

Letters to Santa--a real world attack

Part 9, Lessons Learned

What mistakes did Alabaster make that allowed us to compromise this server, and how could he fix them? List at least three mistakes and solutions.

The problems and fixes given below are not an all-inclusive list. There are probably several that I've missed. Also, we should be careful about blasting Alabaster. Developers are often overburdened and pressured to deploy servers and applications at the first possible moment. A good penetration tester can bring these problems to management and convince them of the need to expend the necessary resources to fix the problems. When management sees the risk that an underfunded IT program exposes them to, they may act.

Apache Struts Vulnerability

Obviously, the problem we used to get into Alabaster's server was the unpatched Struts vulnerability. The obvious fix is, "Patch the server!" If we really want to help Alabaster, we need to give him more detail than that.

How should Alabaster learn that he needed to patch in the first place?

The goal is to patch before the attackers break in, not after. Some things might have helped:

- 1) Read bulletins from all vendors whose software Alabaster uses. This is a good first step, but there are a lot of vendors and it is easy to miss one.
- 2) Read newsletters and distribution lists about vulnerabilities. One example is the SANS @RISK: [The Consensus Security Alert](#).
- 3) If the software was installed from a standard Linux library, using `yum update` or `apt-get update/upgrade` will update it. Many vendors provide update methods of their own.
- 4) Use a commercial vulnerability scanner. This is an important way to double-check your work. Products like Teneble's [Nessus](#) or Rapid7's [Nexpose](#) are kept up to date with vulnerabilities for most commonly used software. A weekly scan will alert you to any critical vulnerabilities that you have missed.

What if Alabaster cannot patch?

Suppose Santa told Alabaster, "This is the Christmas season! We can't have that server down for one minute. Patch after Christmas." Corporations with robust server farms may be able to take servers offline one at a time to patch without affecting performance. If Alabaster cannot do that, he can do other things mitigate his risk.

- 1) Employ an Intrusion Detection or Intrusion Prevention System (IDS or IPS) or a Web Application Firewall (WAF). An IDS monitors network traffic using signatures and algorithms to detect attacks. An IPS is similar, but also has the capability to block the attacks it detects. The attack we used was well known, and should have been detected by a reputable IDS, IPS, or WAF, assuming they were up to date.
- 2) Configure the firewall to do egress filtering. The I2s server needs to accept connections from the outside on ports 80 and 443, or else it is useless as a web server. Does the I2s server need to originate connections to any random Internet address? Usually not. There may be specific

sites the server needs to access for updates or other business, but those should be accounted for. All other outgoing connections should be blocked. If Alabaster's firewall was configured for egress filtering, it would have been much more difficult for us to make the Netcat and BASH attacks work. The web shell would still work even if egress filtering was in place, though.

Development Code and Sites left exposed to the Internet

Sites in development should never be exposed to the Internet at all. The process of putting a site into production should include checks to ensure that all development code and content have been removed.

Passwords Included in Code

This has been forbidden for 20 years, that I know of, and probably longer. It happens a lot, though. Enterprises should use Privileged Access Management (PAM) tools to protect against password usage in code.

Password Reuse

This is a big problem for all users, not just server designers. Alabaster had a password that would have been difficult to crack, but one mistake made it available to us. Every different use (database connection and server login, in this case) should have a different password. Since it is difficult to remember more than a few good passwords, they should be stored in a password vault that is protected by Two Factor Authentication (2FA) or at least a long complex password. If Alabaster had separate server and database passwords, we could have only looked in the database and been stopped if we didn't find anything.

Be careful what you install on servers

The fact that Netcat was installed on the dev/l2s server made our attacks much easier. The only software that should be installed on a server, especially one that is exposed to the Internet, is software that has a documented business case for being there.

Protect SSH

The first question to ask is whether the l2s server needs to be accessible to SSH from the Internet at all. If there is not a good business reason for using it, it should be blocked. If it is necessary, it should be protected with 2FA or digital certificates at a minimum.

Monitoring the Network

It is not reasonable to expect that Alabaster's defenses will block every possible attack, even if he improves them. It is critical that he monitor the activity on his network, especially outbound traffic that could warn him of a compromise.

- 1) Monitor IDS and firewall logs, looking for unusual outbound traffic. We used netcat to cause the l2s server to send plain text output on strange ports to strange IP addresses. This should have set off alarms. The Linux commands in URLs when we used the web shell should have triggered alarms as well.
- 2) Monitor the server logs. The exploit we used gave us access to Alabaster's account. When strange commands that Alabaster did not enter appear in his Bash history, that would have been a clue that something was wrong. If they happened when Alabaster was not present or unusual times of day, that is even worse.