When is a Vulnerability Not a Vulnerability?

Vulnerability Scoring & Improvement-Based Vulnerability Management

A New Net Technologies Whitepaper

Mark Kedgley

CTO - New Net Technologies

www.newnettechnologies.com

Abstract

Vulnerability management is a key security best-practice that serves to prevent the complete spectrum of cyber-attacks. But how do you strike the right balance between maintaining the security of an IT environment that never stands still, and maximizing system performance, uptime and service delivery?

After all, IT systems should exist to serve the business, not constrain it through over-zealous vulnerability scanning. This white paper examines the options for streamlining the management of vulnerabilities through the various scoring systems that exist and pro-poses a new approach of 'continuous and improvement-based vulnerability management.

Are you a 'Checkbox Compliance Cowboy'? Or do you just have slick processes?

Information Security is an industry full of buzzwords, acronyms and clichés. The GRC sector in particular is rife with them (which succinctly proves my point about acronyms – GRC: governance, regulatory and compliance).

For example, the expression 'Checkbox approach to Compliance' is disparagingly aimed at anyone who treats compliance as a project. For these Checkbox Compliance Cowboys, compliance receives focus once a year for a period of a few weeks and with the sole intention of providing enough paperwork to satisfy an auditor, but with little substance beyond that.

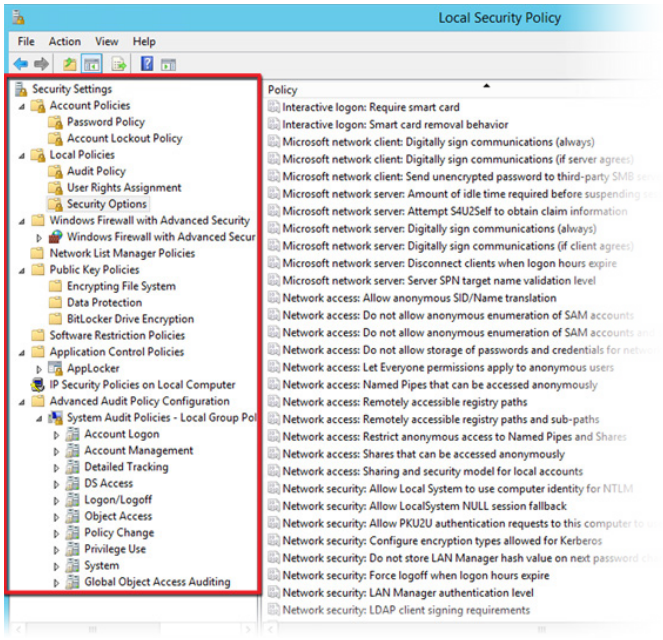

Figure 1: Windows Security and Audit Policy comprises hundreds of settings to mitigate vulnerabilities - but they must be configured correctly to be effective

Don't get me wrong - those that treat compliance as cynically as this are missing the point. Threats to security are constant and therefore security measures and the associated checks and balances of compliance also need to be operated continuously.

But the fact is that most of us would actually prefer to take a checkbox approach, in as much as we are all looking for ways to make compliance a more predictable, less time-consuming and simpler function.

And who can blame us? Security and compliance is a hugely complex task, and the implementation of a hardened build standard is a highly technical project in its own right. Finding the right balance between a configuration standard that protects systems without preventing them from working needs careful consideration.

For example, the recognized industry-standard for Hardened Build Standards are the CIS Benchmark Checklists, covering all popular platforms and applications. Typically each of these contains several hundred recommendations for security configuration, including a rationale and a health warning.

For instance, account lockout settings are a classic example of a security setting with no absolutely ideal setting. Too lenient, and brute force attacks will be rendered more effective, but too strict and Denial of Service attacks become more of a threat.

Three Classes of Vulnerabilities to Deal With

Overlaid with configuration hardening is the related task of patch management. Both disciplines will address vulnerabilities and both can have nasty side-effects.

On this basis, within the overall context of Vulnerability Management, it is valid to group all vulnerabilities together, and indeed, many vulnerability scanners will aim to detect both configuration and software-based vulnerabilities with one scan.

However, because the nature of vulnerabilities and the action required to either mitigate or remediate them are so different, it actually makes sense to segregate their management.

Maintaining a hardened build standard for Windows or Linux hosts is a very different discipline to managing weekly patching exercises and should be measured and handled accordingly. For completeness it is also worth highlighting that there is a third class of vulnerability to appreciate.

Software Security Configuration Vulnerabilities

For example, disabling remote desktop access to a server, limiting remote access to registry paths/shares, password policy settings and User Access Control are all optional security features that you choose to enable for a PC. Similarly in the Linux world, allowing root access via SSH, configuring sticky bits, groups and permissions for paths and removing unnecessary services are all configuration options that will enhance the hosts' security.

Software Flaw Vulnerabilities aka Software-based Vulnerabilities

These vulnerabilities are categorized by the fact that an unintended fault, call it a bug, in a piece of software inadvertently provides a security weakspot.

Software Feature Misuse Vulnerability

There is also a third category of vulnerability which is a subtle variation on the previous two. A Software Feature Misuse Vulnerability is a vulnerability whereby a valid software feature can be exploited to compromise a system. The feature is not a bug or flaw, nor is it a configurable attribute of the system to specifically enable or disable security (i.e. a software security configuration vulnerability), but more of a side-effect of a software feature that inadvertently affects security.

Figure 2: Defining the three classes of vulnerabilities

Groundhog Day - The Traditional Scanner Approach to Vulnerability Management

One of the main obstacles to making vulnerability a streamlined process is that there is a tendency to always be starting at square one.

The vulnerability landscape changes daily with new exploits being discovered and reported, so new scan signatures will always be available. There is also the issue of needing to know which devices you have and where they are located in order to scan them – a secure network is going to be firewalled to prevent scanning activity. Finally it is always better to operate a scan in a focused manner, which means knowing what is installed on the hosts under test in order to specify which vulnerabilities to test for. The alternative is to just run a simple but overkill, ***'route-one-let's-test for every exploit of every package'*** but in a large estate this is just too wasteful of resources and time.

But once the scan results are reported, the real work begins. Each failure needs to be reviewed in turn for its relevance and associated risk. In a large estate, where remediation work could take days or even weeks, which vulnerabilities, for which devices, should you address first?

- For config based vulnerabilities, is it practical to mitigate the vulnerability, given that reducing the opportunity to exploit vulnerabilities invariably reduces functionality (for example, restricting RDP access makes a Windows server more secure, but would compromise support capabilities)?
- Likewise, is it safe to go ahead and patch a system? An update that addresses a vulnerability may well introduce other issues such as feature/functional changes or even a new bag of bugs

Faced with these potentially undesirable side-effects, the first question to ask is ***'How serious is this vulnerability?'*** or, in other words, does the risk posed by the vulnerability outweigh the risk of causing other operational problems?

Figure 3: Doing more harm than good? Vulnerability mitigation/remediation work always requires careful consideration before implementation. A metric that defines the associated risk is highly attractive, but the definition of that risk introduces different issues

Risk Assessment and Vulnerability Scoring Systems

Various systems exist which attempt to categorize and score each vulnerability. Qualys have their own scoring system as do Tripwire ® (and nCircle), but there are also the consensus-based systems, presided over by NIST, which reference the three earlier definitions of vulnerability classes. In turn these are

Figure 4: Assigning a risk score to a vulnerability is a logical step in order to prioritize remediation work, however the scoring systems are necessarily complex and nearly always too opaque to interpret for your environment

- ***Common Configuration Scoring System (CCSS)***, used to score the severity of security configuration-based vulnerabilities
- ***Common Vulnerability Scoring System (CVSS)***, used to score the severity of software flaw-based vulnerabilities
- ***Common Misuse Scoring System (CMSS)***, used to score the severity of software misuse-based vulnerabilities

At a high level, the intention is clear – define how potentially dangerous each vulnerability is. But that isn't such an easy assessment to make and scoring vulnerabilities starts to get very complicated, very quickly.

Each of the Common Scoring Systems factor in the context of the threat: ***'Just how likely is it that this exploit can be used?', 'How real is the exploit?', 'How available are the fixes, and how risky are they?', 'How much damage could be done using the exploit?'***

In the CCSS system the vulnerability is given a 'Base Score' based on the

- ***Access Vector*** (Local, Adjacent Network or Network)
- ***Access Complexity*** (High, Medium or Low)
- ***Authentication requirements*** (Multiple, Single or None)
- ***Confidentiality Impact*** (Complete, Partial or None)
- I***ntegrity Impact*** (Complete, Partial or None)
- ***Availability Impact*** (Complete, Partial or None)

Next, there is a ***'Temporal Score'*** applied, based on

- ***Exploitability*** (Not Defined, Unproven that exploit exists, Proof of concept code, Functional exploit exists or High)
- ***Remediation Level*** (Not Defined, Official fix, Temporary fix, Workaround or Unavailable)
- ***Report Confidence*** (Not Defined, Unconfirmed, Uncorroborated or Confirmed)

Then there is the ***Environmental Score***….do I need to go any further?!

Do vulnerability scoring metrics work?

Consider CVE-2004-2761: The MD5 Message-Digest Algorithm is not collision resistant, which makes it easier for context-dependent attackers to conduct spoofing attacks, as demonstrated by attacks on the use of MD5 in the signature algorithm of an X.509 certificate

This one comes up a lot because it is typically reported with respect to a Web Server using a self-signed certificate. An internet-facing website needs to be operated with an SSL certificate generated using a strong signature algorithm to prevent certificate forgery/spoofing. The CVSS Score is around 5 i.e. Medium to High – Scary!

However, all manner of other web-enabled systems may well use self-signed certificates which are typically MD5-based. So if the website in question is actually a web interface to a non-business critical, internal-use only system, that doesn't hold any PII or other confidential data, and is on an internal web segment behind an internal firewall and the external internet firewall, is this still a Severity 5 Vulnerability?

Context is everything - Intelligent compliance beats vulnerability scanning

From an academic standpoint, all the factors outlined should be taken into account as they allow a quantitative score for any vulnerability to be derived based on its qualitative attributes.

But as the consumer of the scan report you just want a High, Medium or Low severity rating - you don't need to worry too much about how the Vulnerability Score was calculated.

Or do you? Without the context of your estate and network architecture, the risk-level of a vulnerability can only be calculated on a theoretical, not empirical, basis.

Now, the point is that there are no vulnerabilities that should be ignored, but there are any number that within the context of your estate might be tolerated temporarily or permanently due to compensating controls that are in place. SCADA infrastructure components subject to NERC CIP compliance will require the highest level of security, while user workstations segregated from confidential data systems can be treated as lower priority, lower risk items.

With scan results highlighting hundreds of vulnerabilities across the estate, the last thing you need therefore is to be re-reminded every time you scan of the same known-and-acknowledged vulnerabilities. The concept of improvement-based vulnerability management starts with the need to address this issue as a key objective.

Continuous improvement is key - Measure compliance for *your* estate with *your* hardened build standard, taking into account the context of *your* systems

For example, with a large compliance initiative, there could be any number of reasons why servers or network devices will remain in a non-compliant state for months – resource constraints, application compatibility, network architecture – so the requirement to either suspend or exclude compliance requirements for certain hosts or device groups is essential. If we think it will take us 3 months to remediate all vulnerabilities across all systems then we can set time-based milestones for minimum levels of compliance to be achieved, and in doing so, give a realistic set of targets to hit progressively over time without being repeatedly beaten on over all vulnerabilities outstanding.

Similarly there may be a need to make exceptions or adjust compliance requirements, for example, allowing permissions to additional Groups over and above the standard settings advocated by the CIS Benchmark.

Finally the ability to also extend the compliance standard to include additional file integrity monitoring checks over and above the **STIG** or other secure build standard is valuable. For example, security best practices may recommend removing or disabling unnecessary daemons and services, but you can also use your compliance audit to ensure that other essential services are enabled and running, such as encryption, syslog forwarding agents, DLP or AV products. Likewise, ensuring a functional build standard for a host is implemented and maintained in terms of installed software, filesystem structure and network settings gives a dimension of quality control that will eliminate downtime/reduce troubleshooting.