Why HP did not get "Blastered"

Richard Brown, Jonathan Griffin,
Andy Norman, Rich Smith
Trusted Systems Laboratory
HP Laboratories Bristol
HPL-2004-188
October 27, 2004*

E-mail: firstname.lastname@hp.com

active countermeasures, blaster, scanning, virus, worm

In August 2003 the IT industry was brought to its knees due to the release of a vicious worm called "Blaster" which cost the industry billions of dollars. However, HP was largely unaffected because of innovative technology produced jointly by HP Labs and Corporate IT that has been protecting our corporate network for over two years. In this paper we will explain how these new types of worms attack and spread, using Blaster as an example, and then describe the technology we have developed as a countermeasure to these attacks. In particular, we will provide details on the SETI@home-style distribution mechanism we have adopted which enables us to scan 400,000 connected devices every day, and also the payloads we deploy to keep our network safe.

* Internal Accession Date Only
Approved for External Publication

Why HP did not get "Blastered"

Richard Brown, Jonathan Griffin, Andy Norman, Rich Smith

OST/HP Labs

Richard.Brown@hp.com, Jonathan.Griffin@hp.com, Andy.Norman@hp.com, Richard.J.Smith@hp.com

Abstract

In August 2003 the IT industry was brought to its knees due to the release of a vicious worm called "Blaster" which cost the industry billions of dollars. However, HP was largely unaffected because of innovative technology produced jointly by HP Labs and Corporate IT that has been protecting our corporate network for over two years. In this paper we will explain how these new types of worms attack and spread, using Blaster as an example, and then describe the technology we have developed as a countermeasure to these attacks. In particular, we will provide details on the SETI@home-style distribution mechanism we have adopted which enables us to scan 400,000 connected devices every day, and also the payloads we deploy to keep our network safe.

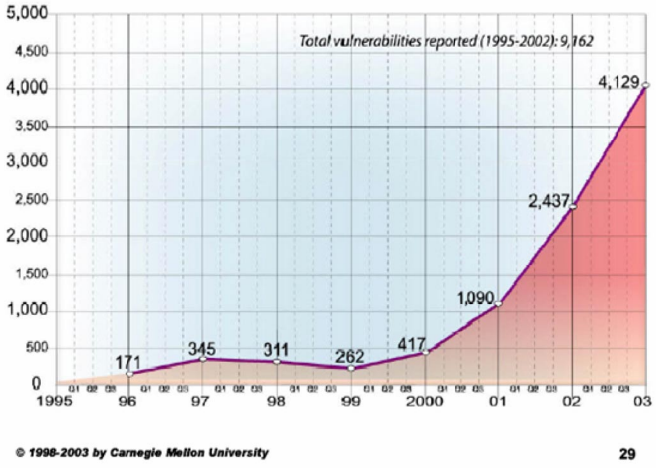
Introduction

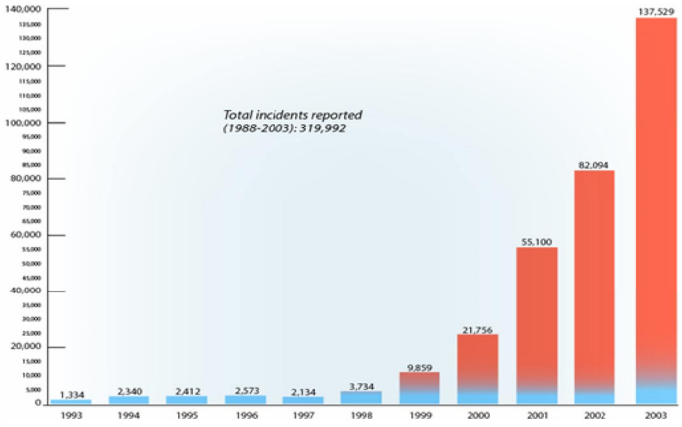
HP Labs, in collaboration with HP's Corporate IT have built a large scale, highly efficient distributed vulnerability scanner that injects re-engineered worm and viral code into machines on the network. This approach allows us to discover machines which are vulnerable to worm-like viruses with negligible false positive rate and also enables us to take immediate actions on those machines to mitigate the threat.

This paper will describe why we thought this was necessary and how we went about building the Active Countermeasures Solution. We will make several references to the worm Blaster which hit industry back in August 2003. The paper will move between a *story telling* style as we provide some history and context around the problem, and technical information concerning the solution itself.

Why a respectable company like HP should want to break into its own PCs

Barely a week passes now without headlines about new computer viruses or worms harming enterprises and individuals. These worms exploit the exponentially increasing number of vulnerabilities found in system software. Around 5000 new vulnerabilities were reported in 2003 (CERT/CC), any one of which could





potentially be used to attack a system. In fact, there were over 137,000 separate security incidents that same year. It's like protecting a building with 1,000 back doors built in to it. System security has to watch all of those doors, but to cause damage a hacker only needs to get through one of those doors by pushing it open slightly. In fact, it only requires a few infected machines to cause havoc on a network. For example, one machine infected with the Slammer worm could easily produce more than 100Mb/s of traffic, hence a small number of machines could saturate even a well provisioned network.

The damage caused by these attacks is very significant: network down time means you are not doing business; cleaning contaminated machines can take weeks and consume excessive amounts of people resource; loss of reputation can be hard to quantify but can sometimes be most significant cost.

Conventional techniques are failing [7] – modern worms spread so fast that the damage is done often before an administrator is even aware there is a problem. There are new variants appearing almost weekly, and the traditional process of testing patches, distributing them for users to install, and then searching for unmanaged machines is too slow. This is why HP decided to play the hacker game as well, i.e. break into our own systems using the same back doors as the worms. Rather than deposit a bad payload, however, we go in and make the systems safe.

Some history – how the project got started

Even though we are concentrating our discussion around the Blaster worm in this paper, it is worth noting that this idea was conceived well before Blaster, back in the summer of 2001 when CodeRed [8] hit industry. This was another worm based attack that exploited a vulnerability in IIS. It connects to a web server and makes a request that causes a stack overflow. This allows it to take over the web server, after which it starts trying to connect to web servers all over the place, spreading itself and filling the network with traffic

It was the timing of CodeRed that was most interesting. Because it hit during the summer at a time when many were on holiday, it was impossible to locate the owners of vulnerable machines in order to ensure their systems were patched. This meant that there were 10,000s of machines vulnerable to CodeRed connected to HP's network for weeks with nothing being done to remediate them. The only way now to protect HP from a CodeRed worm explosion was to get into those vulnerable machines automatically and make them safe. So work began to get hold of a disassembled copy of CodeRed II [9], work out what every instruction was doing and then produce a version which could break in, disable IIS, clean up the damage and leave a message for the users telling them what had happened. In addition, a scanner had to be written that would use this modified CodeRed worm to identify the vulnerable machines. All in all, it took a small team of HP Labs engineers one week to develop and deploy the code. Since then, in joint work with Corporate IT, it has evolved into a critical piece of HP's internal incident management program for protecting against the most serious of worm attacks, and is now called HP's Active Countermeasures Service, or ACS [6].

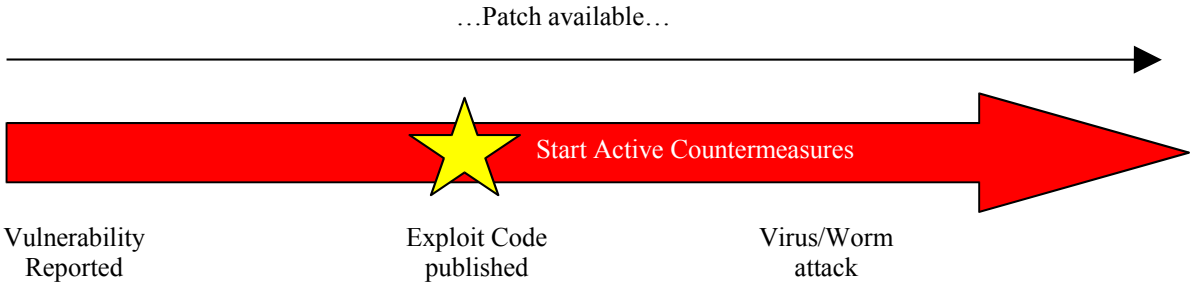
So let's return now to our story around Blaster.

How Blaster attacked the IT industry

Blaster [1, 12] appeared a few weeks after exploit code for a vulnerability in Microsoft's DCOM/RPC [10] service was made public on the Internet. Blaster took advantage of a buffer overflow in DCOM on Windows 2000 and XP. A suitably crafted set of exploit bytes were sent to TCP port 135 on a target machine. If the machine was vulnerable then a server running a command shell was created, listening on TCP port 4444. Blaster then connected to the new server, and sent two commands to be executed. The first invoked a TFTP client which connected back to the attacking machine (which had itself spawned a local TFTP server). A copy of blaster was then transferred to the vulnerable machine via TFTP. The second command caused the transferred copy of blaster to start running on the vulnerable machine – and hence the spread.

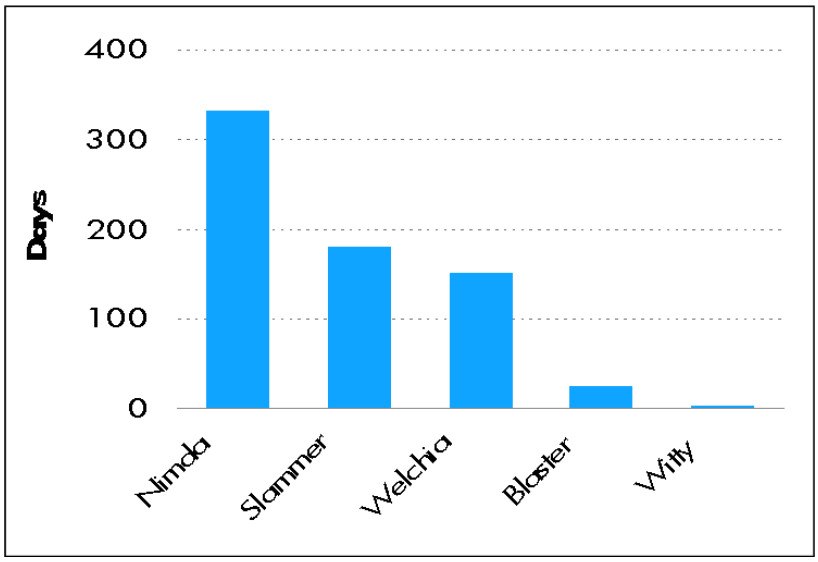
The visible effects of Blaster were a huge increase in network traffic, plus a distributed denial of service attack on the Microsoft windows update site.

Blaster is typical of many of these worm-like threats. The time works roughly as follows:



Typically, a vulnerability is reported in the OS or network applications such as SQL servers or web servers, either by the company that developed the software or white hat hackers. It is usually not long before someone has explored this vulnerability and has posted code on the internet that exploits it. Based on this exploit code, hackers implement their worms and viruses and launch the attack. At any point along this timeline, patches are made available to fix these vulnerabilities. However, in many cases these patches occur too late, after the attack has already taken place, or else they don't get installed on the user's machines due to inconvenient timing or downright apathy.

This essentially is a race to close down the vulnerability before the attack happens. However, the challenge is getting harder due to the decreasing time between the publishing of exploit code and the first attack being launched leaving companies insufficient time to deploy patches, even when they exist.



Hence, the Active Countermeasures System attempts to provide protection to an enterprise network between the time a vulnerability exploit is published, and the time the full patch has been developed to permanently fix that vulnerability. So, for threats which are considered to have potentially significant business impact, as soon as exploit code is published, experts within HP develop a vaccine payload and then combine this with the exploit code to remediate as many machines as fast as possible before the attack – the exploit code gaining access to the vulnerable system and the payload performing the remediation. Note, the vulnerable systems will still require patching when the patches are available, but for now the systems have been made safe. Our experience has shown that this is much faster than the traditional patch management only process, which fails badly when there are a number of unmanaged and unknown machines connected to the network.

How HP responded to Blaster

The DCOM/RPC vulnerability used by Blaster had been known in the security community for some time. When the exploit code was made public, HP's security folks knew that it wouldn't be very long before the first worm to take advantage of that code would be written.

So, on top of the usual global e-mail messages instructing people to patch their systems, HP raced against time to re-use the exploit code to deliver its own custom payload to each vulnerable machine.

Our countermeasure code took a very similar approach the the actual Blaster worm., the significant difference being that the payload we transferred via TFTP to the vulnerable machine was *not* a worm, but remediation code. This code performed different levels of remediation depending on whether it was the first time this particular machine had been seen.

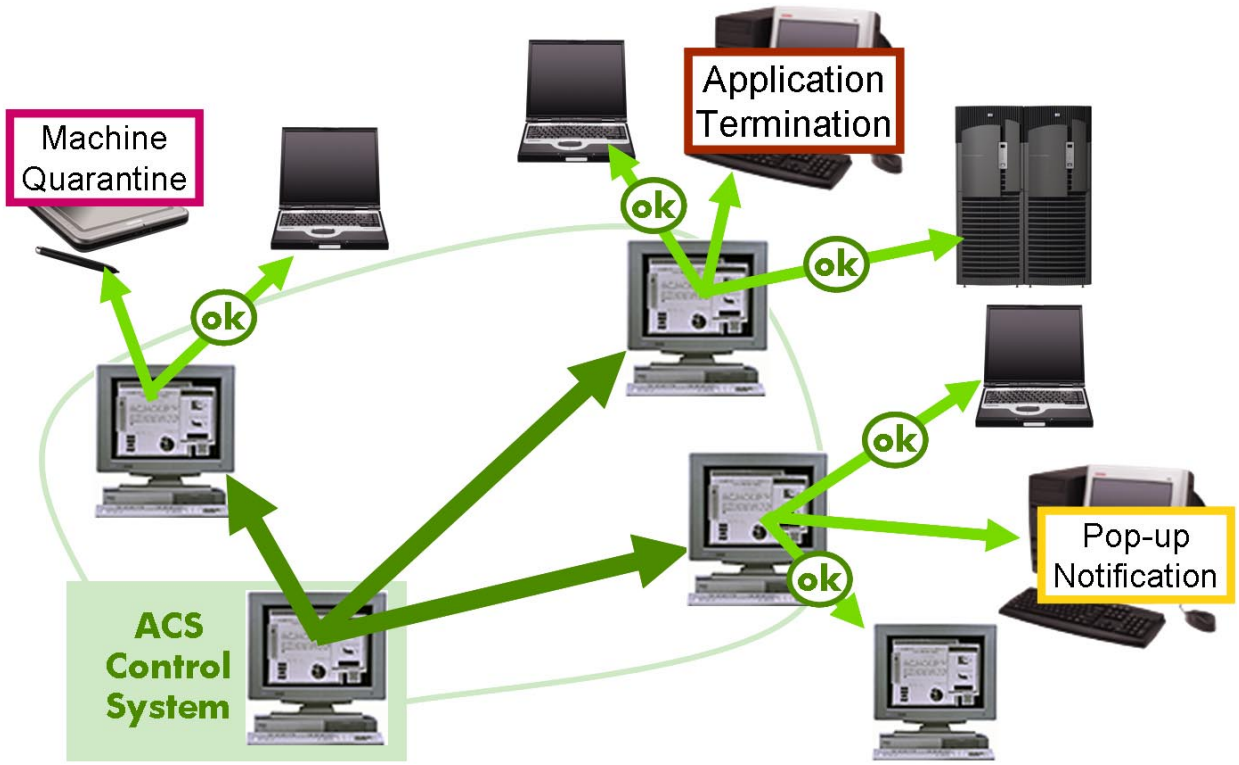
The first time we exploited a machine, any logged-on user was notified by a pop-up message that the machine was vulnerable and given a URL of where to find more information on how to patch and fix the problem. As a way of temporarily making the machine safe, the DCOM service was also stopped. Finally, boot.ini was re-written to make sure that the machine would not reboot automatically, but required manual intervention.

Subsequently, if a user ignored the warnings and manually rebooted the machine without peforming any patching, the next time the machine was exploited, it was immediately taken off the network by halting it.

Thus, before Blaster hit the world, HP had patched or disabled a huge number of their machines.

HP's Active Countermeasures Service

So having explained what problem we are trying to solve and how HP responded to this during the time of the Blaster crisis, let's look now in more detail at how this vulnerability scanner and remediation payload system actually works.



To do this we consider a simplified enterprise network. Consider that in this environment, some of the machines are known and some are not. The ACS control system is communicating with a set of other machines scattered throughout the enterprise's infrastructure to deploy the countermeasure. Together this set of control machines is our distributed scanning and deployment tool. Each of these worker machines has been assigned a sub-set of the enterprise IP address space. As new vulnerabilities are assessed and new exploits are discovered, new payloads are created for these distributed scanning machines to test for machines in their portion of the network that are vulnerable to the attack.

The worker machines are scanning and delivering a corrective action. This action is determined by enterprise policies – it could be a variety of things. In a mission critical environment it could be as simple as a pop-up window informing the operator of the machine that it is vulnerable to a specific attack and that they need to engage the patch management system at the earliest opportunity. Or, in certain contexts, corporate policy may determine that the proper response is to terminate the vulnerable application. For example, if the attack is targeting a web server, the correct response for a certain class of machines is to halt the web server until the machine has been patched. In other contexts, for example mobile computers that employees take home then bring back into the network, the chosen response may simply be to quarantine the machine until it has been upgraded with the appropriate set of patches. The decision of the corrective action remains with the enterprise – and the policy choice depends on the characteristics of the organization. But again, this capability allows administrators to exercise these policy choices over a wider range of machines – including those that they don't know are connected to the network.

SETI@home - How to scan 400,000 devices in a day

The distributed scanning model in ACS was inspired by work done by the distributed.net and SETI@home [5] teams in the past.

With both of the above projects, a volunteer would allow the spare CPU cycles on their machine to be used for performing one small slice of a huge computational task. As machines joined the team they were given the code to execute and the data that needed to be processed. Any significant results from running the code on the data were periodically sent back to the central machine(s).

The model we use in ACS is similar. Scanning code is given out to each volunteer scanning client, along with a number of IP addresses "near" to the client that have not been scanned recently. Once the client has scanned a block of IP addresses, all results generated are sent back to the central server where they can be further analysed and processed for viewing.

Results could be as simple as a given host was not vulnerable to a particular worm attack, or as complicated as letting the server know that a host being scanned was infected, that the logged-on user was notified for the 2nd time and that the host was subsequently halted, after obtaining information such as the netbios name, the workgroup name and the MAC address(es) of the interface cards for more accurate identification of that particular machine at a future time.

The ACS server works in a constant cycle. It knows which IP addresses it wishes to be scanned, hands them out along with scanning code to registered clients, collects the results, hands out more IP addresses etc. If a particular client does not give back results within a given time period, the IP addresses it was given are put back into the unscanned pool and given out to a future suitable client. No addresses are left completely un-scanned simply because a client has gone off-line for any reason.

If the scanning cycle is to be speeded up for any reason, more clients can be added to the team at any time, and then leave when not needed. The process of adding and joining the team is very lighweight... as long as the client is suitably authorized and authenticated with the server, it can join in or leave at any time.

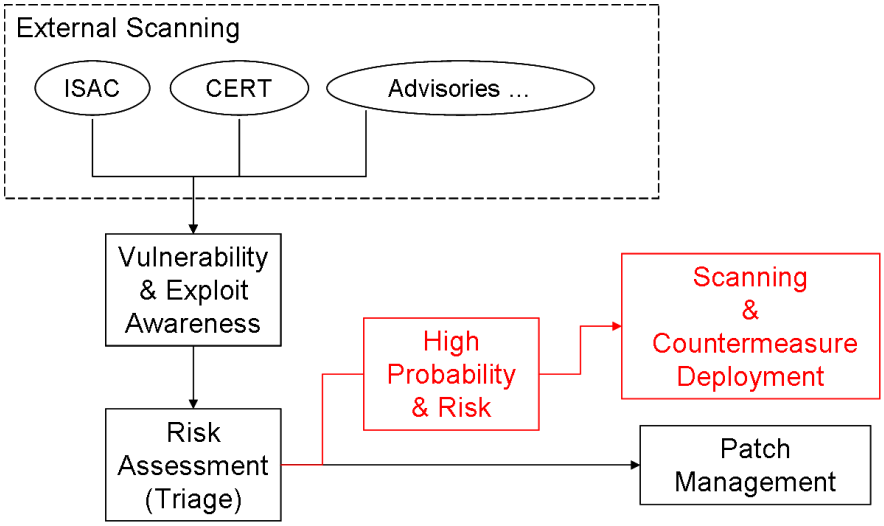
Scanning code handed out to each scanning client is in the form of a dynamically-loadable shared library. When a client has finished a batch of addresses, it contacts the server and asks whether a different library is needed. If so, it downloads that from the server to replace the existing library before working on the next set of addresses. What this means in practice is that should the scanning library be updated (for example, to detect a new

vulnerability, or to perform a new remediation), the clients will all be running the new code within minutes, reporting new results back to the server.

Do I only need the Active Countermeasures Servce?

Many vulnerabilities that are reported are of low importance and should not require something as sophisticated as Active Countermeasures. It should also be reiterated here that the Active Countermeasures system doesn't actually patch the machines, it simply stops them from being a threat to the entire network. Hence it is still necessary to complement the countermeasures system with a more complete incident management program.

HP's approach to managing such threats is as follows:



There is a team of experts who observe and scan for known vulnerabilities, new exploits, new dangers in the environment. They continously gather information from the ISACs, the CERTs, from hacker bulleting boards – collecting knowledge about what are the exploits and what are the vulnerabilities in the world. From this, there is a process of triage – determining the effect and the likely impact on business should this vulnerability be exploited, then deciding on the proper response. For many of these attacks the effect is low consequence, low probability and the right response is to cope with it in the patch management system, to be dealt with over time. It's that small percentage of incidents that are high risk, and have a high probability of occurrence – for which we analyze and deploy our active countermeasures payload. It's the speed of response which is really key here, and this is where many traditional approaches fail.

Only 20,000 more machines to go – but where are they?

Whilst a traditional approach of forcing users or administrators to patch machines under their control might reach 95% of your corporate network, what do you do with the remaining 5% (20,000 machines in HP's case), that no-one takes responsibility for, many of which are old and hidden under tables in machine rooms. This in fact is a major problem as it is often these machines that have not been kept up to date that become the main launch pads for further attacks. Using the active countermeasures approach, a payload can be created to break into the machine and take it off the network all together.

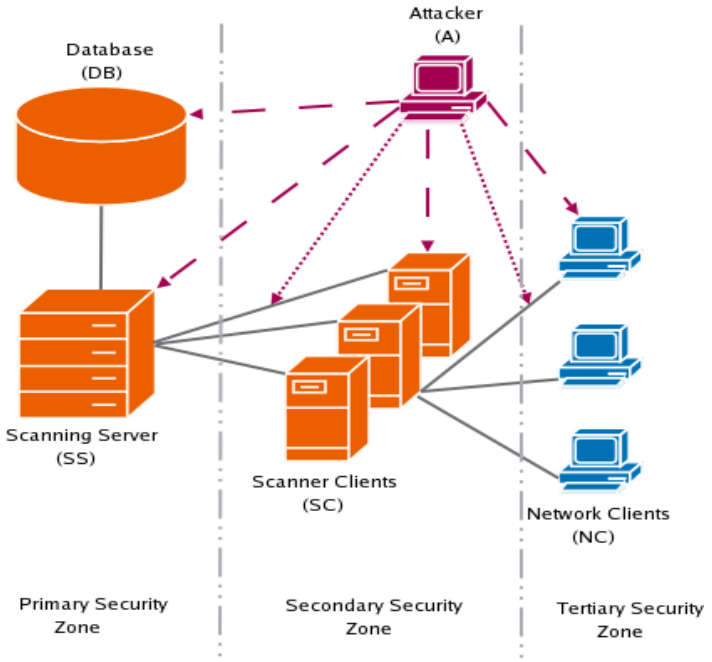
But my machine really isn't vulnerable – honest!

A significant feature about this technology is in respect to false positives, i.e. a system is shown to be vulnerable when in fact it isn't. Using the approach of actually breaking into a system through the same back door as an attacker produces zero false positives. It is really indisputable, even though our experience has shown that people

will still try to claim their machines are safe. We have examples where we have had to penetrate a machine and produce a directory listing of the hard drive in order to prove to folks that their machine is indeed vulnerable

Can this approach be compromised and used as a means of attack?

The Active Countermeasures System clearly has the potential for an attacker to try and use it to launch a more damaging attack than they previously could. Such a risk is obviously unacceptable and places strong requirement on the countermeasures systems security. The goal of the security applied to the system is to ensure that the effort required to utilize the countermeasures framework as an attack vector is much greater than for a conventional attack vector that would achieve the same result. In other words, the countermeasures system should not provide an attacker with a more effective attack vector, nor should place the network at higher risk than it would be without it being present.



We have performed a detailed threat analysis of the system. Traditional authentication and secure channels ensure that the primary and secondary security zones shown in the diagram above are safe. However, the main area of vulnerability seems be between the scanners and the end client systems. This is because the current implementation uses a two stage payload: the first stage executes the exploit to prove a system is vulnerable, and once inside a system, it pulls a second payload from the scanner to do the actual remediation.

This approach has the advantage of being very flexible in that one can deploy a range of payloads iteratively for the same exploit scan. However, it does introduce a new vulnerability: as there is no authentication, confidentiality or integrity assurance in the tertiary zone (i.e. the scanner attempts to break into the machine), an attacker is able to target the link between the scanner and the client. An attacker may be able to target a known subset of the data (e.g. TFTP calls) traveling between a scanner and a client and act as a man-in-the-middle (proxy) altering payloads. An attacker can then return a secondary payload containing malicious code to the end client which will be subsequently executed. The issue is significant as an attacker no longer has to search for vulnerable machines, they will present themselves by requesting a secondary payload from the scanner.

We believe this threat can be overcome by redesigning the payloads so that the remediation part is included as part of the initial access to the system, i.e. remove where possible all two stage payloads. The reason for highlighting this case to show how easy it is to introduce new threats into a system when attempting to fix existing ones.

Welchia – definitely not the way to do it!

At this point it's probably worth saying something about other attempts at tacking the problem of worms, one of which in particular ended in disaster. We refer in this case to the Welchia [11] worm that was meant to be benevolent, scanning the network for vulnerable machines and putting things right. Specifically, it attempted to download the DCOM/RPC patch from MS Windows Update website, install it and then restart the computer. It checked for active machines to infect by sending ICMP echo requests or PING packets, which resulted in increased ICMP traffic. There were two things wrong with this: Firstly, applying patches automatically is fraught with difficulty, since it is hard to be sure what specific patch levels are required for certain critical applications. Secondly, this was indeed a worm; it spread very unpredictably and when things started to go wrong, it was impossible to contain. In fact, this so-called good worm is now causing the industry more problems than some of its more malicious counterparts. Bruce Schneier, a well respected expert in IT security had this to say about Welchia:

> *Patching other people's machines without annoying them is good; patching other people's machines without their consent is not. A worm is not "bad" or "good" depending on its payload. Viral propagation mechanisms are inherently bad, and giving them beneficial payloads doesn't make things better. A worm is no tool for any rational network administrator, regardless of intent*.

So, rather than create countermeasures technology that behaved exactly like a worm, we chose to build the distributed scanner based on the SETI@home approach described earlier.

What next

Currently HP Labs is committed to working with HP Services to roll the Active Countermeasures Service out to customers. This might not sound particularly challenging but in fact there are a number of unanswered questions that need to be addressed if HP are to make money from this. For example, how do new payloads get written, who writes them, how do we set up service level agreements, what is HP's liability exposure, what is the business model…? We have to remember that the existing Active Countermeasures system has evolved into what it is today through close collaboration with HP's own Corporate IT. As a result we have not followed through detailed and structured processes for setting such a system up. This is exactly what would be required for delivering such a service to customers. Hence, HP Labs with HPS are currently seeking out pilot customers with whom we can develop these approaches and answer some of the critical questions.

Of course, HP Labs has more to offer than just the countermeasures. We have also been working on a system called Virus Throttling [4]. If we assume that sooner or later, some machines will get infected no matter how good one's security defenses are. Then the challenge becomes how to ensure that infection spreads no further. The virus throttling technology slows down the number of outgoing connections to new machines, e.g. to one new connection per second. We have shown that for normal use there is no noticeable difference in performance, since users normally only require a few connections (mail server, web proxy, file server…). However, a worm by its very nature will be attempting to make lots of new connections in order to spread itself. The virus throttle is able to detect this behavior of the worm and stop if before it's able to spread any further.

We believe the combination of vulnerability scanning and virus throttling is a compelling offering to customers, so we are actively working to transfer the virus throttling technology to other parts of HP in order to offer the complete package.

In terms of future technology directions, we want to continue to explore biologically inspired metaphors for approaching viral attacks, and also look at different types of attacks. For example, new forms of stealth attacks [2, 3] could become the next wave of challenge as virus writers work hard to overcome existing countermeasures schemes – an obvious next step for attackers to take, in their efforts to avoid detection. Fast, noisy propagation strategies can incite a speedy response from vendors and consumers alike. A slower, quieter approach would have the benefits of going unnoticed for much longer, perhaps until a critical mass of infection; by which time it would be too late for effective mitigation.

Summary

The Distributed Scanner enables us to make HP's network safe against critical vulnerabilities before a worm or virus breaks in. In particular:

- It gives IT staff confidence because of the very low false positive rate
- It provides coverage: it scans whole address ranges and finds unmanaged as well as managed machines.
- By breaking into vulnerable machines it allows immediate mitigation of the threat: IT staff don't have to wait to find the owner or physical location of the vulnerable machine.
- SETI@home style scanning is remarkably efficient: a handful of scanning servers distributed around the HP regions can scan HP's entire network in less than 24 hours.

References

1. eEye Digital Security, 'Blaster Worm Analysis', *http://www.eeye.com/html/Research/Advisories/AL20030811.html*, Nov. 2003.
2. Thomas H Ptacek & Timothy N. Newsham, 'Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection', *http://www.securityfocus.com/library/745*, Jan 1998.
3. Stuart Staniford, James A. Hoagland, Joseph M. McAlerney, 'Practical automated detection of stealthy portscans', Journal of Computer Security, Vol. 10, Issue 1-2, Pp.105-136, 2002.
4. Matthew M. Williamson, 'Throttling Viruses: Restricting propagation to defeat malicious mobile code', ACSAC 2002, Las Vegas.
5. SETI@home, *http://setiathome.ssl.berkeley.edu/*
6. Andy Norman & Matthew Williamson, 'Hitting back at Code Red', HP Labs Technical Report HPL-2002-111, *http://www.hpl.hp.com/techreports/2002/HPL-2002-111.html*.
7. David Moore *et al.*, 'Inside the Slammer Worm', IEEE Security and Privacy, Vol.1 No. 4, Pp.33-39, *http://www.computer.org/security/v1n4/j4wea.htm?SMSESSION=NO*, July-Aug. 2003.
8. 'CAIDA Analysis of Code-Red', *http://www.caida.org/analysis/security/code-red/*, CAIDA Organization, (last updated Apr. 2003).
9. eEye Digital Security, 'CodeRedII Worm Analysis', *http://www.eeye.com/html/Research/Advisories/AL20010804.html,* Aug. 2001.
10. LSD, 'Critical security vulnerability in Microsoft Operating Systems', bugtraq, *http://marc.theaimsgroup.com/?l=bugtraq&m=105838687731618&w=2*, 16 July 2003.
11. Sophos plc., 'W32/Nachi-A', *http://www.sophos.com/virusinfo/analyses/w32nachia.html*, Aug. 2003.
12. F-Secure Corporation, 'F-Secure Virus Descriptions : Lovsan', *http://www.f-secure.com/v-descs/msblast.shtml*, Sept. 2003