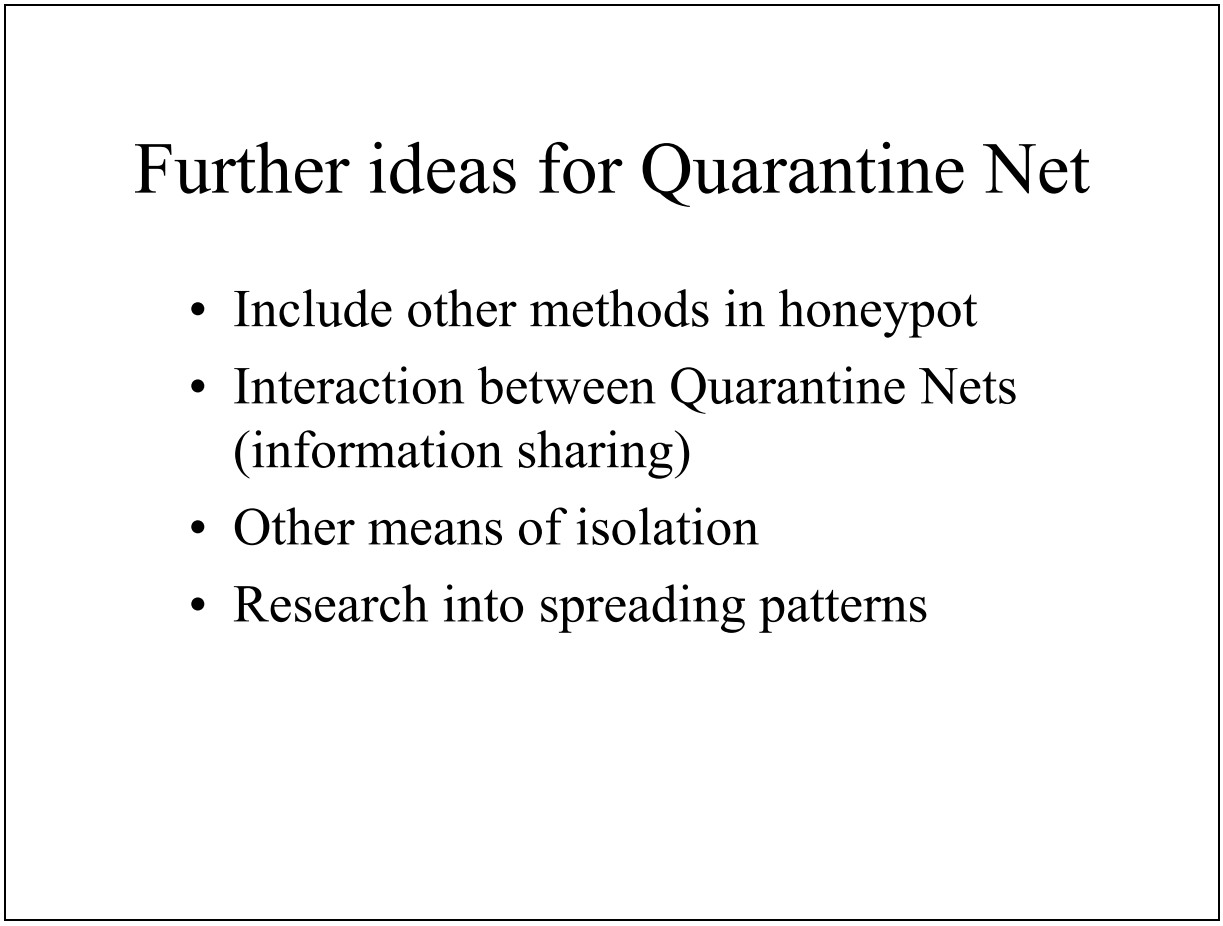
Operational results – management interface



Administrative interface. The "day-to-day management" of QNET can mostly be done through a web-interface. For example, authorized persons can manually (de-)quarantine computer systems, search for various properties, view statistics, etc.

List of active reasons. The screenshot in the back shows the list of possible grounds for quarantining, e.g., "DCOM" for Blaster-like infections, or "Illegal content" (typically used for manual (de-)quarantining) when a user is found to distribute illegal material. These so-called "reasons" are also used to personalize the webpage that is shown to a user who's system is quarantined.

Add IP. The screenshot in front shows the interface that may be used (by authorized persons only) to manually quarantine computer systems. After only given the IP address of the affected system (or, likewise, the MAC address or VPN username, etc.), QNET takes care of appropriately modifying the network configuration such that this system is indeed isolated.



Among the ideas for further work on QNET, are the following:

Other detection mechanisms. As already mentioned, detection of exploits for network services is not the only possible way to identify malicious activity on a network. Other possible means of detection include scanning outgoing email for spam and viruses, or analyzing network flow data for suspicious activity. Honeypots incorporating such techniques, as well as some others, are under construction at the moment.

Coupling of honeypots. If honeypot software would be running on various networks, they could be coupled to amass infection data from more than one source at a time. Not only could this increase the probability of catching malicious systems inside the protected networks, it could also be used to proactively protect all networks from external threats once these are encountered on one honeypot.

Other means of isolation. The actual isolation of malicious hosts at the network level is currently done by means of policy based routing. This, however, implies that all the hosts in quarantine are still able to connect to each other in some cases. A more solitary confinement of such systems would be preferable, so other techniques such as VLANs or very small IP ranges are currently looked into.

Research into epidemical spreading. When data from several honeypots on different networks becomes available, interesting research into the spreading characteristics of new worms or viruses could be initiated. If, for example, some networks appear more vulnerable than others, the characteristics of such networks could be determined, possibly enabling administrators to increase the security of their network.

Conclusions

- Quarantine net has been designed, developed and deployed at UT's network; more academic networks have shown interest
- The honeypot allows early detection of viruses and worms
- Abuse staff is not overloaded in times of new outbreaks
- The idea of quarantine net works well

Related work. A solution comparable to QNET, is Perfigo's CleanMachines product [5]. A difference between QNET and CleanMachines is that QNET employs a honeypot that *passively* detects infected computers, whereas CleanMachine scans for specific open ports on a computer that connects to the network. Another difference between QNET and CleanMachines is that QNET can be seen as a 'plugin' to the existing network infrastructure, whereas CleanMachines takes a more central role.

Concluding remarks. In this paper we have outlined the design and application of QNET, which limits the threats caused by computer viruses and worms to networks, as well as reducing helpdesk work load. It has also shown to be effective in communication with end-users.

The successful experiences with QNET at the UT have led to the foundation of a spinoff company [6], which supports QNET deployments, and further develops the concepts described in this paper.

References.

[1] Vern Paxson. Bro: A System for Detecting Network Intruders in Real-Time. In *Proceedings of the 7th USENIX Security Symposium*, January 1998.

[2] SNORT. The Open Source Network Intrusion Detection System. `http://www.snort.org/`.

[3] David Moore, Colleen Shannon, Geoffrey M. Voelker, and Stefan Savage. Internet Quarantine: Requirements for Containing Self-Propagating Code. In *Proceedings of IEEE INFOCOM*, April 2003.

[4] Stuart Staniford, Vern Paxson, and Nicholas Weaver. How to 0wn the Internet in Your Spare Time. In *Proceedings of the 11th USENIX Security Symposium*, pages 149–167, August 2002.

[5] Perfigo Inc. `http://www.perfigo.com/index_old.html`.

[6] Quarantainenet v.o.f. `http://www.quarantainenet.nl/index_uk.html`.