Quokka

WHITEPAPER

What Is Vulnerability Management?

Quokka

Whitepaper

What Is Vulnerability Management?

In 2017, Equifax, a major American credit bureau, disclosed that it had become the victim of one of the largest data breaches in history. Over three months, the private records of more than 150 million individuals were compromised. The cause? A vulnerable version of Apache Struts was left exposed and unpatched for almost five months.

Vulnerability Management 101

Vulnerability management is the act of identifying and remediating security risks within any IT infrastructure. While failing to apply critical patches promptly is a common target of exploitation for attackers, the above incident was hardly the only vulnerability that plagued Equifax at this time. In addition to unpatched software, insufficient network segmentation, inadequate encryption of personally identifiable information, and ineffective intrusion detection systems all coalesced into a months-long cyberattack that left tens of millions of people exposed.

The issues listed above suggest that Equifax's vulnerability management process was severely lacking. While nobody can definitively say that this attack could have been prevented had they done one thing or another, their overall lack of security hygiene not only allowed attackers into the network but also gave them nearly unrestricted access once they were inside.

By establishing a system for identifying, classifying, and remediating vulnerabilities, Equifax could have made the overall impact of this incident significantly less severe. For example, quickly applying security patches can close holes in the outer layer of a network, while establishing robust role-based access controls around private data can limit what attackers can access should they make it inside.

Reactive vs. Proactive Remediation

The problem with most vulnerability management methods is that they rely heavily on reactivity rather than proactivity. Once a vulnerability, no matter how small, has impacted your business, mitigating it is always a risk. A breach only takes a few minutes to have drastic (and often unexpected) consequences, so proactively protecting against vulnerabilities before they can even occur can greatly improve your security posture.

But what does it mean to be proactive when it comes to remediating vulnerabilities? Take, for example, the above case of an Apache Struts vulnerability. A well-executed but reactive approach to managing vulnerabilities in a deployed instance of Apache Struts might be to automatically apply any security updates as they are released. While this would ensure that your deployment of Struts is "as secure as possible" given the circumstances, it doesn't account for the possibility that it had already been compromised before the underlying vulnerability was disclosed and patched.

Proactive remediation, on the other hand, considers the possibility that everything is vulnerable and strives to mitigate the overall impact long before it becomes a problem. In the example of Apache Struts, a proactive approach to mitigating the vulnerability might be to design the network to isolate unrelated services and subnets to prevent lateral movement within the network. Or, you could enforce multiple authentication and authorization mechanisms to better protect private information.

While these approaches might proactively mitigate problems, it’s important to note that they don’t proactively remediate those problems. True proactive remediation would be to identify the vulnerabilities in Struts long before it is even deployed, which can be accomplished in several different ways, such as manual penetration testing or (even better) deploying static and dynamic analysis tools above and beyond that which is already deployed by the underlying project itself.

To Trust, or to Zero Trust?

One of the most popular ways to manage vulnerabilities within modern network infrastructure is to assume that they are already being exploited and act accordingly. This is called a zero trust model, and it encourages structuring authentication and authorization around the idea that "the call is coming from inside the building."

This is an effective security methodology, as it explicitly limits the amount of damage that most attackers can accomplish from within a single system. A well-designed vulnerability management program can also bolster your network. By using a zero trust model, you look at the network from an attacker's point of view, allowing you to better identify any potential holes in your infrastructure.

The same concept of zero trust can also be applied outside the network. When you assume that an application like Apache Struts already has unreported vulnerabilities, the way you deploy it will change. You could proactively limit the ports and addresses through which it communicates or run it within a sandbox environment. Depending on how critical it is to your business, you could even run your own scans and analysis on it, ultimately contributing to the overall security of Struts for all consumers.

Although utilizing more robust techniques like security fuzzers and other static analysis platforms can help identify some low-hanging vulnerability fruit, newer AI-powered dynamic analysis tools can identify previously unknown vulnerabilities in even closed-source applications. These tools can be especially valuable because they learn from every vulnerability identified, ensuring they stay updated with the constantly changing security landscape.

Taking It Further

Vulnerability management isn't a goal—it's a practice. You're never "done" managing vulnerabilities in the same way that you're never done growing. There is no one-size-fits-all solution, meaning it's on you to identify your risks and mitigate them based on your own needs. Zero trust models can be combined with vulnerability management methodologies like reactive and proactive vulnerability remediation to create a robust security program that can reduce risk and increase confidence in your procedures.