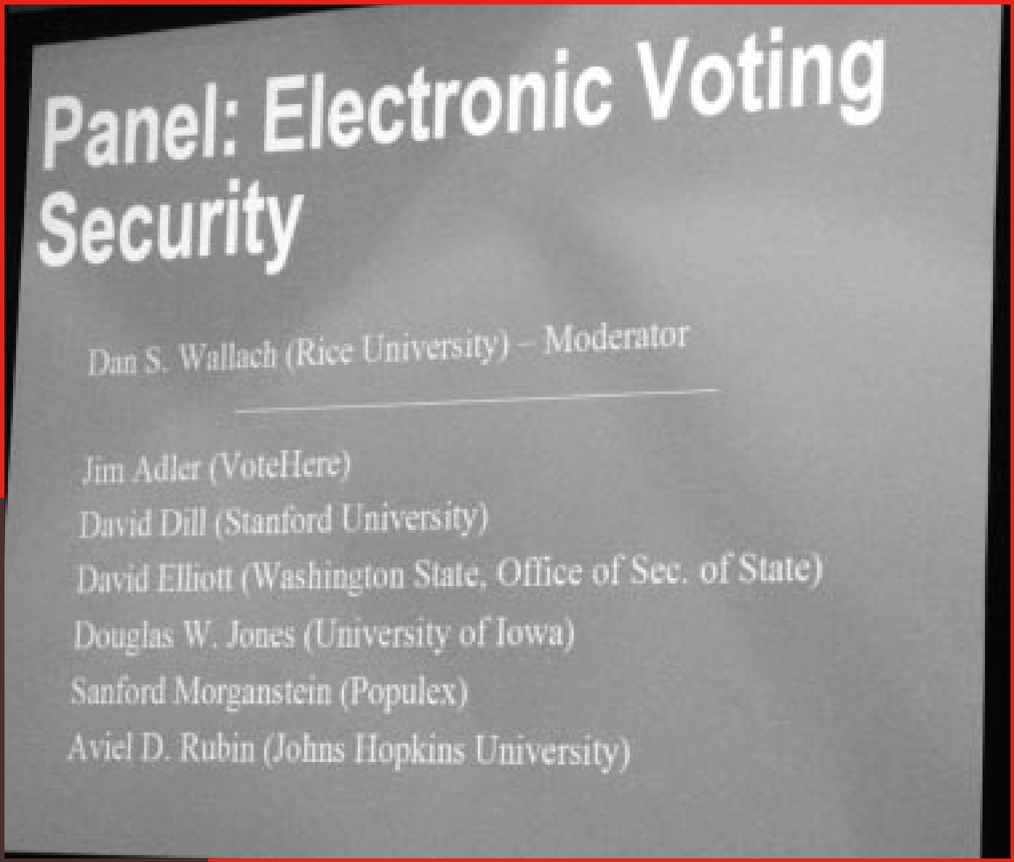
;login:

THE USENIX MAGAZINE

December 2003 • volume 28 • number 6



Panel: Electronic Voting Security

Dan S. Wallach (Rice University) – Moderator

Jim Adler (VoteHere)
David Dill (Stanford University)
David Elliott (Washington State, Office of Sec. of State)
Douglas W. Jones (University of Iowa)
Sanford Morganstein (Populex)
Aviel D. Rubin (Johns Hopkins University)

inside:

Focus Issue: Security

Guest Editor: Rik Farrow

USENIX

The Advanced Computing Systems Association

how are we going to patch all these boxes?

by Ryan Russell

Ryan Russell is an independent security expert, author, and speaker. Presently, he is contracting at BigFix, Inc. where he is helping them expand their UNIX support.

ryan@thievco.com

Chances are excellent that I don't need to point out to you that you've been beset by worms. If you have a sizable network and run Microsoft Windows, you have probably seen the front lines of the worm battle. If you run a Microsoft-free environment, you at least get to see the fallout in terms of constant probes and email-borne malicious code (or bounces, when the worm decides to be your email address for the day).

Are worms strictly a Windows problem? Technically, no. There have been worms for most of the major operating systems, and that includes most of the UNIX-like flavors. For all practical purposes right now, however, worms are a Windows problem. I happen to be one of those people who thinks that this has primarily to do with market share rather than any particular technical reason. Call me when Linux has had a 90%+ market share for a couple of years, and I'll be more than happy to revisit my opinion. I just don't think that the malicious code authors will give up and find something useful to do simply because the UNIX security model is "too hard."

That said, this article is about patching, rather than worms themselves. Obviously, the worms are the driving factor behind the recent interest in keeping patches up-to-date. This doesn't mean that your non-Windows operating systems aren't just as deserving of having their patches kept current, far from it. It's just that in terms of fire-fighting, you tend to go after the towering inferno first, and worry about the kitchen fires later. As we'll see, though, maybe while you're figuring out patching, you can take care of *all* your computers, and save some future headache.

Let's review. Here's the list of things that you're supposed to do that will "help keep you secure":

- Install a firewall
- Don't run unnecessary services
- Install patches
- Run antivirus (AV) software and keep it up to date
- Use an intrusion detection system (IDS)
- Educate users

Does this list look familiar? I think it's probably something like 15 years old now (with the addition of more recent developments like IDS, of course). Do you do all these? Of course you don't; at least, you don't do them right. It's not your fault, you can't. You've got nowhere near enough budget or management support. The closest I've seen to this being done right is in the military (well, certain military) networks. I don't doubt that I'll get email from net/sysadmins at these facilities disabusing me of that impression, too.

Sure, you've got the easy parts "done." You've got the firewall, and an enterprise AV software license, and the IDS is running. Maybe you've taken a stab at user education. How about the patches and disabling of unneeded software? Is the AV software actually installed, running, and up-to-date everywhere?

The basic pattern is that the pieces that are few, central, and under your direct control (firewalls, IDS, central servers) are well managed. The pieces that are many, distrib-

The challenge is putting patches on and keeping them on.

uted, and under user control are poorly managed. This is not surprising. With the central devices, you can do amazing things. For example, count how many you have. You can physically lay hands on them. You can lock them up and keep other people from laying hands on them.

Sadly, the centralized pieces, the ones you can actually get a handle on, have become less effective. Firewalls seem a lot more porous in the past few years. Internet applications have become very adept at traversing various types of firewalls. You've got computers coming and going, portable computers, VPN links, wireless access points (authorized or not). The two most prevalent worm attack points, HTTP and SMTP, were the very first two to be enabled at the firewall when it went in.

Signature-based technologies (namely, most AV and IDS products) have one basic drawback: The threat must be known in order for it to be caught. Generally speaking, in order to create a signature for something, you have to have caught and analyzed that something. This means that AV always lags behind the malicious code, and IDSes almost always lag behind the exploit. It has been reported that the SQL Slammer worm spread to the majority of vulnerable hosts in as little as eight minutes. That's not quite enough time to capture the worm, analyze it, create a signature, distribute it to customers, and have them put the new signature into production. That's an extreme case, but it illustrates the basic point. Keep in mind also that IDSes generally are completely passive; they only detect. Even if you could detect the problem within minutes, that wouldn't solve it. However, knowing that it is going on is the first requirement for fixing it.

Please don't take that to mean that I'm saying these technologies are useless, or that you can quit buying them. Not at all. They are still very much your minimum price of admission for installing an Internet connection. Sure, the IDS signature may not be available for a day after the worm is launched, but how else are you going to know that the worm has been running on your network since yesterday and is still going strong? You're going to need it to tell you which inside boxes are infected. There is always the option of using a packet capture program and manually reading through the packets. You should be capable of doing so when needed, but your IDS is supposed to automate part of this process for you.

What I am saying is that this (relative) weakening of the perimeter, coupled with the increased threat of the upsurge in worm volume, means that the level of security of your inside machines must increase. The word "security" means, in the context of worms, installing patches. Not all worms utilize only known vulnerabilities to infect a host, but the majority of them do, and it tends to be the most effective vector. One of these days, we will see a worm that uses an unknown vulnerability (no patch available ahead of time, a.k.a. a 0-day worm), but by definition, there is no patch to put on before it hits, so we won't worry about that for this discussion. (The general mechanism I talk about is still helpful in the case of a 0-day worm for clean-up, though.)

So, the challenge is to apply the various security patches and, secondarily, a little host hardening and monitoring. I further qualify that: The challenge is putting patches on and keeping them on. Why is keeping them on a challenge?

Imagine this scenario: A new Microsoft security hotfix is available. You mentally add it to the to-do list, until you get wind of a new worm that takes advantage of that vulnerability. So, you log in as domain admin and run your script:

```
C:\admintools\domainrun mydomain c:\dl\q8675309.exe
```

For purposes of discussion, this fictional (though quite feasible) tool finds all hosts in the domain "mydomain," uploads the program given, and runs it. Are we done?

Of course not. Here's a partial list of things that might have gone wrong:

- Some hosts were not in the domain.
- Some hosts were not on at the time.
- Some hosts were on, but not on the network at the time.
- Some hosts were Win9x, and the patch was for NT, 2000, and XP.
- Some hosts have the service that allows the remote install turned off.
- Some hosts didn't have a new enough service-pack level for that hotfix.
- Some hosts were running another language version of Windows and needed a slightly different hotfix.
- On some hosts, the patch program crashed.
- Some hosts were patched, but then configuration was changed later, and the patch got downgraded.

The list goes on, and that's just the Windows patches. For non-Windows, you can substitute "root password was changed" for "not in the domain," or similar. The concept and problems still apply.

You might imagine running the same command periodically, which fixes some of the problems. You might think about including similar commands in your login scripts. Either of these looks somewhat workable, until you've accumulated a couple of hundred patches and realize how long it is going to take for each cycle.

Having stated what I think the current set of challenges is, allow me to discuss how I think the challenges can best be met. In the interest of disclosing my biases to the reader, I want to point out that as of this writing, I'm working as a contractor at a company that sells enterprise software that does patch management (among other things). A lot of my research comes from having seen how that software works, and what I'm going to suggest you do looks a lot like what they sell. However, I believe they are on the right track and that the solution is valid. They aren't the only ones who sell software like this, so please take my statements as general and make your own decision about how you're going to get a system like the one I will describe. I'm not here to make a sales pitch.

In order to continually determine whether a patch is needed on a particular host, you will need a piece of software continually running on that host. I'm going to call this piece of software an "agent," but you can substitute whatever word you like. This agent must periodically check the system to see whether the host has a particular patch. Note that it is checking, not applying (yet). The checking process should be much quicker, overall, and therefore tolerable to do frequently. The agent will do this for each patch it knows about. Therefore, there must be some sort of list of all existing patches you wish to worry about, and must be updated as new patches are made available. I'll refer to this as "content." To recap, there is an agent that runs on every host, and based on some content it is given, it can determine which patches are in place and which are not.

Astute readers will be asking themselves how the agent gets onto the host in the first place. Ah, so there is a bootstrapping problem? Yes, and it's an ongoing bootstrapping problem, because new hosts get added all the time, and old hosts will somehow manage to lose their agents on occasion.

Allow me to briefly discuss the bootstrapping issue without going into great detail. There are a number of ways to do this – there's no single "best" way – and you may choose to use some combination of techniques. First, you could manually visit each host. It's boring, but at least it's obvious. Second, the techniques that I derided as having many problems for delivering a patch are actually not a bad way to get your 80% of hosts covered in a hurry. Use your domain credentials to blast the installer out, either with a command line program or by using the login script facilities. Third, you need some way to monitor your network for agentless hosts. You might do this with network scanning (say, the agent runs on a particular port and has a particular banner), you might tie in with your DNS/DHCP infrastructure, or you might ask your routers and switches which hosts they know about. I recommend all of the above.

Once you've got your agent running everywhere, what does it do exactly? Well, a good agent can do anything you like, but I'm here to talk about patching. You want the agent to download and install the patch. What, automatically and with no intervention? No, of course not. Don't run screaming quite yet, I'm not talking about a completely automatic, blind-faith patch install. I'm talking about putting on the patch that you will put on eventually, but at a schedule you pick, and without having to trek all the way to every host. The way we do this is by having all the agents report in to the central console.

In our design, the central console serves several functions:

- It tracks all the agents (so we know when they disappear).
- It reports which hosts need which patches (with appropriate sorting and searching functions).
- It issues commands to the agents to perform a particular patch install (at the prompting of its human operator).

Clearly, there is much more that the console could do (e.g., statistics, history, inventory tracking), but I'm going to stick with discussing these base functions.

As a console operator, your job is to look at what patches are needed and to apply them. After the patches are applied, the central console will report back on whether those agents now indicate their hosts are patched. Can you get away with doing Select All and clicking Go? Not if you're like most organizations. Many organizations have to deal with change control, patch program quality concerns, breaking software compatibility, service-level agreements, and similar issues. Well, no problem, your patch management agent is still going to make your life easier.

What does your change-control procedure say you should do to approve a patch for mass deployment? Test on your testbed hosts? Select those, and click Go. Test on some portion of your network? Select your IT department desktops, or your least favorite users, or whatever subset you like. You might even try finding willing volunteers, to avoid impacting your own long-term employment. Click Go. Want to make sure the patch is rolled out when help-desk personnel are available? Schedule according to day, hour, geography, follow-the-sun, whatever, click Go.

Have a set of hosts that your vendor supplies as "black boxes," even though they are running a stock version of Windows that needs to be patched like the rest, and they refuse to support you if you patch yourself? Well, sorry, can't help you there, don't click Go. They probably wouldn't let you put the agent on in the first place.

And, of course, when you've neglected your rollout schedule of patches for whatever reason, when the day rolls around that a worm is released for a vulnerability that you haven't fully vetted the patch for yet, you can decide if you'd rather have the worm or the patch of unknown quality. At least, if you decide the latter, you've now got the means to implement that decision in a hurry. (I would not recommend that admins not follow change-control procedure, or that they have faith that their vendors always produce flawless patch software, but most of the time you'd rather not pick the worm.)

So, is everything perfect? As described, such a system will let you determine which patches are needed, apply them according to your schedule, and verify that they stay there. This meets all of our requirements.

An actual implementation always leaves things to be desired. Could such a system lessen security in any way? Just like any software, it must be written with security in mind, specifically meaning, no security holes. Yes, if your agent has holes, then you've added that many more holes to each host. At least if these are discovered in the agent, you've got an easy way to upgrade the agent software in a hurry. (Actually, self-upgrade is not necessarily an easy problem, so if you're investigating such software, make sure to check into that.)

So, our agent is bug-free; what else can go wrong? Can an attacker pretend to be the console, and tell all the agents to download and install backdoor.exe? One would hope not. One solution that seems reasonable is to use public key cryptography to issue commands. In other words, PGP/GPG-sign the commands at the console and have the agents verify the signature against a stored key. And don't forget to deal with replay attacks (check that old, saved, signed commands won't work later).

And, obviously, keep careful control of the console. The public key crypto will help against some attacks, but if an attacker has full control of the console over a period of time, including installing a keyboard sniffer, it may not make any difference.

My hope is that this article has made you think a bit about some of the practical steps you can take to help keep your network worm-free. No solution is perfect, but we should still look for tools that can make a significant impact. The tools I've discussed here would help maintain a base level of security against worms and script kiddie-level attacks. Simple patching may not help save you from a clever, determined attacker, but who has time to worry about them with all these worms on our networks?

REFERENCES

"eWEEK Review of Patch Management Solutions," *eWEEK*, *http://www.eweek.com/article2/0,3959,1246104,00.asp*.

"PatchLink Helps Keep Windows Closed," *Network Computing*, *http://www.networkcomputing.com/1318/1318f3.html*.

"Windows Patch Management Tools," *Network World Fusion*, *http://www.nwfusion.com/reviews/2003/0303patchrev.html*.