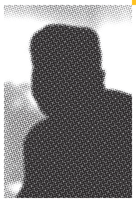
STEVE MANZUIK

your defense is offensive



Steve Manzuik is the founder and moderator of Vulnwatch (www.vulnwatch.org). He is currently a product manager for eEye Digital Security, and over his 17-year career has worked for Ernst & Young, IBM, and Bindview Razor.

smanzuik@eeye.com

TODAY, INFORMATION TECHNOLOGY is flooded with various gadgets and products that are all marketed to help improve security. Some of these products do work as advertised while others do not. One would assume that the security products themselves are secure, but the reality is that some security products may in fact be more of a danger to your networks than a benefit. This article outlines some of the known vulnerabilities in security products as well as some new attack vectors that may not have been considered. In doing so, the intent is not to call attention to specific vendors, but the reader may notice that some vendors have more issues than others.

In the Beginning

By searching the Open Source Vulnerability Database (OSVDB) for the keyword “security,” one finds security problems dating back to 1996: OSVDB ID 6519—IPFW address:mask syntax firewall filter leak.

While this flaw does not lead to the use of IPFW as an attack vector, it does show that security flaws within security products have existed for quite some time. Lets look at some of the newer, more serious issues out there.

More Relevant Issues

I’ll start with a security technology that everyone has been convinced that they need: firewalls. When you search online vulnerability databases such as OSVDB (www.osvdb.org) using “firewall,” you find approximately 160 different vulnerabilities. Of course not all of these are serious enough to be used as an attack vector, but some are.

The first vulnerability is OSVDB ID 4412—Checkpoint Firewall-1 SmartDashboard Overflow. This vulnerability allows a remotely authenticated user to elevate her privileges and execute arbitrary commands. This issue is over a year old and, according to OSVDB, there are no known patches or workarounds for it, but exploit code does exist. The level of exposure to a vulnerability such as this is limited, as you do need to be an authenticated user, which would lead one to assume that various log files will offer evidence of your dirty deeds; of course, those log files are only good if stored in a secure manner and if the evil user doesn't know to cover up the evidence.

Obviously, allowing code to be executed on your firewall is a bad thing, and the scenarios where this can be abused are endless.

Also under the category of abusing firewalls we find a handful of Zone Alarm vulnerabilities. According to OSVDB there are 16 different ways one could abuse Zone Alarm. However, out of those 16 vulnerabilities only two offer the ability to be used in an attack other than your run-of-the-mill denial of service. The first of these was discovered by eEye Digital Security and is an overflow that was present in the SMTP processing agent. Those of you familiar with this vulnerability are probably getting ready to correct me by saying that this is not remotely exploitable, since it requires the malformed SMTP `RCPT TO` string to come from the client. This is only partially correct: yes, the malformed command must come from the client side, but that does not rule out the potential to abuse this flaw.

For example, a malicious Web link could easily be crafted and used to trick users into sending an email with the correct malicious string, causing the system either to crash or, even better from an attacker's perspective, to execute commands with system privileges. Probably the easiest and most reliable attack to initiate here, not that I would know anything about attacking systems, would be to upload and execute something, netcat perhaps, that could give you system-level shell or back-door access to the system. Simply executing netcat to listen on port 53 or some other nonobvious port is pretty common, and, as far as I know, netcat is not on any of the anti-virus vendor hit lists.

The second vulnerability in Zone Alarm is very similar, albeit more difficult to exploit, since it requires abusing a specific device driver installed with Zone Alarm. Again, this needs to be achieved on the client side. This vulnerability is more likely to lead to crashing the system than to successfully executed commands, but the potential for abuse is there and it does work.

To broaden this article beyond a discussion of broken firewalls, which would ultimately lead to a rant that might be construed as antifirewall technology, I will move on to another popular security technology: intrusion detection systems (IDS).

IDS

As you probably know, there are host-based (HIDS) and network-based (NIDS) intrusion detection systems; for the purposes of this article, I will make most NIDS and HIDS vendors cringe by simply lumping them together.

As my first example, I will use the Snort vulnerability that was found just over two years ago. I have personally witnessed environments still running older versions of Snort, which is why I am using it as an example. Basically, an attacker can send a specially crafted packet that will cause a heap overflow and execute commands.

Let's think about this one for a minute. Here you have a system that is typically installed on sensitive network segments and is typically in a position to see most network traffic. Rather than just abusing the sensitivity of this system, a smart attacker would use this vulnerability to gather data. For example, by using this vulnerability to obtain a remote shell, one could capture sensitive information, since that is essentially what NIDS is already doing.

Luckily, this vulnerability has been fixed and is not present on any new versions of Snort. My next example, for those good at noticing patterns, was discovered by eEye Digital Security.

OSVDB ID 4702 explains how a flaw in the way RealSecure, Preventia, and BlackIce reassemble SMB packets can be abused to run arbitrary code with system privileges. This vulnerability can be exploited with one simple SMB packet; to quote the eEye advisory, "code execution is effortless."

Once again, we have a system that is typically used in sensitive locations, offering an attacker access to sensitive data. Much like the Snort attack, a smart attacker would not abuse this flaw but silently use it to gather sensitive data. In fact, by combining this with the numerous ways an attacker could bypass IDS signatures, the attacker could easily gain and maintain access to the system. A careful attacker could pull this off without being detected.

Another example of the same IDS products leaving organizations open to attack is the ICQ Protocol overflow that was used in the Witty worm. The Witty worm is a great example of how systems designed to protect you can leave you vulnerable.

All of the above examples have been known for quite some time. Most of them have been addressed by the vendors, although that does not mean that there are no longer any systems that can be attacked using these methods. The following scenarios are ones that may or may not have been thought of or reported but that do illustrate how the very infrastructure we have built to protect our networks can in fact be used against us.

Patch and Systems Management

At Blackhat Amsterdam 2005, Chris Farrow presented a talk that I wrote and researched, outlining many of the flaws in the current patch and systems management process and including some ideas on how these flaws can be abused. Most of these attack scenarios are completely theoretical, and while variations on or portions of the attacks have occurred, no one has performed an attack against the patching or systems management infrastructure . . . *yet.*

During the Blackhat talk, the following disclaimer was given: No vendor products will be named unless information is already public; most of these flaws apply to multiple vendors; and any vendor-specific issues will be disclosed to the vendors before they are publicly disclosed.

The patching of workstations and servers has become a necessity in maintaining system security and uptime. Almost every organization today finds itself forced to install some sort of patch on an almost monthly basis. This has become a very expensive problem for organizations, and many vendors have gladly stepped up to the plate with various solutions designed to help manage the patching and configuration of systems, thus helping to increase the overall security of an organization. Too bad many of these vendors did not consider the impact of their very own products on an organization's security.

Before we dive into the specific flaws in the various products let's look at the anatomy of a Microsoft patch. Patches released by Microsoft are digitally signed, these signatures are available on the patch download server, and Microsoft controls the patch process with an XML file named mssecure.xml. As a bare minimum this seems like a reasonable way to handle patch distribution, but can you think of any vendors out there that do not even bother to do this bare minimum?

Patch and systems management products can be lumped into two categories: agent-based and agentless technologies. Obviously, to be managed or patched the agent-based systems require an agent to be installed on the host systems. The agentless systems do not require an agent but usually require privileged credentials or some sort of trust relationship on the network.

Most of these systems communicate over the following protocols: HTTP, RPC/DCOM.

You will notice that these do not include any protocols that are natively "secure" by means of encryption. Some products do have an option to run over HTTPS while others do encrypt agent-to-master communications, but most do not.

Patches are fed to the master systems directly via Windows Update over HTTP. Some systems download new patches for each run; others store the patches in a central repository. Some systems are able to simply push out patches and configuration changes; others require custom scripts to be created.

So far, while explaining at a very high level how most vendors have approached patch and systems management, I have already uncovered some potential flaws that should be investigated. They are:

- Lack of encrypted communications between the console and the agents
- Lack of true authentication (some products) between the agent and the console
- Patch repository as a potential attack vector

LACK OF ENCRYPTED COMMUNICATIONS

Some vendors do not encrypt communications via the agent and the console. This leaves them open to replay and man-in-the-middle attacks. Imagine a sophisticated attacker placing himself between your servers and the system that has the ability to alter the configuration of those systems.

LACK OF TRUE AUTHENTICATION

During testing and research for the original Blackhat talk, I found that some agents only check that the machine name sending the commands matches that of the console, and then simply do what they are told by this machine. Obviously, this is a huge mistake in terms of security, since an attacker can easily not only discover the machine name but also spoof it.

PATCH REPOSITORY AS ATTACK VECTOR

The patch repositories on some systems have, by default, extremely weak directory and file permissions, leaving the patches themselves vulnerable to modification. While some systems combat this by checking the digital signature each time they issue a patch, others simply check once, save the patch in the directory, and then assume the patch is valid so long as the file name matches. These are all weaknesses in the patch products themselves; as with any weakness, however, these can be exploited only if there are attack vectors designed to take advantage of them.

Attack Scenarios

The first potential attack vector comes from an internal threat (admittedly not as sexy or cool as an external attack):

- Compromise the patch repository
    - The attacker gains access to the central patch repository, modifies a patch to install malicious code, and waits for that patch to be rolled out.
    - The attacker gains access to the central patch repository, follows the numbering sequence of vendor patches, and places a malicious patch with the next patch name, knowing that the system will not overwrite what is in the repository.
- Sniff internal network for agent-to-console communications or console-to-system communications
    - Look for credentials as they will have privileged access.
    - Watch for specific commands to figure out and document what the agent will respond to and how it will respond.
- Man-in-the-middle the system and substitute the payload with malicious code
    - Adjust patch targeting to prevent a patch from being installed, leaving a system vulnerable. This would work only if the agent does not report patch success/failure back to the console, though such traffic can also be modified or created.

An internal malicious user could pull off one of the above attacks undetected as long as the user is smart enough to learn exactly how the system works and what inputs and outputs it expects. System administrators explicitly trust what they see on their consoles and do not have the time or reason to double-check the system.

The second attack scenario involves an external attack and abuses a couple of flaws found specifically in the Microsoft patch process. The first issue with this process is that it is regularly scheduled. This gives an attacker a window of opportunity—more on this shortly. The second issue is that while Microsoft publishes an XML file containing everything a user needs to validate a patch, that XML is distributed from the same systems that the patches are distributed from. You will see why this can be a problem in the following scenario. The attacker:

- creates a trojan patch, digitally signs it, and creates the proper XML file that some systems will look for;
- patiently waits for the next "Patch Tuesday";
- goes after the infrastructure of the target;
- redirects requests for known patch sites to a site containing spoofed patches.

The system will receive what it believes to be a valid XML file and then begin to download the executables. Your base will then belong to the attacker.

The trojan patch could address the actual problem and simply install its own additional code. And it could be digitally signed, obviously not with Microsoft's key but with another. Many patch management systems only check that there is a signature and do not actually validate that signature.

Solutions and Conclusion

The scenarios outlined above are based on nothing more than high-level research of vulnerabilities and how specific products work. While products that run and "secure" Microsoft environments were used, in all of the examples these flaws can extend to other operating systems.

The bottom line and the entire point of this article is that organizations need to start putting more thought and research into what products they use to protect their infrastructures. Basing purchasing decisions on who has the cutest booth babes at the various conferences may make sense for general IT products and services but not when selecting a security or systems management vendor.