Building an Advanced Vulnerability Management Program

Deron L. Grzetich

March 31, 2014
Abstract

Organizations continue to struggle with implementing a robust Vulnerability Management (VM) program that is capable of showing a quantifiable reduction in risk to the business despite the existence and need for these programs for the last decade or more. This paper examines some of the historical challenges of building a vulnerability management program including the required foundational elements, organizational alignment, and technological challenges. It also builds on this foundation to allow an organization to generate the maximum benefit in terms of risk reduction and a reduction in the work effort required by IT or operational groups or those responsible for applying the remediation or fix. One key element that will be examined are the options available to provide for risk re-ranking of identified vulnerabilities through the use of threat intelligence information, network path determination, and mitigating control effectiveness tracking. Finally, this paper will examine the options for remediation workflow tracking and overall reporting and metrics that may be used by the program to track overall program effectiveness.
Introduction

Vulnerability Management (VM) programs have been deployed by most organizations in some from or fashion over the past decade or longer and have historically been driven from various needs ranging from regulatory/industry compliance to altruistic attempts to reduce risk that is introduced by information technology. While most organizations recognize the importance of a VM program they have generally failed to build robust programs able to show a true reduction in risk through the treatment of vulnerabilities that present the most risk to the organization. Some of the inhibitors to building a robust program are related to a lack of the foundational elements such as asset management or workflow management and remediation tracking systems. Established programs may also lack the right level of comprehensiveness and completeness of coverage in their assessment activities or the ability to sort through the mounds of data generated by these tools. Furthermore, even fewer programs have the ability to infuse threat intelligence information, network path or accessibility data, or factor in the reduction in risk afforded by mitigating controls and their associated effectiveness. Finally, reporting on the effectiveness of the program and presenting an accurate depiction of risk is a key element that is often missing from programs due to many of the previously mentioned factors. This paper aims to examine historical failures of VM programs and presents possible solutions to the path forward to build robust effective programs that are able to reduce risk and the steps that can be taken to iteratively increase maturity.
Method

The method to complete this research was to leverage previous experiences of the author in both building VM programs and years of assessing VM programs, building future state roadmaps and strategy, and implementing VM programs for various large organizations in the financial services, insurance, energy, professional services, and oil and gas industries. In addition, research on available technologies and the benefit they afford will be conducted to support potential solutions to the various problems encountered by those building and managing vulnerability management programs.

Vulnerability and Patch Management Defined

It is important to fully define what is meant by a VM program as it often has different meanings for different people and organizations. VM is sometimes confused with the act of identifying vulnerabilities through assessment technology that scans active systems on the network. While assessment is a component of a VM program, by itself it does not constitute a full, robust, or closed-loop VM program. A VM program, for the purposes of this paper, is the process whereby vulnerabilities are identified on a recurring basis and are delivered to the party responsible for treatment according to a pre-defined service level agreement. The treatment that is applied may include the application of a fix or patch, change in configuration or the application of a mitigating control, or acceptance of the risk incurred by doing nothing. A key element in this definition is that all identified vulnerabilities that are required to be treated reach a determined state and that the determination status is tracked throughout the lifecycle of the vulnerabilities through a defined workflow. That last aspect provides for reporting on the effectiveness of the program in addition to tracking risk that have been accepted.
To further illustrate the process it is depicted graphically below as a set of steps or actions:

![Diagram](image)

**Figure 1. Standard VM process flow**

It should also be noted that patch management, which is often mistaken as vulnerability management, is only one of the available treatment options available to those charged with vulnerability remediation. For the purpose of this paper we differentiate between an "on rails" patching process and the patching activities that are driven by the output and workflow tracking of a VM program.

An "on rails" patching process is the process by which vendor-supplied patches are regularly applied to systems and applications that may address both security as well as stability and usability issues. While it is possible to have a VM program that provides a "list" of patches that must be applied during the patch cycle it can also be driven by a "patch all" mentality whereby all patches are applied regardless of the issue or risk that is being addressed by the patch. It is not uncommon to see this mentality taken on by IT organizations that do not have a defined patch management process or are not being adequately directed by their counterparts in information security. It is not necessarily a bad approach as it does not rely upon the guidance or directive from a group outside of the
function that applies the patches to the organization’s systems. It mitigates security risks through the application of patches in the absence of direct guidance in terms of which patches to apply and allows the output of the VM program to measure the effectiveness of the patching process. The obvious drawback of this approach is that the organization may be spending cycles tracking down non-application or application issues for patches that do not significantly impact or reduce risk in the organization.

For organizations that have developed a patch management process it is common to observe a definition of the attributes of the process such as the types of platforms (i.e. systems and applications) that are included, the patch cycle and application cycle, and the expected outcomes of the process. An example of such an attribute table for a fairly mature organization is shown in the table below as an example:

<table>
<thead>
<tr>
<th>Platform</th>
<th>Application</th>
<th>Completion Cycle</th>
<th>Expected Outcome (Within Cycle)</th>
</tr>
</thead>
<tbody>
<tr>
<td>End Point Operating System (i.e. Windows)</td>
<td>30 Days</td>
<td>14 Days</td>
<td>95% application/adoption</td>
</tr>
<tr>
<td>End Point MS Applications (i.e. Office)</td>
<td>30 Days</td>
<td>14 Days</td>
<td>95% application/adoption</td>
</tr>
<tr>
<td>End Point 3rd Party Apps (i.e. Adobe, Java, etc.)</td>
<td>30 Days</td>
<td>30 Days</td>
<td>80% application/adoption</td>
</tr>
<tr>
<td>Server Operating System (Windows)</td>
<td>30 Days</td>
<td>30 Days</td>
<td>95% application/adoption</td>
</tr>
<tr>
<td>Server Operating System (Unix/Linux)</td>
<td>90 Days</td>
<td>30 Days</td>
<td>95% application/adoption</td>
</tr>
<tr>
<td>Server Applications (Microsoft)</td>
<td>30 Days</td>
<td>30 Days</td>
<td>95% application/adoption</td>
</tr>
<tr>
<td>Server Applications (3rd Party)</td>
<td>90-120 Days</td>
<td>30 Days</td>
<td>80% application/adoption</td>
</tr>
<tr>
<td>Middleware, Custom Applications, Database</td>
<td>120 Days</td>
<td>60 Days</td>
<td>80% application/adoption</td>
</tr>
</tbody>
</table>

**Figure 2. Example of patch management attributes for a mature "on rails" program**

While the table above outlines the expected outcomes within the patch cycle it also assumes that the organization will continue to increase application and adoption over time in order to decrease risk. However, it may not be possible to attain 100% compliance due to
exceptions that are granted for application or system stability issues. These exceptions would be formally tracked in the VM program as part of workflow management discussed later in this paper.

An astute reader will note that the table above does not address critical “off cycle” or “0-day” patches that are periodically released by vendors. While not covered in the table above it would be an element of the overall patch management process and would utilize the same deployment mechanisms as the “on rails” process. The decision on whether or not to deploy the patch “off-cycle” would be driven as an output of the VM program. This is one of the connection points between a robust VM program and a patch management process.

Patch management is effectively the railroad for patch deployment while the VM program provides both guidance on “off-cycle” patches as well as acts as a measure for the patch management process. The process diagram below further illustrates the interconnection between patch and vulnerability management:

![Diagram of data feeds and process flows for patch and vulnerability management]

**Figure 3. Example data feeds and process flows for patch and vulnerability management**
As we can see from the diagram above it is understandable why one may confuse patch and vulnerability management as they are tightly tied together. Because of this tie we will continue to examine patch management in the context of vulnerability management throughout this paper.

An additional topic not yet addressed through the definitions presented in this paper is configuration management. While in the same vain as patch management, being that is it a potential treatment option for identified vulnerabilities, it is often a secondary action based on the absence of the vendor-supplied patch. Configuration management is the process of ensuring that a consistent secure configuration is applied to like IT assets. As an example, all Windows server systems should have a secure baseline configuration standard applied as part of the build process and tracked throughout the life of the asset. While this allows for the configuration of a system to act as a potential mitigating control for vulnerabilities, it often has a side benefit to IT operations in terms of troubleshooting and problem management. Although important to the overall health of an organization's IT environment it will not be a focus topic in subsequent discussion in this paper.

**VM in the Context of Risk Management**

At the core of a VM program is a drive to reduce and manage IT risk in the organization. To that end, it is important to frame how a vulnerability impacts risk both initially and over time as we identify vulnerabilities, assign and track remediation actions and get vulnerabilities into a determined state. The following graphic illustrates the lifecycle of the risk of a vulnerability over time for discussion:
As we can see from the graphic above the risk level related to a vulnerability will change over time. Initially a vulnerability may be discovered by the vendor, researchers, or those with malicious intentions. If the vulnerability is disclosed to the vendor for correction or patch development there will likely be a period of vetting the vulnerability and patch development prior to announcement or vendor confirmation or announcement. There also exists an alternate path in which a vulnerability is discovered and included as an exploit in an exploit kit or is actively used in a targeted campaign, at which time the vendor will be made aware of the existence of an exploit and back into the vulnerability. The graph above only shows one possible path and had been simplified for the purpose of discussion.

Regardless of the path, if the vulnerability is one that allows code execution or escalation an exploit for the vulnerability will be created. At this point there is an inflection point in the risk level as we have a confirmed vulnerability for which an exploit but no patch exists.
We also see that we now enter the area of preventative control influence at this point in the timeline. The effectiveness of mitigating controls that have been deployed may significantly change the slope of the risk line in the graph. Preventive controls, in the context of the cyber kill chain, may include those that prevent delivery of the exploit to those that prevent exploitation of the target, installation of additional payloads/binaries, and those that disrupt communications between the compromised system and its handlers or external parties. Examples of preventive delivery technologies may include web and email filtering and proxy solutions or malware prevention systems that operate at the network layer such as advanced malware prevention solutions and intrusion prevention systems (IPS). Examples of preventive exploitation and installation technologies may include host-based AV and anti-malware technologies, application whitelisting and control solutions, and application sandboxing/containerization technologies.

At some point in the timeline a patch or fix is made available by the vendor to address the confirmed vulnerability. Subsequently the patch or fix is considered for application as the vulnerability enters either the “on rails” patch process or is identified by the VM process as requiring remediation. This is the point where the risk enters the area of VM program influence whereby the vulnerability is identified by assessment activities, assigned for remediation, and tracked by the workflow management process. The risk associated with the vulnerability will decrease steadily over time as remediation actions are taken and applied to vulnerable assets in the environment. Further, we generally see a plateau of the deployment of the remediation actions as we may not cover all vulnerable assets. This may be due to inadequate asset management, therefore we may not know of all assets that require remediation, or exceptions that need to be accepted due to operational issues encountered when applying the corrective action.
While it may be tempting to utilize preventive controls to mitigate the risks imposed by vulnerabilities, a challenge exists in tracking and maintaining which controls are effective in preventing exploitation and the current completeness of coverage and operating state of the controls. Each vulnerability would require the examination of the controls deployed, their effectiveness against the threat, and their current deployment and operating status. It is not common to see preventive controls used as a risk reduction method with quantifiable results in all but the most mature and robust VM programs.

**VM in the Context of Threat Intel, Monitoring, Alerting, and Response**

It is also important to understand how a VM program fits into the other complimentary services and processes generally served out of the information security program. This view will allow us to understand how a robust VM program benefits the overall information security program. The VM program serves a vital function in a true services-integrated and threat-intelligence led security program in that it feeds the current vulnerabilities and disposition of assets as contextual information to inform adjacent processes. To further illustrate this concept we consider the model as defined below:
As we see from the graphic above, no one process is independent of the others and iterative improvements in the different process areas will result in an increase in maturity for the entire program. As the focus of this paper is on the VM program we can see that VM takes feeds from the asset management process in terms of asset ownership, criticality, and sensitivity. This information is used to identify asset owners for remediation actions as well as risk re-ranking of identified vulnerabilities based on the value of the asset. Further, VM informs the cyber threat intelligence process by feeding in the current status of vulnerabilities in the environment and the threat intelligence process feeds information into the VM process regarding the highest risk vulnerabilities based on the analysis of data through the threat intelligence process. Likewise, VM feeds the threat intelligence process with information about the current risk posture of the environment as well as supplying
data that will allow the threat analysts to quickly identify high risk vulnerabilities and their pervasiveness in the environment as data regarding exploits and their associated vulnerabilities is fed into the program. Finally, the output of the VM program is fed into the security alert analysis and response processes that can be used as contextual information about assets and their disposition during analysis and response activities.

Examples of contextual information provided by the VM program include the current set of vulnerabilities (at last assessment), current configuration information, other gathered information such as installed applications and patches, and asset attributes such as ownership, criticality, and sensitivity. This contextual information is critical to ensure that the right decisions on reducing risk, monitoring the environment, and responding to alerts and incidents are made.

**Historical Challenges of VM Programs**

Organizations that have attempted to build VM programs in the past, and have been unsuccessful, have often been plagued by a variety of challenges or mis-steps that lead to an ineffective program or the adoption of a robust VM program. It is important to highlight some of these challenges as it will assist an organization that is creating a VM program from the ground up with not making some of the same mistakes of the past. Some of the challenges have been focused around general organizational issues that include:

- A lack of authority to conduct assessment activities leading to incomplete vulnerability data or coverage;
- A lack of assignment of which group will be responsible for the various aspects of the program including identification, remediation, and reporting (RACI);
• A lack of understanding of the different vulnerability treatment options that may be applied and the differences between a patch management and vulnerability management program, and;

• Contention or lack of an agreement between the group responsible for vulnerability identification and remediation assignment (information security) and the group responsible for application of the remediation (information technology).

The assignment of responsibilities and their formal acceptance by all parties involved in the VM program is key to building and maintaining a successful program. It is not uncommon to find that the information security function or group is responsible for conducting assessment activities (scans) which lead to the identification of vulnerabilities. This group is also often tasked with assigning remediation activities to other teams or groups which reside outside of the information security function. Finally, the same team that identified vulnerabilities and assigned remediation is also often responsible for reporting the status of the program to management. One attribute of underperforming VM programs is that a general mis-trust between these two groups exists and leads to contentious interactions, rejection of identified vulnerabilities, fear that management will view reports generated by this program as a failure of those responsible for remediation, and lack of action that results in the organization absorbing an ever increasing amount of risk.

Another challenge, that spans both organizational and technological areas, is Asset Management (AM). Immature AM programs often lead to issues in gaining full visibility into the environment or ensuring that assessment activities cover all of the IT environment. In addition to potentially missing systems during assessment, ineffective AM plays a key role in limiting our ability to risk re-rank vulnerabilities and assign remediation actions to the proper asset owners. The most noticeable impact of an immature AM program is that the
VM program may identify assets with vulnerabilities that have no owner of record in the AM program. Unfortunately, this leads to remediation actions being assigned to a generic bucket or to no one at all, leaving the vulnerability to exist in the environment for extended periods of time.

In addition to the organizational challenges there are technological challenges to building a robust vulnerability management program that may play a role in limiting program maturity. These challenges can be in the form of a lack of effectiveness of the assessment technology due to configuration or missing or lacking features of the tools. Others may be related to the lack of integration between assessment and workflow management or data enrichment solutions. The focus of this paper, being that it is on vulnerability management, may lead the reader to believe technological challenges are solely related to the assessment technology. While assessment technology is certainly an element of the VM program it is not the only technology and other technologies including workflow management, IT governance, risk and compliance solutions, asset management technologies, and network path determination tools must also be considered. Some examples of the challenges centered on the technology used in VM programs includes:

- A lack of assessment scan coverage of systems and applications due to limiting factors such as firewall/filtering blocking scans, misconfiguration of the technology, or lack of periodic asset discovery scanning to ensure all network segments are covered;
- A lack of scan depth of assessment due to the use of unauthenticated versus authenticated scans;
• A lack of coverage in the breadth of technologies and applications outside of the base operating systems (i.e. databases, COTS applications, web applications, etc.) that are included in our program;
• The failure to include configuration assessment with vulnerability assessment scans where appropriate, and;
• The lack of a standardized output format or interfaces that can easily be used by other technologies for both data enrichment as well as workflow tracking.

It is important to note, specific to the assessment technology, that the issues center around four distinct themes. These themes are a lack of the coverage, lack of depth, and lack of comprehensiveness of the scans as well as the standardized output of the results of the assessment activities. The theme of coverage ensures that all assets included in the program are scanned on a periodic basis and vulnerabilities are reported. The theme of depth is that the scans are able to identify vulnerabilities on the assets at the right level through authenticated scans. The theme of comprehensiveness ensures that all technologies, in addition to the operating system, are included in the assessment process. Finally, the theme of standardized output ensures that the data generated by the assessment tools is able to be used by other technologies for data enrichment and workflow tracking.

One final set of historical challenges that have limited the maturity of VM programs center around the lack of a defined and automated workflow and tracking mechanism, the lack of prioritization of the vulnerabilities that need to be remediated, and the general deficiencies of an organization’s asset management program. Historically, VM programs have been setup as open-loop processes whereby scans are conducted and remediation assigned but never tracked to ensure that all vulnerabilities requiring remediation have reached a
determined state. Additionally, those responsible for conducting scans often don’t discriminate when assigning the criticality of the vulnerability, outside of the assessment vendor supplied ratings, when assigning remediation actions based on vulnerability criticality. The rating assigned to any given vulnerability is that of the "vulnerability in a vacuum" and not in the context of if and how it is being used in the wild, which systems are most critical or vulnerable due to their use and network location, or if compensating controls exist that diminish the risk due to effectiveness of the control or increase the risk due to ineffectiveness. Finally, if we are to enrich the vulnerability data we are generating to focus on the most critical assets, and to ensure that remediation actions are assigned to the appropriate owners, then we will need to rely on an asset management system that not only tracks the standard attributes of location and ownership of assets and applications, but also the criticality or sensitivity of the assets included in our program. Historically, organizations have not fared well when it comes to managing assets and maintaining accurate inventories. In addition to inaccurate information on assets it is unlikely, outside of regulatory compliance requirements, that the organization has assigned criticality and sensitivity ratings to its assets.

This set of challenges leads to a number of issues that directly affect the effectiveness of a VM program. The lack of a closed-loop process often leads to an ineffectiveness process due to the lack of tracking which leads to a lack of reporting and metrics which makes the
program effectiveness nearly impossible to determine. The lack of discrimination of the risk of vulnerabilities often leads to excessive workloads for those responsible for remediation without the benefit of ensuring that the highest risk vulnerabilities are addressed first. Finally, the lack of accurate asset information affects both our ability to risk re-rank vulnerabilities based on asset criticality and our ability to assign remediation to those responsible for the assets that requires action.

The Path Forward

Despite the historical challenges it is possible to build an effective, risk-reducing and robust VM program given the proper organizational and operational responsibility assignments, the proper assessment technology and configuration, the ability to create a closed-loop process to enable tracking, remediation workflow and accountability, and the enrichment of vulnerability data to ensure that the highest risk vulnerabilities are being addressed first. In order to design an advanced VM program this paper will divide the discussion across the following program elements:

- Organizational and operational responsibilities
- Assessment technology considerations
- Vulnerability data enrichment sources
- Workflow, reporting and metrics considerations

Each of the above program elements will be discussed with consideration paid to the possible solutions available. While the paper will not focus on a specific set of technologies or vendor solutions it may use industry-leading technologies as examples and in the discussion of potential solutions.
Organizational and Operational Responsibilities

As noted in the historical challenges section, it is crucial to obtain alignment with the various teams and groups that will carry responsibilities in the VM program and that these responsibilities are clearly defined. The easiest way to achieve consensus is through management level support, the development of a RACI (responsible, accountable, consulted, and informed) model and the creation of a cross-functional VM team. Assuming management level support for the program exists our next step is to build out the RACI model, and as such an example RACI model is presented below:

<table>
<thead>
<tr>
<th>Task</th>
<th>Information Security</th>
<th>IT Operations</th>
<th>Risk Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vulnerability scanning and identification</td>
<td>A/R</td>
<td>-</td>
<td>I</td>
</tr>
<tr>
<td>Asset management program/contextual data feed</td>
<td>C</td>
<td>A/R</td>
<td>-</td>
</tr>
<tr>
<td>Threat intel/vulnerability risk re-ranking</td>
<td>A/R</td>
<td>I</td>
<td>I</td>
</tr>
<tr>
<td>Assignment of remediation</td>
<td>A/R</td>
<td>I</td>
<td>I</td>
</tr>
<tr>
<td>Treatment of assigned vulnerabilities/remediation actions</td>
<td>I</td>
<td>A/R</td>
<td></td>
</tr>
<tr>
<td>Tracking of remediation actions taken and escalation requests</td>
<td>A/R</td>
<td>C</td>
<td>I</td>
</tr>
<tr>
<td>Reporting of program KPIs and metrics</td>
<td>A/R</td>
<td>I</td>
<td>I</td>
</tr>
<tr>
<td>Program enhancement requests</td>
<td>A/R</td>
<td>C</td>
<td>C</td>
</tr>
</tbody>
</table>

As we can see from the RACI it becomes clear that vulnerability identification, risk re-ranking, and assignment of remediation is the responsibility of the information security function. IT Operations is responsible for asset contextual data feeds into the program (as asset management generally falls to IT Operations) and treatment or application of remediation actions. Information security picks the flow back up to track the actions taken and to report out the metrics or key performance indicators (KPIs) of the effectiveness of
the program. Meanwhile, the Risk Management function remains informed throughout the process and uses the metrics and KPIs to further drive maturity in the program through enhancement requests. The following graphic shows the relationships among the various tasks and groups involved in the process:

![Vulnerability Management Program Workflow](image)

Figure 7. Example flow of the VM program for IT Security and IT Operations

The creation of a cross-functional team including representation from Information Security, IT Operations, and Risk Management is key to ensuring that cooperation and coordination remain in place initially and on an on-going basis. It is also important that this cross-functional team collaboratively determines the success criteria of the program as this will be used as a measure of the program’s effectiveness. In addition, IT Operations needs to ensure it conveys to the team any limitations in its ability to provide asset contextual
information and in the current patch and configuration management processes or technologies. It is not uncommon for IT Operations to work with Information Security at this point in the development of the program to identify potential solutions to both process and technology issues affecting remediation actions. Finally, the joint team must gain consensus that the goal of the program is to measurably reduce risk in a logical fashion by getting identified vulnerabilities into a determined state through a defined and measured process.

Assessment Technology Considerations

As important as process is in creating a successful VM program, the right assessment technologies enable the full value of the process by accurately and comprehensively identifying vulnerabilities in the environment. To begin our consideration of assessment technologies it is important to understand the attributes of the program that are being supported by the technology. By framing the discussion of assessment technology around these attributes we will see that the attributes will drive technology selections that ensure the goals of the program are being met. The attributes and their associated definitions are:

- **Breadth** – The coverage of assessment activities in terms of the technologies and layers that are included;
- **Depth** – The level of detail resulting from assessment activities as we assess a particular system or type of application, and;
- **Comprehensiveness** – The completeness of coverage of assessment activities in terms of percentage of systems of a particular type that are included in the program.

Focusing first on the **Breadth** of coverage, organizations with newly formed or immature programs tend to focus solely on the operating system layer while a comprehensive VM
program focuses on most all layers of the delivery stack. For our advanced program we will need to ensure we have vulnerability and configuration coverage over the following layers (product logos for illustrative purposes only):

![Program layers and illustrative technologies by layer](image)

Figure 8. VM Program layers and illustrative technologies by layer

One item to note is that the application development layer, which in our model sits between the application delivery (i.e. presentation) and application integration layers, will also be considered for assessment activities. However, we assume that vulnerability identification in custom developed applications will be detected through dynamic application security testing (DAST) or a combination of static and dynamic testing called interactive application security testing (IAST) which allows for a combination of static code assessment plus the
interactivity allowed in dynamic application testing. Pure static code analysis testing (SAST) in our model is incorporated as part of the secure development process, and as applications are being developed they will be assessed and treated in a complimentary process to our VM program. Only vulnerabilities in application code that affect production systems and data will be assessed and addressed in our steady-state VM program.

It is highly unlikely that a single assessment technology will adequately cover all layers. This problem is exacerbated if we include configuration assessment as part of our comprehensive VM program. However, some assessment technology has functionality that allows for vulnerability and configuration assessment of a single, or in some cases multiple layers of our model.

The next factor we consider is the Depth of coverage of our assessment technologies. The concept of depth is that the assessment tools are able to assess the layer at a level that allows the technology to provide the right level of detail in its output. There are some common factors or issues when the right level of depth is not achieved, these include:

- Performing unauthenticated scans that do not allow the technology to access all areas or parts of the systems and applications for assessment;
- Scanning applications dynamically without providing user rights on the account being used by the assessment technology, effectively limiting its access to all parts of the application, and;
- Selecting assessment technologies that do not have the capability to achieve or allow for the right level of depth.

As we can see most of the issues regarding depth relate back to necessary access to the system, device or application. This brings up a critical point when both selecting and
enabling assessment technology at the various layers mentioned above in that the technology must be both fit-for-purpose and enabled through a process that ensures that the depth of assessment is adequate. As we on-board a new assessment technology it is helpful to follow a pre-defined process that guides our decision on the technology as well as to ensure that we address the issue of assessment depth. An example process is as follows:

![Figure 9. Example process to incorporate a new system, type or layer.](image)

As we can see from the example process, two of the sub-processes around account creation will require tight integration with the IT Operations or Access Management teams to ensure that the account created is appropriate for the technology being deployed. It is vital for Information Security to be very prescriptive in the requirements of the access level required, account permissions, and authorization levels of the account so the principle of least privilege can be maintained. If the required level is not know it is recommend that there is a collaborative discussion between the system operators and Information Security whereby Information security provides a set of requirements for access that the system operators are able to address by layering in their expertise of the platform. This helps to stop system operators from attempting to adhere to vague requirements for access and reduces the amount of cycles spent attempting to “get access right”.

---

Building an Advanced Vulnerability Management Program | Deron L. Grzetich
Finally, we need to address the **Comprehensiveness** of our assessment activities. Once a specific layer or category of systems has been chosen for assessment and inclusion in the program it is imperative to ensure that these systems are included in scheduled assessment activities. This is perhaps the most difficult attribute to address in that it is highly reliant on the level of maturity in our asset management program. This is also not an attribute that is specifically addressed by the assessment technology we chose and is more of a program-level issue, however it will be dealt with in this section as the technology is a feed into ensuring the comprehensiveness of the program.

That being said, it is not uncommon to utilize the output of the VM program and assessment activities to act as a counterbalance or check of the accuracy of our asset management program. As such, we should include a process to deal with assets that are in our environment and not being scanned (i.e. missed by the VM program) as well as assets that are identified through assessment activities that are not in asset management (i.e. live assets that may not be actively managed). In a robust VM program we would acquire a feed of asset information into a central repository that will be used to match the associated vulnerabilities to the asset. Depicting this as a flow diagram we would see:
The key questions that need to be addressed are:

- Do we have an asset inventory that was supposed to be assessed yet wasn’t, and;
- Do we have an asset that was assessed that was not in our inventory from asset management?

Both of these decision points, when the answer is affirmative, need to be addressed by sub-processes of our asset and vulnerability management programs. It is understandable that non-mobile assets (i.e. network equipment, racked servers/applications, etc.) will be easier
to track and reconcile as part of this process, however options such as host tracking (i.e. utilizing a watermark on the system) for network-based assessment scans or the use of host-based scanning technology may be useful to address mobile assets (i.e. laptops and mobile devices). The net of the discussion around the comprehensiveness of assessment activities is that we need to develop processes that address or correct assets not being assessed as well as assets that are not in asset management. And, although we address metrics and KPIs in a later section, we are able to identify a few key metrics from this flow that could be valuable as measures of operational effectiveness of both asset and vulnerability management.

Vulnerability Data Enrichment Sources

In the organizational considerations section above we have a step in the flow diagram that required that the Information Security function risk re-rank the identified vulnerabilities that need to be addressed. In this section we consider the various sources and approaches to re-ranking that we may apply to the program. To be clear, we are attempting to differentiate between the intrinsic risk rating of the vulnerability (i.e. risk in a vacuum) versus the risk as we re-rank by adding in contextual information (i.e. exploits exist for the vulnerability, value of the asset, network accessibility, etc.) to ensure that the highest risk vulnerabilities are remediated first. To better understand the problem and potential solutions, consider the following scenario:

- Organization A is a global insurance company with offices in 70 countries, approximately 85,000 end points, 22,000 servers, and 35 internet POPs;
- Organization A utilizes a vulnerability assessment tool that scans the environment and reports on identified vulnerabilities as part of a new VM program;
• Identified vulnerabilities are kept at their default risk rankings/values and we assign ALL severity 4-5 (on a 1-5 point scale) for remediation;

• This results in over 3,000,000 vulnerabilities (1,500,000 considered high risk by the assessment tool) that need to be addressed on over 100,000 systems (80,000 are affected by the high risk vulnerabilities noted above) throughout our enterprise assigned to IT Operations for remediation;

• To address the 3,000,000 vulnerabilities we would need to apply 1000 different patches (375 if we consider only high risk vulnerabilities), and;

• IT Operations responds to this remediation request by stating that it will take approximately 12-18 months to perform full remediation.

To further illustrate both the problem as well as to see how risk re-ranking may be able to solve our problem we consider the graphs of both the vulnerability count and host count for the categories of all identified vulnerabilities as well as a reduction by only focusing on the high risk vulnerabilities as ranked by our assessment technology. In addition we graph the total count of patches that would need to be applied to address the categories of vulnerabilities we have identified. It is important to note that the patch count may be slightly inflated as a patch for the various operating systems included in our assessment activities may be counted more than once. For example, a patch that remediates a vulnerability on Windows Server 2003 will not be the same as what applies to Windows Server 2008 or 2012 and will be counted as 3 patches that are required to address a single vulnerability across three platforms. The graphs will be updated later in this section to illustrate the impact or value of our re-ranking process.
The question that falls from this scenario is, if we can only remediate a fraction of the vulnerabilities on a monthly basis then how do we focus on vulnerabilities that create the largest amount of risk to our environment? The answer to that question is not likely to be supplied by IT Operations, nor should it, and needs to be considered and answered by those
responsible for the identification of vulnerabilities (i.e. Information Security). This is where we employ a methodology to re-rank through the addition of contextual data and what follows are some ideas on how to approach this problem as well as the sources of information that will aid this process. A few ideas that we will explore around risk re-ranking and data sources includes the use of:

- Patch availability and inclusion in exploit kits
- Asset values and criticality/sensitivity ratings
- Network accessibility or the “Jump Box problem”
- Effectiveness of currently deployed mitigating controls

One of the most basic, but perhaps most underutilized pieces of data, is considering the vulnerability in the context of patch availability as well as the use of the vulnerability in common or public exploits or crimeware/exploit kits. If a vulnerability does not have an associated patch release then it becomes difficult to mitigate the risk of the vulnerability as mitigating controls would need to be configured or implemented, or configuration changes be made to the affected systems. Therefore, as we begin to build our program we focus on the vulnerabilities that can be treated through patch application first. What we see from this action is that we have a reduction in the total number of vulnerabilities affecting our organization, the number of affected systems, and the total number of patches that need to be applied.

An additional piece of contextual information that is also very useful at this stage is the inclusion of data around the vulnerability and the existence of an exploit that is publically available or included in the various crimeware exploit kits that are readily available for purchase on-line. In addition to the patch being available or the exploit being part of a common exploit kit we also have a consideration of the use of exploits in 0-day attacks.
There is some differentiation between exploit kits and 0-days in that exploit kits will generally contain exploits from both current time (0-day) plus exploits from prior years that remain effective due to inadequate 3rd party application patching processes.

As an example, the following is a sample list of the top exploit kits from 2013 and their associated vulnerabilities (note: according to the author, limited to a sample of exploit kits still in circulation and/or active):

<table>
<thead>
<tr>
<th>CVE</th>
<th>Technology</th>
<th>RedKit 2.2</th>
<th>Nuclear 3.x</th>
<th>Fiesta</th>
<th>Blackhole</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVE-2010-0188</td>
<td>Adobe</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>CVE-2011-3402</td>
<td>MS XP-2008</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>CVE-2012-0507</td>
<td>Java</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>CVE-2012-1723</td>
<td>Java</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>CVE-2012-4681</td>
<td>Java</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>CVE-2012-5076</td>
<td>Java</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CVE-2013-0074</td>
<td>MS Silverlight</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>CVE-2013-0422</td>
<td>Java</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CVE-2013-0431</td>
<td>Java</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CVE-2013-0634</td>
<td>Adobe</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CVE-2013-1493</td>
<td>Java</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CVE-2013-2423</td>
<td>Java</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>CVE-2013-2460</td>
<td>Java</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CVE-2013-2471</td>
<td>Java</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CVE-2013-2551</td>
<td>MS IE</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CVE-2013-3918</td>
<td>MS IE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 12. Table of sample of exploit kits, associated exploits and affected technologies
For clarity, exploit kits allow an attacker to utilize multiple exploits that affect multiple technologies in an attempt to compromise a system and gain control. The target system only needs to browse to an infected site, or be redirected through iFrame redirects, main-in-the middle, SEO injection, or other means to get the target to load the exploiting site in their browser. Once compromised the target system checks into the exploit kit's console and is available for use by the attacker. The screenshot below is an example from the Blackhole exploit kit's management console:

![Blackhole exploit kit interface](image)

**Figure 13. Example of the Blackhole exploit kit interface**

In addition to exploit kits, many attacks in 2014 and preceding years relied on 0-day exploits which are exploits that were not discovered by researchers or the product vendors and were only discovered through incident response and forensics once they were used in the wild to compromise systems. If we examine the list of 0-day exploits released in 2013 that were used to compromise systems as part of coordinated attacks we see only a slight
overlap with the vulnerabilities included in common exploit kits. However, it is not uncommon to witness former 0-day exploits to be included in exploit kits once the details of the vulnerability and associated exploit are released. Below is a set of common vulnerabilities that were the subject of 0-day attacks in 2013. Where possible we have included the affected technology and associated campaign if known and highlighted the vulnerability in red if it was included in an exploit kit from Figure 12 above (Source – Microsoft BlueHat v13 2013, presentation by Lin and Bu of FireEye Research Labs):

<table>
<thead>
<tr>
<th>CVE</th>
<th>Technology</th>
<th>Campaign/Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVE-2012-4792</td>
<td>MS IE</td>
<td>CFR Watering Hole</td>
</tr>
<tr>
<td>CVE-2013-0422</td>
<td>Java</td>
<td>Crmeware</td>
</tr>
<tr>
<td>CVE-2013-0634</td>
<td>Adobe</td>
<td>LadyBoyle Campaign</td>
</tr>
<tr>
<td>CVE-2013-0640</td>
<td>Adobe</td>
<td>Pwnie Award</td>
</tr>
<tr>
<td>CVE-2013-1493</td>
<td>Java</td>
<td>JP Targeted</td>
</tr>
<tr>
<td>CVE-2013-1347</td>
<td>MS IE</td>
<td>DOL Watering Hole</td>
</tr>
<tr>
<td>CVE-2013-1331</td>
<td>MS Office</td>
<td>South Asia Targeted</td>
</tr>
<tr>
<td>CVE-2013-3163</td>
<td>MS IE</td>
<td>NGO Watering Hole</td>
</tr>
<tr>
<td>CVE-2013-3893</td>
<td>MS IE</td>
<td>Operation DeputyDog</td>
</tr>
<tr>
<td>CVE-2013-3897</td>
<td>MS IE</td>
<td>KO/JP Targeted</td>
</tr>
<tr>
<td>CVE-2013-3906</td>
<td>MS Office</td>
<td>Multiple Campaigns</td>
</tr>
<tr>
<td>CVE-2013-5065</td>
<td>MS Windows</td>
<td>Unknown</td>
</tr>
</tbody>
</table>

Figure 14. Table of 2013 0-day vulnerabilities and associated campaigns

If we generalize using this small sample of vulnerabilities (totaling 25 different vulnerabilities) and graph the associated count of vulnerabilities and percentage of the total by technology we find:
While this sample is artificially small we are able to draw a few initial conclusions from the analysis of the data provided and how it may be applied to our VM program. First, we can’t focus our patching efforts solely on the operating system layer as 3rd party applications (i.e. Java, Adobe, MS Office, etc.) represent a significant amount of the vulnerabilities and associated exploits in both 0-day exploits from 2013 as well as their inclusion in exploit kits. This also means we must ensure we are addressing the depth issue as discussed in the technology considerations section above to ensure we are getting adequate data on the current status of our systems. Second, we may be able to apply this data in a way that allows us to focus on vulnerabilities that represent the most risk as they have exploits that are currently circulating publicly available forums.

If we apply a risk re-ranking process of first limiting our results to high risk vulnerabilities for which a patch is available and then re-ranking again based on the vulnerabilities inclusion in 0-day exploits or exploit kits we can graphically see the reduction in the amount
of vulnerabilities that must be addressed, systems or hosts affected, and the total number of patches that must be applied.

Figure 16. Graphs highlighting the reduction in vulnerabilities, hosts, and patches after risk re-ranking
While the numbers presented in this paper are representative, they were taken from actual client data as this re-ranking process was applied to a large corporate environment. What we see from the graphs is that using this re-ranking process allows us to focus remediation activities on vulnerabilities that present the most risk to our organization. The results show that we were able to reduce the workload from 100,000 to 39,000 (a 61% reduction in hosts that need to be treated), and the patches that need to be applied from 1,000 to 180 (an 82% reduction in the number of discreet patches that need to be applied). It should be noted that leading assessment technologies are incorporating this contextual data as a competitive advantage (i.e. Qualys QualysGuard) and additional cloud-based solutions have come into existence based on this data enrichment concept (i.e. Risk I/O).

Further, and as part of the sustainability of the VM program, we would also monitor for these high risk vulnerabilities on an on-going basis and ensure that remediation actions are assigned if they find their way back into the environment as this isn’t a simple one-time treatment of the vulnerabilities. As a foot note, it would be interesting to monitor for the existence of these vulnerabilities as part of an on-going security monitoring program whereby alerts are generated on the new existence of old high risk vulnerabilities would be treated as security incidents.

If we shift our focus to other data sources for risk re-ranking we find ourselves discussing the value, criticality, or sensitivity of the asset as a piece of contextual information that may be used to rank identified vulnerabilities. This is by far one of the most difficult data sources for large organizations to provide if a solid asset management program is absent. It requires that an organization understands or tracks:

- The asset's physical location and attributes, who owns the asset (at the various layers) and who manages and maintains the asset;
• The value of the asset based on the application it hosts or the data it stores and processes, and;
• The asset’s significance in terms of regulatory compliance requirements if applicable.

Gathering and maintaining this information for a large set of assets becomes too cumbersome if the organization has disparate asset management solutions or has lacked the necessary rigor to run a solid asset management program in the past. For many organizations it is unlikely that the required level of asset information will be readily available, however organizations may bridge the gap (until a robust asset management program is deployed) through a High Value Asset Inventory (HVAI) program. The goal of HAVI is to segment assets by large swaths in terms of risk so that some data is available on the highest risk assets in the organization. The output is a sort of binary measurement of the risk of an asset, classifying the top 20% of assets in terms of risk to use as a contextual data element in our re-ranking. While this data would be valuable in re-ranking vulnerabilities it is more likely to be used in a Risk Management program to assess risk levels or compliance metrics. A high risk vulnerability with an exploit in a readily available exploit kit that has a patch available is likely to require treatment regardless of the asset value.

The last two examples of enrichment data sources of network accessibility and effectiveness of mitigating controls are linked but were separated for both clarity and discussion. We may consider the network accessibility of a vulnerable system when re-ranking the risk of a vulnerability. However, this also entails that we understand the current network-layer filtering technology that is deployed along with its current configuration. For example, if we have a vulnerability that allows for remote unauthenticated code execution yet requires
access to a specific service running on a port and protocol that is filtered at the network layer from a majority of our systems (i.e. a secured filtered subnet) we may consider reducing the risk. If the asset is internet facing and accessible from any location outside of our organization we may increase the risk of the identified vulnerability. In addition, we also need to consider if a system is compromised how it could be used as a jumpbox to attack additional systems within our organization. This is based on the same ideas presented earlier regarding network accessibility. Being that inspecting the network accessibility and potential to be used as a jumpbox following compromise would be highly laborious task if completed manually, a few technology solutions have filled the gap over the past few years. Two technologies in this space are RedSeal and Skybox, however they also serve additional functions or processes such as risk assessment of firewall and network configurations and risk modeling for potential changes in the firewall or filtering rule sets applied to these technologies.

If we change focus to the effectiveness of mitigating controls it is easiest to start with an example. As such, imagine an end user’s system running a vulnerable version of Java. This system is picked up by our assessment activities and risk re-ranked to be higher due to its inclusion in an exploit kit. Network accessibility ranking is not used as this particular exploit only requires that a user have access to the internet which happens to be allowed out of our corporate firewall. However, at the host layer we happen to be running a containerization/microVM solution (i.e. Bromimum) on the system and it is active. In the simplest terms this technology allows for the vulnerable version of Java to execute in its own segmented environment from the host system and attempts at compromise do not allow for interaction of the underlying operating system. At the corporate network layer we have deployed a web proxy and an advanced malware prevention system (i.e. Z-Scalar and FireEye Web). While our proxy may miss this exploit attempt (or not have a signature for
it) we do have detection for the exploit in our advanced malware prevention. The question becomes, is the risk really as high as we ranked it based on the risk of the vulnerability in the context of inclusion in an exploit kit? For now the answer is yes just based on the fact that gathering the data from the various tools and their effectiveness against specific exploits for specific vulnerabilities is not an automated process. Not to mention that we need to not only consider the effectiveness of the tool against specific attacks we must also understand the operating state of the control. For example, what if we allow our users to disable the containerization solution? What if the user is not on the network and therefore not being protected by the advanced malware solution? It is this author’s opinion that while the inclusion of mitigating control effectiveness as a data enrichment source holds promise, that it will not be realized until the process can be automated and updated in near real time.

Finally, we end this section with a potential flow diagram for vulnerability risk re-ranking process before addressing the considerations for process workflow:
One final aspect worth mentioning would be to go back to the discussion around threat intelligence, security monitoring, alert handling and response processes and how these are interconnected. In addition to the data enrichment options in this section we would also receive input from the analysis and response team regarding the increased use of an exploit that is associated with a vulnerability that may need to be treated regardless of the flow above. In addition, we may also receive information from threat intelligence that allow us to identify exploits for vulnerabilities that are just being integrated into exploit kits. Mainstream vendors may not have the same level of insight into this type of data and once our program is operating at steady-state we may include addition decision boxes and modify the flow as presented above.
Workflow, Reporting and Metrics Considerations

Workflow, reporting and metrics are often the most underrated components of the program yet they tend to be the “glue” that holds it together and measures operational effectiveness for stakeholders and management alike. Perhaps our lack of focus in this area is due to some of the historical challenges noted above, however it is vital for a robust VM program to have pre-defined workflows for all scenarios of vulnerability treatment and a technology that allows for seamless integration between sources and service management systems.

While this was a feature that was added to many of the leading assessment technologies in the past, a robust program will likely have many sources of vulnerability information through assessment at multiple technology layers (i.e. network, OS, application, etc.). Therefore it is critical that we have a centralized repository for vulnerability information across layers and a location to enrich or re-rank through the additional of contextual information (i.e. asset value, network accessibility).

Going back to the definition and purpose of the VM program it is important to remember that the goal was to get identified vulnerabilities into a determined state. As many assume that the only determined state is remediated or not, we must also consider the alternatives and what is meant by “determined”. Some examples of the determined state include:

- Patched, configuration change, remediated, or fixed
- Exception requested/granted and issuance of proactive exceptions
- False positive finding
- Mitigated through technology or existing effective controls
For each of the above determined states we will require a defined workflow and process that will be implemented in the central tracking and workflow engine. The above processes are best displayed as a series of actions in a flow diagram across a swim lane format that outlines the groups or functions that are responsible for taking a decision or action. The process begins by assessing the environment, identifying vulnerabilities, risk ranking and so on as presented in the flow diagram below:

![Diagram](image-url)

**Figure 18. Example flow for workflow tracking**
While the first determined state and the patch/configuration management process is straight forward it needs to be enabled in our system as an automated process. The reasons for automating are that it removes the requirement that we manually track the remediation of a vulnerability as its status will be updated through a feed from subsequent assessment scans. We also need a way to enable the escalation process for vulnerabilities marked as patched that were not treated in the agreed upon timeframe.

Exception handling is broken into two categories for clarity. First, our standard exception process is enabled when IT Operations or the group responsible for remediation identifies an operational issue through their defined testing process (i.e. UAT) after the treatment of a vulnerability is applied. This process workflow would end with sign-off by an individual at the right (i.e. executive) level of the organization has the authority to sign-off on the risk as well as be responsible for the risk on an on-going basis. This also allows for reporting of the excepted vulnerabilities that would be useful for risk management, audit and compliance, and the security analysis and response teams. The second exception type is the “proactive” exception. This is invoked when IT Operations proactively identifies an operational issue with a treatment option (i.e. patch application) as part of the "on rails" patching process. Allowing the team responsible for treatment the option to proactively seek approval for exceptions will cut down on the noise generated by the system and frustration level of all involved in this process. Again, this exception would follow the standard sign-off process and would be included in reporting as an exception as mentioned above.

There is also the issue of false positives that make their way into the workflow system. While it is the goal of the assessment team to reduce false positives to a manageable level, by allowing those teams responsible for remediation the option to suggest that the vulnerability is a false positive and provide proof we open up yet another communication
channel. Once the false positive is verified by the assessment team they may make modifications to the assessment tools or process as necessary to further eliminate false positives (i.e. noise) in the system. Tracking false positives in a centralized system also allows for another data point on the operating effectiveness of the overall program.

Finally, we have the “no action” process which literally means that no action will be taken as the remediation team feels that a mitigating control is effective and no treatment is required at the current time. It is anticipated that this process flow would be the least used in favor of treatment/remediation, exception, or false positive remediation processes. Early in the build out of VM programs it is likely that we will limit our processes to the three mentioned above and allow for the documentation of the mitigating control through the standard exception process. In more mature programs we would consider including this process as the risk of an untreated vulnerability versus one that is mitigated by a compensating control presents different levels of risk.

We end this section with a discussion on the tracking of operational metrics, or key performance indicators (KPIs), and risk metrics through key risk indicators (KRIs). While immature organizations often confuse the two type of metrics they are distinct in what they convey to the audience. KPIs measure the operational metrics of a process or program and offer insight into process inefficiencies and effectiveness that may require corrective action. It also allows us to track the effectiveness and to a defined goal state and examine the root cause of deviations in process effectiveness and the overall effect of process changes. KRIs on the other hand track the indicators of risk which give the organization a sense of its operating risk as measured by the program. By understanding how much risk we are currently carrying (i.e. the sum of identified and untreated and excepted vulnerabilities) we are able to view both the current total risk as well as how risk trends over time. This metric
acts as yet another indicator of the effectiveness of the program at reducing risk in our environment which is the ultimate goal of our VM program.

It is not uncommon to see immature VM programs only report on one metric, and that metric is the total number of identified vulnerabilities. This number is usually very large and they attempt to show the effectiveness of the program through a reduction in the overall count of vulnerabilities. This metric is also not specifically a KPI or a KRI in any sense which is not the goal of a metrics program. While the metrics that are enabled and tracked as a program matures will likely evolve as more data and data types are available, we present some options for metrics to consider as the program is built. Starting with some examples of KPIs:

- **Comprehensiveness** – The measure of comprehensiveness of coverage for each layer included in our program in terms of the percentage of assets assessed as part of the normal assessment process.

- **Depth** – The number of systems with successful authentication (where required) that allows the proper level of depth in our assessment activities by percentage of successful versus failed authentications.

- **Completion** – Tracking the successful completion of scheduled assessment activity as a raw number of failed versus successful to identify issues with oversubscription of a particular scanner or network segment that may require remediation.

- **False Positive Rate** – The rate or rejection, both by the assessment team's analysis and through the false positive workflow process of verified false positives tracked as either a percentage of the identified positive to false positives or a raw number. The goal is to limit the number of false positives that make their way into the remediation workflow process and those generated by the assessment technology.
• **Layer Coverage** – Tracking in terms of the layers that should be included in the future state program versus what is currently covered. It is unlikely that this metric will rapidly change over time, however tracking the progress towards the future state of the program informs management on progress and drives upcoming or planned activities.

• **Re-ranking Process Effectiveness** - Tracking the reduction in workload as a percentage reduction in the patches that need to be applied (versus all identified) and the hosts requiring remediation (versus all hosts identified with vulnerabilities). This metric helps to convey the effectiveness of the ranking process and allows for remediation of the highest risk vulnerabilities.

If we change focus to KRIIs we consider:

• **Remediation Success** – Tracking the success of remediation actions through a comparison of required remediation to successful remediation. This is an indicator of effectiveness of the remediation process but also shows residual risk trends through non-remediation (or success through remediation).

• **Exceptions Requested and Granted** – Percentage of vulnerabilities identified where exceptions were requested as well as granted. This can be tracked as a percentage of the identified for remediation over the exception requested numbers, and as a percentage of requested to granted to highlight when the exception process may be being abused.

• **Risk Trend of Tier 1 Vulnerabilities** – The highest ranked vulnerabilities, which we will refer to as tier 1, will enter the system for remediation. If the re-ranking process allows for asset value to be included in the ranking process we may be able to track the risk trend over time by tracking the number of tier 1 vulnerabilities
identified to those remediated over those identified. A percentage works best in this scenario as the number identified is likely to change over time. It is also possible to use another measure such as tier 1 vulnerabilities remediated by host as an average per host number.

- **Risk Trends by Business Unit, Layer, or Geography** - Tracking of the tier 1 vulnerability trends and total counts by business unit, layer of technology, or geography. This will highlight failures of remediation actions across the various business units and geographies to highlight process issues on a per geography or business unit basis. We can also consider looking at the tier 1 trends and count by layer or specific technology to identify trends in risk associated with a specific layer that may highlight issues with specific processes or mis-management of specific systems or layers in terms of vulnerability remediation.

- **Overall Risk Score by Layer** – While we are opposed to having a “stoplight” graphic, a single risk indicator number per layer may be a good metric for a dashboard that is a conglomeration of some of the above metrics that result in a single score. Given the sheer number of options in calculating this score we will refrain from making a specific suggestion here.

Finally, how these metrics are gathered and reported is very important to the maintenance of the program and the success of the program in terms of meeting the goals and mission. All of the above metrics become available with a proper workflow management system and engine from which we can pull automated accurate numbers for calculation, comparison, and display. This also allows for trending over time across all layers as we utilize a single centralized management system. The reporting from the system should also be fit-for-purpose for the audience and level. Care must be taken in creates the reporting and dashboards that fulfill the needs of the recipients with the
most technical information being provided to the technical teams and management level metrics being displayed to management in terms of a score card.

Conclusion

This paper is just the beginning in terms of outlining the requirements for building an advanced vulnerability management program. Topics such as workflow management integration with external systems, the underlying assessment technologies that should be considered on a per-layer basis, and detailed process flows need to be addressed. In addition, we also need to consider the steps an organization would take to deploy an advanced program in a phased or iterative approach. All of these topics are fair game for inclusion in a revised version of this report.

That being said, we set out to address the historical challenges of building VM programs and their general ineffectiveness and describe the path forward. This is achieved by ensuring the program has a solid strategy and foundation in terms of the layers we include, the assessment technology we choose, and the overall management of the program through workflow and metrics tracking. Iterative improvements and a phased approach are going to be required to build a robust program as the sheer amount of change an organization would need accept to deploy the program as a “big bang” is more than most can handle. One of the most important steps in building a program is to not make the same mistakes of the past when setting the foundation and making iterative improvements. This report endeavored to address these challenges and present some potential solutions.