



Framework for building a vulnerability management lifecycle program

<http://searchsecurity.techtarget.com/magazineContent/Framework-for-building-a-vulnerability-management-lifecycle-program>

August 2011
By Diana Kelley

Vulnerability management is about more than patching. To build a truly robust program an organization needs to incorporate inventory management, configuration management and change management into the patching lifecycle. And for even more effective control and governance, penetration testing and patch and control validation should be added to the mix as well. That's a lot of moving parts, and depending on your organization, these parts could span multiple business units and geographic locations. Getting it right and keeping it running smoothly can be a challenge.

We will present a framework for building a vulnerability management lifecycle. Using examples from practitioners, you will get a from-the-trenches view of what works and what doesn't when trying to win the ongoing vulnerability management war.

Vulnerability management lifecycle: defining vulnerability management

Computing environments are complex systems comprised of hardware, software operating systems and platforms, applications, services, and the people who interact with all of the above to get their jobs done. Vulnerabilities can exist anywhere in environment, and managing vulnerabilities is a non-trivial task.

At its simplest, vulnerability management (VM) is a matter of applying security patches as they become available. But robust VM is about more than patching – it is about defining the risk posture and policies for an organization, creating a complete asset list of systems, applications, and services, scanning and assessing the environment for vulnerabilities and exposures, and then taking action to mitigate or accept those vulnerabilities. One way to mitigate a vulnerability is to patch it, but there aren't always patches available -- and even when there are, it isn't always possible to apply them. Another issue is that most networks are continually evolving; introducing new services and applications can impact the vulnerability profile of the system as a whole.

All of these are reasons why an effective vulnerability management program needs to be part of a larger lifecycle, one that takes into account the existing network, new additions, ongoing testing, change management, ticketing, validation, and multiple mitigation types

-- including patches and compensating controls. Because aspects of that lifecycle interact with multiple departments and constituencies within an organization, it must be part of the fabric of the business operations, not the exclusive domain of security. Larry Whiteside, CISO for the Visiting Nurse Service of New York, says that to have a truly successful vulnerability management program, it needs to be approached as “an organizational problem that includes operations, IT architecture, security, and the business.”

Core vulnerability management processes apply across all networks

One of the most common pitfalls in VM can be trying to do too much too soon and getting overwhelmed by the magnitude of the problem. J. Wolfgang Goerlich, network and operations manager for a midsize money management firm, recommends starting small. Rather than “trying to do everything at once and having too much for the team to accomplish, build success on a select number of critical systems and processes” and grow the program from there, he says.

To do that, first think about the process steps in the vulnerability management lifecycle. Each organization is unique and may wish to implement these steps in different ways, but the core milestones in the lifecycle share applicability across most networks.

Policies and postures inform the entire lifecycle because risk acceptance is ultimately up to each organization. For example, most entities don’t wish to take on penalty costs for failing to adhere to a compliance mandate, so these organizations will write compliance to regulations into their policy requirements. Business drivers play in to this as well: For example, an organization that relies on keeping its intellectual property highly protected will write a data classification policy that puts a high priority on data confidentiality. Include risk assessment and business impact analysis as part of defining acceptable security posture. Policy definition work may feel abstracted from application patching, but it’s essential for the impact assessment and prioritization phase later in the lifecycle.

Accurate asset inventory vital to vm success

Once policies and postures are defined, the next step is determining what currently exists in the network. Create an inventory listing all operating systems and applications, including off-the-shelf and custom applications, databases, and application servers. Accuracy of these lists is critical to the health of the VM lifecycle, so ensure that updating the list(s) is part of the process. Goerlich says he uses “a product lifecycle model that includes a vulnerability assessment during the project evaluation, during implementation, and again post-implementation.”

Once it has been implemented, a new service or application should be placed into the inventory list. But not every organization is able to put new devices or services through an assessment prior to having it on the network. Consider an environment such as higher education where an influx of new students and their devices enter the network every semester. Seth Shestack, associate director of information security for Temple University, handles asset inventory for student devices by cordoning unregistered

devices into a quarantined network and having students register the devices and install endpoint protection before allowing them on to other parts of the network. In this model, registered devices become part of the asset inventory list as they go through the registration process.

Vulnerability scanning options abound

Asset scanning (and discovery) can be included as part of vulnerability scanning, but true vulnerability scanning goes deeper into the actual exploits and risks. Vulnerability scanning can also be done externally or internally. Internal scanning can target more devices since an external scan may not be able to get past security mechanisms to get to the internal network -- such as IPS devices in active prevention mode -- but does provide a more realistic view of what an outside attacker sees. And it can also validate that security mechanisms, such as IPSes and firewalls, are doing their intended jobs and preventing outsiders from being able to see deeper into the network.

Vulnerability scans can be done non-intrusively in passive OS fingerprinting mode that returns the patch level and other basic information about the host, such as which ports are open. Credentialed scanning goes a step further to gather more detailed information about the target, such as which applications are installed using customer-supplied passwords and/or community strings for SNMP scans. Scanning can also check against recommended configuration levels as defined by policy.

Another option is to perform automated or manual penetration tests as part of the scan process. Penetration testing actively attempts to exploit a system. A simple example is to attempt to login to a database or wireless router using a vendor-supplied default password. A more complex example is to use blind SQL injection on a Web application to extract passwords or other information from a backend database. A penetration test can also expose attack paths through a network that may not be visible using a standard vulnerability scan. Applications pose another level of complexity. Custom applications, less commonly used and niche applications, and highly customized popular applications, should be scanned and pen tested using custom rules or manually. Most large vendor scanning tools can't ship with scan and test plans that include all the possible vulnerabilities for all applications; this is not a shortcoming of the tools themselves, it is more about the sheer numbers of applications running on networks.

After running vulnerability scans, including credentialed scans, host-based scans, and penetration testing where applicable, an organization should have a list of vulnerabilities or issues. Goerlich has integrated scan results with a ticketing solution. "When something needs to be investigated, the results of the scan are exported and attached to a new ticket," he says. At Temple, the link between scanning and ticketing has been done manually, but Shestack says the university is currently in the process of integration. While automatic integration can speed up the process, the important step is to ensure there is a connection between the scan results and systems or processes used in the next phase for impact analysis, prioritization, and remediation.

Vulnerability impact analysis and prioritization

With a list of issues either in a spreadsheet, a scan result report, or integrated into the trouble ticketing system, the next step is to understand what kind of impact the vulnerability can have on the organization and to prioritize the response activities. While it would be nice to be able to fix all vulnerabilities as soon as they're discovered, the reality is a bit more challenging, which is where the policy and posture work done in the first step really comes in. If an asset or data on the asset is deemed to be highly critical or have a high business impact, priority for the fix will be greater.

Other information that can be useful in the prioritization process is information from publicly available sources, such as the National Vulnerability Database and the Common Vulnerability Scoring System calculator. Vulnerability scanning tools use standard scoring to provide scoring metrics, and vendor advisories many times provide their own scoring on vulnerabilities as well. Whiteside at the Visiting Nurse Service of New York uses a combination of resources, including Symantec DeepSite and ISAC feeds when creating a scoring metric for his vulnerability program. But he does caution that while "vendors create their own risk calculation based on vulnerability information, it doesn't tell the whole story for my enterprise." Whiteside's team places a metric around the vulnerability that includes location, type of system and other factors unique to the environment to come up with a score that is adapted to his enterprise. Higher ranked vulnerabilities must be patched or mitigated within 30 days, while others can be placed on a 60- or 90-day fix cycle.

Another point to consider in the impact and prioritization phase is whether or not a system is truly vulnerable. In some instances, a compensating control such as a firewall may already be in place that prevents the vulnerability from being exploitable. In cases like these, it may be possible to tune the scanner to stop scanning for that particular vulnerability on the protected host or to create an exception report that "scores down" the impact due to the presence of the compensating control.

Vulnerability management lifecycle: remediation and mitigation

With a list of vulnerabilities tuned to the real risk and impact of the organization, it is time to remediate or mitigate. Patching is the most commonly discussed remediation technique: If there is a patch available, now is the time to apply the patch to golden images and testing servers to ensure it can be applied without unintended consequences, such as disruption of service. Some organizations deploy patches directly through their vulnerability management solution consoles, while others deploy through an operations management console used for other kinds of software delivery like maintenance updates and new software deployments.

But there's not always a patch for known vulnerabilities in commercial software, for example when a vulnerability is known and the vendor doesn't have a patch ready yet. Custom software, including outsourced and in-house applications, can lag in the patch department or the development team may not have the time and resources to create a

patch or rewrite the code. And some systems may not be available for patching for other reasons: For example, if an audit cycle is in progress and all system changes, including patching have been frozen, or in the case of some certified special-purpose devices, like those used in health care, the federal sector, or payment systems.

When patching isn't a possibility, organizations can address vulnerabilities in other ways, like compensating controls; for example, Web application firewalls with custom rules to prevent exploits. Other options include stronger access controls, continuous monitoring solutions, and sandboxed or terminal services to separate the service from the rest of the host.

Validation confirms patches applied and working

The last phase in the vulnerability management cycle is to validate that the fix has been applied and is working as expected. Validation is often done using the same tools and techniques employed in the vulnerability assessment and scanning phase. Whiteside runs validation scans a few days after the fix window. "If the vulnerability still exists, it shows the fix has failed," he says. "On the reverse side, if the vulnerability is no longer reported, [we have] verification that the process is working as expected." At Temple, Shestack also uses agent-based, host integrity checking that forces Windows Update to be on and ensures Windows applications stay patched.

Upsides to a mature vulnerability management program

Though taking the time to implement a holistic vulnerability management lifecycle may seem like more work than a simple patch program, there are valuable upsides. As the program matures, efficiencies can be found that increase automation and reduce risk.

All of the experts interviewed for this article agreed that increasing integration between components within the lifecycle can bring significant benefits. Goerlich currently uses Microsoft System Center because it's integrated with change management and helps his company to be more proactive. The goal is to "put the system into production in a secure fashion to begin with and bring on new systems and services in a manageable way," he says. But integration for all components may not always be possible, as Whiteside observes: "Bigger players have done a good job with integration, but if you're using smaller vendor tools, integration will be harder; you can't expect them to play in every single sandbox." So look for areas and solutions where integration wins can be implemented and keep an eye out for increased integration as solutions and the program matures.

As the program matures, look for success areas to expand on. Companies are putting additional focus on software vulnerability management and adding deeper database scanning and monitoring into their programs.

An area that vendors and organizations could improve over time is metrics and trending data. Whiteside notes that trending is very important, but missing from many of the vendor tools. And Goerlich says, "What we really need is solid information regarding

what the threats are and at what frequency they're occurring. Knowing the likelihood of exploit is very helpful."

Robust vulnerability management is about more than patching -- it's about creating a repeatable lifecycle that includes multiple components that are part of a larger risk management and control program. Building a VM lifecycle may take some time, but it's time well spent if it means a reduction in exploits, as Shestack says, "It's much easier to do incident prevention than it is to do incident response."

Diana Kelley is a partner with Amherst, N.H.-based consulting firm SecurityCurve. She formerly served as vice president and service director with research firm Burton Group. She has extensive experience creating secure network architectures and business solutions for large corporations and delivering strategic, competitive knowledge to security software vendors.