Third Party Software Security Working Group

Appropriate Software Security Control Types for Third Party Service and Product Providers

Version 2.3 / October, 2015

FINANCIAL SERVICES | Information Sharing and Analysis Center

Contents

Working Group Model

The FS-ISAC Product & Services Committee is responsible for defining, developing and delivering products and services that solve critical business problems for FS-ISAC members. A common practice used for developing new products and services is to form a working group of members to focus on the definition and rationale on behalf of member firms.

According to PwC, detected security incidents have increased 25% this year, while the average financial costs of incidents are up 18%.[1] It is increasingly critical that organizations understand the risk associated with sharing data with third parties; however few organizations take this step. In fact, only 20% of organizations evaluate the security of third parties with which they share data or network access more than once a year.[2] This trend of ignoring the risk posed by third parties cannot continue.

For this reason, the FS-ISAC Product & Services Committee asked several member firms to form the Third Party Software Security Working Group to determine what additional software security control types would be appropriate to add to vendor governance programs. The Third Party Software Security Working Group was established with a mandate to analyze control options and develop specific recommendations on control types for member firms to consider adding to their vendor governance programs.

The members that participated included:

Working Group Members

1. **Scott Anderson,** Citi
2. **Gerry Brady,** Morgan Stanley
3. **Mark Connelly,** Thomson Reuters
4. **John Cooney,** FIS Global
5. **Josh Corman,** Sonatype
6. **Jolyon Clulow,** Deutche Bank
7. **Mahi Dontamsetti,** State Street
8. **Don Elkins,** Morgan Stanley
9. **Paul Fulton,** Citi
10. **Keith Gordon,** BofA
11. **Simon Hales,** HSBC
12. **Royal Hansen,** Goldman Sachs
13. **Brian Heemsoth,** Aetna
14. **Chauncey Holden,** RBS Citizens Bank
15. **David Hubley,** Capital One
16. **Wayne Jackson,** Sonatype
17. **Richard Jones,** JP Morgan Chase
18. **Amit Khosla,** FIS Global
19. **Malcolm Kelly,** HSBC
20. **Andrew Lavender,** UBS
21. **Tim Mathias,** Thomson Reuters
22. **Greg Montana,** FIS Global
23. **Ben Miron,** GE Capital
24. **Anne Nielsen,** Veracode
25. **Rishikesh Pande,** Citi
26. **Jim Routh,** Aetna
27. **Bill Schineller,** Blackduck Software
28. **David Smith,** Fidelity
29. **Hinrich Voelcker,** Deutche Bank
30. **Chris Wysopal,** Veracode

---

[1] PWC Global State of Information Security Survey 2013 – October, 2013
[2] PWC Global State of Information Security Survey 2013 – October, 2013

Executive Summary

Third party software is the new perimeter for every financial institution. According to Gartner, "since enterprises are getting better at defending perimeters, attackers are targeting IT supply chains."[3] Further, recent breach reports such as Verizon's Data Breach Investigations Report underscore the vulnerability of the application layer, including third party software. This new perimeter of third party software must be addressed.

Fortunately, the majority of financial services firms and many technology vendors are investing in improving software security control practices within the lifecycle of software development to provide products and capabilities that are more resilient to attack. Pushing innovation in the marketplace while protecting information assets exposed in emerging technologies (like mobile computing or cloud services) is a continual challenge and dilemma for financial services firms. The financial services industry has historically provided leadership in the development of effective vendor management practices to reduce the risk of exposure of customer and employee information. Financial institutions have led the implementation of effective governance models for third parties providing IT products and services for over a decade. Many IT vendors have incorporated prudent risk management controls into their product development processes as a result.

Evolving vendor governance practices have helped to improve information risk management for firms applying these standards and for vendors that incorporate these standards into their product development. Codified practices like the Shared Assessments Program have had a positive effect in encouraging vendors and industry firms to work collaboratively with the goal of improving information risk management practices and better serving customers and employees. However, as the financial services industry increases their reliance on third party software services and products, we are seeing an increase in the number of breaches stemming from software vulnerabilities. This trend necessitates improved controls operating in concert with vendor management practices to advance the relationship between security and third party software service providers and commercial off-the-shelf software (COTS) vendors.

Mandate

The mandate of the Third Party Software Security Working Group is to identify control types to incorporate with vendor governance programs in order to improve information protection capabilities when using third party services and products in the supply chain for financial institutions' customers and employees.

Selecting controls to add to a vendor governance program requires collaboration between third party suppliers and financial institutions. Many software service and product vendors have adopted industry leading practices for software security embedded in their product development processes such as BSIMM (www.bsimm.com). Several leading commercial software providers have codified software security practices in an effort to encourage other providers to adopt leading software security practices through the SAFECode organization (www.safecode.org). It is the responsibility of the financial services industry to make software security requirements explicit rather than implicit. This means that a clear set of control requirements needs to be consistently communicated and applied to specified services and products.

---

[3] Gartner, "Maverick*Research: Living in a World Without Trust: When IT's Supply Chain Integrity and Online Infrastructure Get Pwned" October 2012

By aligning on these control types as an industry, financial institutions can improve the adoption rate for vendors, and ultimately can promote software security from an outlier request to a standardized norm. The objective is to have clearly defined control types that are consistently applied to enable sharing of artifacts between financial institutions based on vendor acceptance for release of the artifacts. Software security controls are an integral part of building high quality software.

If a vendor controls the development and build process, then they also are responsible for applying appropriate security controls. This is the premise upon which the Working Group built the controls for addressing third party risk.

The Working Group selected three control types for adoption by financial services and vendors, two of which apply directly to third party vendors while the third applies to the supply chain that financial institutions rely on for software development. The Working Group decided to revise this document and update the third control type (Open Source) to include a Bill of Materials (BOM) that identifies the open source code libraries that are part of commercial software packages. This third control type was expanded from the original version to include another type of open source code management that specifically relates to the acquisition and use of third party commercial software.

> By aligning on these control types as an industry, financial institutions can improve the adoption rate for vendors, and ultimately can promote software security from an outlier request to a standardized norm.

> If a vendor controls the development and build process, then they also are responsible for applying appropriate security controls.

Controls

Each control type was evaluated based on practices that were adopted and implemented by one or more financial institutions and the institutions' experiences with the control type. In all three controls, several of the Working Group members had practical experience with the implementation of the control type for third party vendors. For the second and third control types there are currently two preferred vendors that offer solutions satisfying the control requirements defined by the Working Group. The specific vendor solutions were discussed in depth by Working Group members. During these conversations members shared implementation experiences and evaluated of the effectiveness of the control type. The Working Group chose not to identify a single vendor solution for each control type but agreed to share experiences with vendor products with other financial institution members.

	Control Type	Description
1	vBSIMM process maturity assessment	A derivative of BSIMM that specifies selected practice areas of BSIMM specific to vendor supplied software and uses the vBSIMM activities to determine process maturity of the product development function of the vendor.
2	Application Security Testing	This control type identifies various methods for assessing the risk posture of an application if vendor controls are immature.
3	Policy management and enforcement for consumption of open source libraries and components	This control type identifies consumable open source libraries for a given Financial Institution, identifies the security vulnerabilities by open source component and enables the Financial Institution to apply controls or governance over the acquisition and use of open source libraries.
4	A bill of materials that clearly identifies the open source code libraries that are part of a commercially developed software package offered to financial service firms. This control includes a schema to be shared with software vendors and software tool providers that defines the format for information sharing within the BOM concept.	This control type is used exclusively for software providers and open source code management tool providers to identify consistent definitions of open source code libraries and versions to standardize the bill of materials across open source management capabilities providers.
5	Contract Language	No matter which controls an acquiring firm decides to implement with their third party vendors, acquiring firms should also work with their legal team to incorporate vendor responsibility for software security into every applicable vendor contract.

Summary of Controls

The Working Group recommends adoption of the three control types for each financial services member firm. The financial institution will determine how best to implement the control type into its vendor governance and software acquisition process and which vendor services or products to use. The first control type, vBSIMM process maturity assessment, does not require member organizations to implement any vendor product or solution, however the Working Group recommends that financial institutions participate in education and certification of software security assessment professionals by using education, training and certification services provided through FS-ISAC. The second control, binary static analysis, and third control, policy management and enforcement for consumption of open source libraries and components, both require the use of vendor services or products.

The Working Group does not endorse any of the vendors or products within the control types. The Working Group believes that each of the three control types are required for financial institution member firms to achieve third party software security, and will share information on implementation experience to help other financial institutions with adoption and implementation.

Each of the control types is covered in more depth in the subsequent pages of this paper based on the evaluation and assessment work completed by the Working Group members. In some cases, vendor management professionals will need help from software security assessment professionals to effectively introduce the control types to the vendor management program in their respective financial institutions. In the third control type (open source supply chain governance) the implementation of the control type will not likely require the involvement of vendor management professionals. The implementation is more likely to be administered and enforced by application development leads, software quality assurance professionals or architects with the application development function.

Control Type 1: vBSIMM Process Maturity Assessment

Assessing the maturity of controls integrated with the product development process is the primary focus of the first control type. This is to overall security being dependent on the aggregate assessment of controls working together as opposed to a single control (e.g. static scanning). The maturity of the security controls is a leading indicator of the potential for security vulnerabilities in a specific release of software and subsequent releases of software by the vendor. Much has been published on common practices for improving software quality and specifically security. Twenty financial intuitions use BSIMM (Build Security in Maturity Model) to benchmark the maturity of the controls within a development process or program. This is a model of observed activities or practices that number over one hundred within twelve practice areas. Financial intuitions may use different technology solutions and practices to improve software security and BSIMM provides for a way of measuring maturity across different enterprises.

For this reason, BSIMM was considered as a potential source of measuring the software security maturity of vendors providing products and services to financial institution member firms. Vendor BSIMM (vBSIMM) was derived from BSIMM activities and practices that apply to third party software development by several financial firms represented on the Working Group. vBSIMM provides a method for measuring software security maturity across vendors that use different technical tools, methods and techniques to improve software security maturity. The key is that multiple controls or "touchpoints" are required in the software development process to achieve maturity.
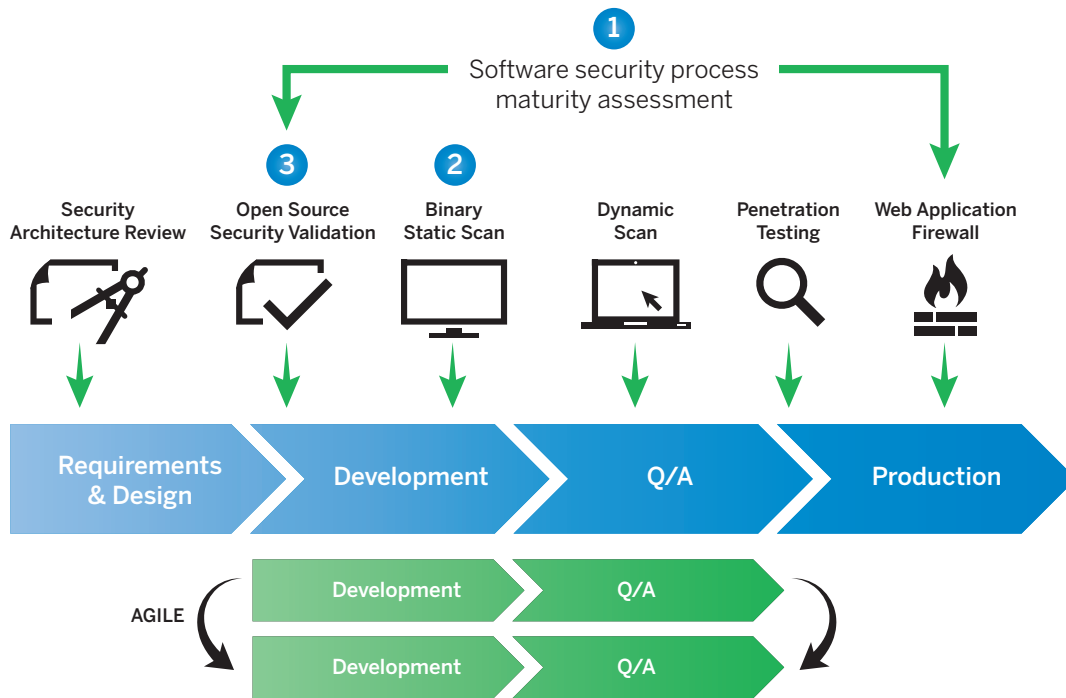

Figure 1: *vBSIMM Framework*

Practice Area 1: Threat Modeling or Applying Risk Controls in the Design of Applications and Products

A common misconception among those involved in software development is that all security vulnerabilities are introduced once the code is developed. This practice area addresses techniques for approaching the risk of building functionality in the design of the application or product. Often significant security flaws in a web or mobile application have their root in a specific design decision as opposed to errors in coding practices leading to an implementation defect, such as cross-site scripting. Design flaws often have a far more significant impact than coding flaws in a given application in both severities of the flaw and the associated cost of remediation. A common technique used to identify and address security vulnerabilities in application or product design is called threat modeling. Threat modeling involves taking a data flow diagram of proposed functionality and identifying the methods/techniques that a malicious attacker (often referred to as a threat actor) can use to break functionality or compromise credentials (called the attack surface).

Threat modeling involves taking a data flow diagram of proposed functionality and identifying the methods/techniques that a malicious attacker (often referred to as a threat actor) can use to break functionality or compromise credentials (called the attack surface).

This information can be used to consider adjustments to the application/product design to make the application more resilient to common attack vectors or methods. Threat modeling is difficult to teach to application designers and often requires an apprenticeship where one professional learns techniques and methods from another more experienced professional over time. For vendor products, this type of analysis during design may end up providing information that may drive changes to product functionality (adding security features to the product).

Decisions on where to store sensitive data and how to protect them in web and mobile applications offer an excellent example of how critical these types of design decisions are from a security and privacy perspective. A Working Group member bank noticed that formal threat modeling was only used in approximately 15% of the software vendors and service providers participating in its initial threat modeling program (see Appendix 1 for full case study). However, other techniques for considering security features in the application and for determining how to best protect information were applied by vendors and represented a higher maturity for this specific practice area. It was common for COTS (commercial off-the-shelf software) vendors to apply less formal techniques for assessing the risk during design discussions, without creating formal artifacts or threat models. The bank's assessor had to determine which of these activities were consistent with BSIMM activities and identify ways to evaluate the effectiveness of these techniques to assess the practice maturity score.

For example, when Apple developed its mobile map functionality it made a conscious choice to give users control over the use of their location information. Users must use the provided dialog box to permit their location information to be accessed. This functionality, which mitigates the security liability of obtaining user location information, probably evolved from a design discussion and ultimately a decision to convert this mitigation into user experience feature.

Practice Area 2: Code Review

Code review as a practice is one of the most commonly misinterpreted practices by COTS vendors. Code review is a well-established process or technique for improving software quality. Code reviewers will review code looking for the enforcement of development standards, proper syntax, techniques applied to comments, the use of specific statements and input/output validation approaches. The methods of applying code reviews certainly improve quality but unless the reviewer understands software security, these reviews do nothing to improve the risk caused by security defects. Code resiliency can be improved through code reviews as long as the reviewer has the requisite software security skills and the number of developers is small enough to make this method scalable.

Code review as a practice is one of the most commonly misinterpreted practices by COTS vendors.

If a development team has fewer than six developers then a manual code review by a security expert is a fundamentally sound approach to addressing security defects in the code. If there are 60 developers, then approximately 10 reviewers with the right level of security skill are required to spot security defects. Finding the required number of software security professionals to do manual code review gets more difficult the larger the development team. In practice, development teams over 50 developers need to consider other alternatives to improve security in the coding process. The use of commercial products to complete automated code reviews (code scanning) for both quality and security is increasing but it is more prevalent with financial institutions than with COTS software vendors. The Working Group developed a "rule of thumb" for product development teams over 50 developers: the vendor needed to acquire and use a commercial product to do code scanning in development to achieve a high level of maturity for this practice. Using different commercial products to do code scanning across development teams does not influence control maturity as long as a method for enforcement was in place in each case.

In practice, development teams over 50 developers need to consider other alternatives to improve security in the coding process.

Practice Area 3: Security Testing

Methods and techniques for applying security testing varied significantly across vendors. In reviewing an example of applied vBSIMM, the Working Group saw that almost all of the vendors had some methods or approach in place for system security testing at a minimum of a level 1 maturity.[4] A few of the COTS software vendors demonstrated a high level of rigor and discipline (i.e.: level 3 maturity) in their security testing process that applied to functional requirements and security vulnerabilities. Both software development and COTS vendors' demonstrated maturity in the controls they had in place. In some cases a separate Quality Assurance team handled responsibility for security testing while in other instances the developers applied testing scripts and evaluated the results with Q/A reviewers who did additional testing and analysis. Several vendors developed complex testing scripts for testing access and entitlements for their product. A handful of firms hired other third party firms to do testing and provide detailed reports on the results. As with other practice areas the financial institution must distinguish security testing from quality testing techniques in the assessment process.

Dynamic testing tools for security are not a requirement for system testing but they do make it easier to implement testing procedures for larger development teams. The Working Group found that many of the vendor firms in the example of applied vBSIMM would capture security vulnerabilities from the testing processes and prioritize the results based on risk and then assign responsibility for remediation in the bug tracking process.

Many COTS vendors have developed sophisticated processes for defect tracking and management, and some even feed security testing results into this process. Some vendors do not allow applications with security vulnerabilities to be promoted into production without approval from a senior officer (e.g. CTO). A few software vendors include manual penetration testing as part of their security testing process, however most software vendors only used manual penetration testing services upon client request.

Practice Area 4: Manual Penetration Testing

This was the practice area that was the most familiar to all of the software vendors participating in the example execution of vBSIMM reviewed by the Working Group.[5] Manual Penetration Testing (or pen testing) has become more of a commodity service with common attributes that do not differentiate the service from one provider to another. A few vendors with less experience in software security have misinterpreted this practice as a network pen test performed annually vs. an application pen test conducted in Q/A.

Several vendors had their own ethical hacking teams internally that conducted pen testing of their products and these teams were always separate from the development teams. Results were often shared with the CTO or head of development to assign responsibility for remediation. The remediated products were then retested before release. Often vendors applied pen testing to an entire release prior to production as a standard process.

---

[4] To read the full example, see Appendix 1

[5] To read the full example, see Appendix 1

Practice Area 5: Configuration Management and Incident Response

This practice includes vulnerability management tailored to how the vendor harvests findings about security identified through one or more of the other practice areas. The vendor prioritizes the remediation effort, assigns the work and tracks progress. Automation of this process is not required for maturity but having a process that is applied consistently is required for level 1-3 maturity. COTS suppliers tend to have more mature processes in place for vulnerability management throughout the lifecycle when compared to the software service providers.

One of the most significant lessons learned in the initial vBSIMM implementations[6] was how differently the contract terms and conditions were interpreted and implemented by various software service providers. For example, each provider that participating in a sample program (Appendix 1) spent many weeks negotiating a contract that included a notification clause with the bank; however the interpretation of how to enforce the clause varied greatly. To understand these differences, the enterprise running this sample program asked participating vendors many questions about interpreting the requirement to notify customers within a reasonable time frame of any security incident.

When asked about their definition of what a "reasonable time period" was for disclosure to a customer of a data breach, the most common response was that a customer would only be contacted after a third party cyber security forensics vendor was hired to conduct an assessment and determined that the customer's data was actually impacted.

When asked about their definition of what a "reasonable time period" was for disclosure to a customer of a data breach, the most common response was that a customer would only be contacted after a third party cyber security forensics vendor was hired to conduct an assessment and determined that the customer's data was actually impacted. The implication is that for a vulnerability discovered in production of a hosted application, the customer was not likely to receive notification. In addition, in a significant incident, the customer would only be notified weeks after the event when a forensics vendor confirmed that the breach impacted data related to the customer. In other words, the customer would not have visibility into the risk and therefore would not be able to manage that risk.

[6] To read the full example, see Appendix 1

vBSIMM Results

The results of vBSIMM are delivered as three documents:

• A spreadsheet that is used to capture the questionnaire responses

• A spreadsheet used to document the assessment results

• Assessment artifacts provided by the vendor.

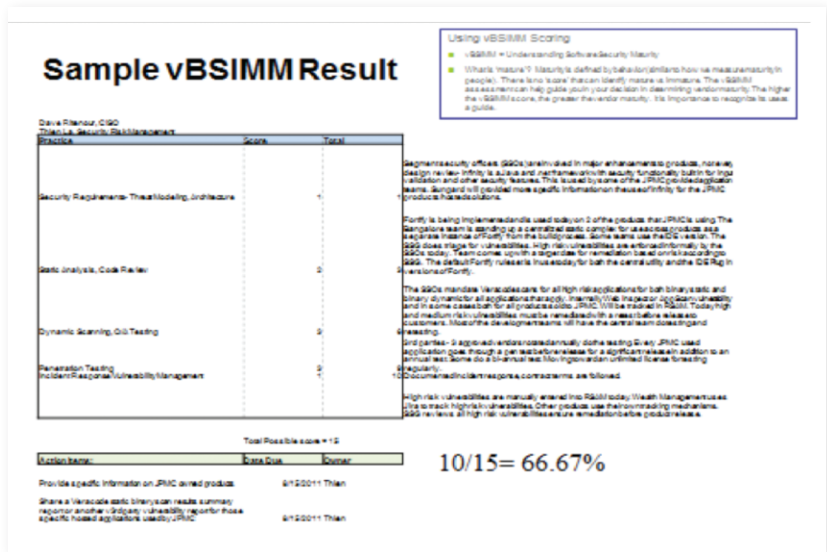**An example of a vBSIMM assessment is provided to the right.**

Figure 2: *Sample vBSIMM Result*

The spreadsheet documenting the assessment results has a column for recording the maturity score of each practice area. Practice area observations are recorded in another column. The action items are also captured in the spreadsheet and they are tracked in the vendor management system for follow up in the next review or sooner when specified.

The highest maturity score is 15 out of 15 (5 practice areas x 3). However one or more practice areas may not apply to some types of vendors. To normalize the scores across all vendors, the maturity score is divided by the number of practice areas. The result is a percentage used to measure results across all vendors in different categories.

The Working Group recommends that the completed assessment results spreadsheets[7] be shared across all financial institutions through an information sharing process established through the FS-ISAC portal. This recommendation was made to the FS-ISAC Product & Services Committee and is under consideration. The Product & Services Committee is preparing a preliminary design of the information sharing process. One design goal is to include an explicit approval by vendors to share assessment artifacts with all FS-ISAC members. Both the Product & Services Committee and the Working Group recognize that the ability to share one vBSIMM assessment with multiple financial institution clients is advantageous for the vendor since the vendor does not have to complete multiple vBSIMM assessments for multiple financial institutions.

---

[7] To see a sample assessment spreadsheet, see Appendix 2

The Working Group also recommends that an education and certification program be administered by the FS-ISAC to certify vBSIMM assessors for consistency and quality of the assessment process. The FS-ISAC is considering accepting bids from third party vendors to conduct the education and certification process for financial institution members. These vendors must have previous knowledge and experience conducting vBSIMM assessments.

The education would be offered several times a year and would be applicable for vendor governance as well as information security staff from member firms. The Product & Services Committee will review the need for certification with the potential information sharing capability and make a final recommendation to the FS-ISAC Board for approval of this control type and related support requirements.

The vBSIMM approach satisfies the need for assessing the process maturity of third party software vendors and services providers. The challenges with this control are ensuring consistent quality of assessment results and training vendor governance professionals to understand software development practices well enough to complete assessments. Each financial institution will manage its own unique vendor assessment practices, and must determine the appropriate integration of vBSIMM assessments with its existing vendor governance practices.

Control Type 2: Binary Static Analysis

The second control type addresses the need for understanding the vulnerability density of every version of every product being supplied by third party vendors. Static analysis tools used for software security testing are common in financial institutions, with several products available for in-house software development. The difference between those products implemented in the development processes and binary static analysis is that the latter does not require access to source code. Sharing access to third party source code is problematic for software suppliers as they need to ensure proper protection of their intellectual property. Binary static scanning provides a method for determining security vulnerabilities without the need to access to source code, a significant benefit for vendors.

The Working Group considered two leading providers of binary static scanning services—Veracode and HP Fortify on Demand. The majority of Working Group members are using one or both services today. One vendor is not recommended over the other. However, a summary of the differences based on Working Group member experience is provided at the end of this section.

Similar to the detective control of using static analysis in the development process, binary static analysis uses a set of tests to identify software vulnerabilities, optimizes the results to reduce false/positives, and uses a set of rules to represent the policy that is enforced.

Security Policy Selection

Choosing the right rule set or policy is the responsibility of the financial institution applying this control. For example, an enterprise may want to consider specific vulnerabilities that must be detected (i.e.: OWASP top 10) or a specific regulatory requirements to meet (i.e.: PCI DSS) or may wish to identify a specific vulnerability that is common in their vendors (custom rule) that they wish to apply. The ability to modify and customize a rule set or policy is available in both vendor solutions although it is a bit easier to implement using Veracode. The Working Group recommends using a combination of OWASP Top 10 and PCI compliance as a baseline policy. Additional rules can be added on top of this baseline as needed.

Artifacts

The Working Group recommends that a number of artifacts be prepared in advance of engaging in static binary analysis to help the third party software vendors understand and comply with the vendor governance requirement for binary static scanning including:

1. Letter from the head of the vendor governance function (consider asking the CIO to sign the letter as well).

2. An overview of the binary static scanning process.

3. A defined set of responsibilities for the third party software supplier, the security analysis provider (the Working Group reviewed Veracode and HP Fortify On Demand) and the enterprise throughout the process.

4. An artifact describing the risk classification definition and the recommended remediation time based on the severity of the vulnerability.

5. A sample artifact that the third party software supplier receives from the security analysis provider and a sample summary of what the enterprise receives from the security analysis provider.

6. An acknowledgement of several items:

   • That this level of product information (i.e. software vulnerabilities) was not shared in the past.

   • A recognition that the vendor's remediation prioritization must balance the level of risk posed by discovered vulnerabilities and the resources required to fix defects and create new functionality.

   • The enterprise and the vendor need to come to consensus on the right level of urgency for remediation priorities.

Exposing security vulnerabilities for a specific version of software at a specific point in time provides an opportunity for a meaningful dialog with the third party software provider regarding remediation priorities. The security analysis provider managing the static binary analysis will deliver detailed information about defects and vulnerabilities to the third party software supplier. Together, the security analysis provider and third party software supplier will determine if there are any additional mitigating controls that may change the risk profile of the vulnerability.

The third party software provider chooses when to release the summary results to the enterprise within a few weeks. Some third parties choose to remediate vulnerabilities and submit a new release of software after the original scan. The summary results from the latest scan are then shared with the enterprise.

The vendors of binary static analysis make it easy to score the vulnerability results either using a letter grade score based on the selected policy or with score based on five categories of risk. The most important aspect of this process is to discuss the findings with the third party software supplier in order to agree on remediation priorities. This discussion often provides insight into how the third party software vendor deals with software security risk and remediation prioritization.
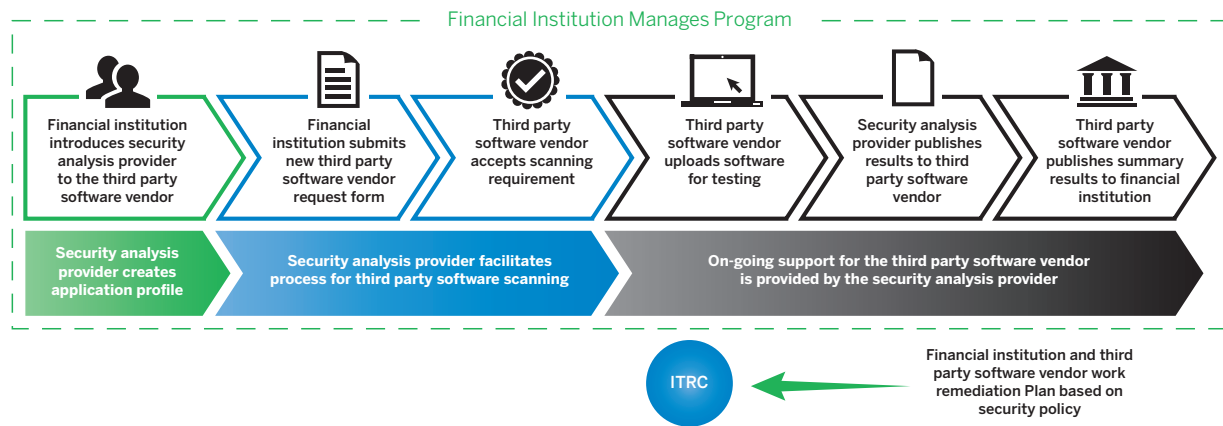

Figure 3: Process for collecting static binary analysis artifact

Veracode vs. Fortify On Demand

The two security analysis vendors that provide binary static scanning services are similar in their approach and both capabilities will identify security vulnerabilities in the software that is scanned. The Working Group observed the following strengths for each security analysis provider based on their track record providing the services over time to the respective member firms:

Veracode	Fortify On Demand
Historical strength in static binary analysis	Level of support for systems integrators
Mature cloud-based service offering	Access to broad portfolio of application security testing products and services under HP's umbrella
Program management service dedicated to building and executing a vendor application security testing program	Scalability to handle large third parties
Approach to vendor application security testing which shifts the cost of analysis to the third party software provider	

Choosing one vendor over the other is up to each financial institution that applies this control. What is important to the financial institution should be the ability to encourage the firm's third party software vendors to accept responsibility for applying effective controls in the development process that improve resiliency and security of the products and/or systems. Veracode offers a unique approach to this with their Vendor Application Security Testing (VAST) program. Veracode's VAST program manages the process of collecting binary static analysis artifacts, while working with software vendors to embed software security in the development process. Additionally, the VAST program incorporates a shift of responsibility and cost burden onto the third party software vendors over time while also increasing the amount of software in scope for this control type for the financial intuition.

Migration of responsibility for security assessment from the financial institution to the third party software vendor is clearly moving in the right direction for the industry. Enabling third party software vendors to share scan artifacts with many financial institutions makes adoption more compelling for the third party software provider.

The Working Group was interested in the ability to share scan result artifacts across the financial institution community. This implies the hosting of a site to store the scanning artifacts and provide the third party vendors with a mechanism for releasing results to specific financial institutions. Neither Veracode nor Fortify on Demand has the mechanics to support this capability today but could develop this capability in the future. This would provide a significant advantage for third party software providers since they deliver the assessment results to any financial institution at any time based on their needs without having to scan code for each respective financial institution. Also a single remediation prioritization effort and roadmap from the third party software vendor could satisfy the needs of the entire community.

Control Type 3A: Policy Management and Enforcement for Consumption of Open Source Libraries and Components

Control Type 3 does not apply to third party software development or COTS vendors. It is included as a control because it represents how the supply chain is feeding internal software development processes within financial institutions today. The majority of internal software created by financial services involves acquiring open source components and libraries to augment custom developed software. The Central Repository (formerly Maven Repository) is one of the largest open source code repositories. Sonatype's analysis of this repository estimates that about 90% of all software development requires the downloading of components.[8] Open source code is available freely and reviewed by many independent developers, but this does not translate into software components and libraries free from security vulnerabilities.

---

[8] From Sonatype Press Release dated Sept 9 2013, "The composition of today's applications is often as high as 90% open source components and 10% custom source code (Based on an analysis of the Central Repository and 1000+ Repository and Application Healthcheck Risk Assessments)"

Aspect Security, a software security consulting vendor, estimates that about 26% of the most common open source components have high risk vulnerabilities in them.[9] The more these open source components are shared, the more widespread the vulnerabilities become. Therefore, it is essential to have a control to protect the flow of open source components into the development process.
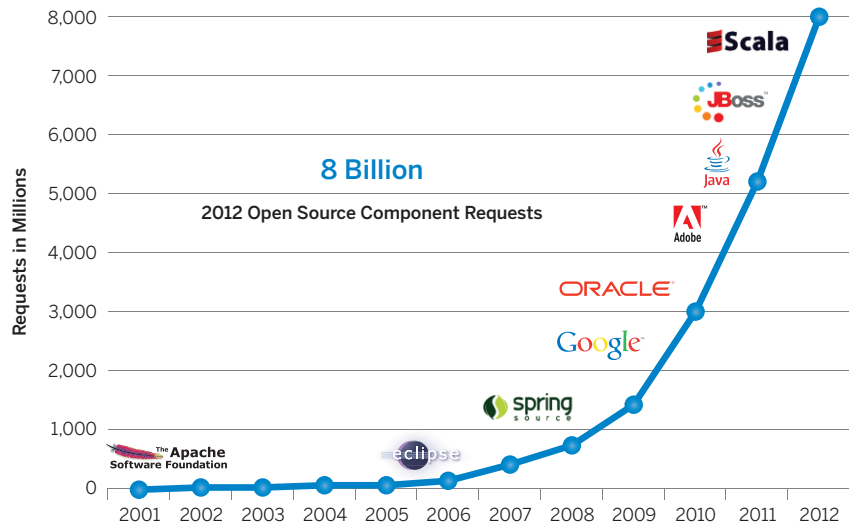

Figure 4: *The growth of open source component usage*

When application developers seek to build new functionality to meet business needs, they turn to open source libraries for access to components that dramatically improve the time to market of their delivery. The most appropriate type of control for addressing the security vulnerabilities in open source, including older versions of the open source, is one that addresses vulnerabilities before the code is deployed—i.e. by applying policy controls in the acquisition and use of open source libraries by developers. Therefore a combination of using controlled internal repositories to provision open source components and blocking the ability to download components directly from the internet is necessary for managing risk. In fact, Gartner recommends that "if open source is used, ensure that the frameworks and libraries used are legitimate and up-to-date, and that the compiler used hasn't been compromised."[10]

There are several technology solutions that address part or most of the needed features to apply lifecycle management controls for open source components. The Working Group has experience with three solutions that offer partial functionality. Two of these vendor solutions have been available on the market for more than 5 years (Palamida and Black Duck) and provide for legal liability as well as security of open source libraries once acquired.

---

[9] Aspect Software "The Unfortunate Reality of Insecure Libraries" March 2012

[10] Gartner, "Maverick*Research: Living in a World Without Trust: When IT's Supply Chain Integrity and Online Infrastructure Get Pwned" October 2012

They both have an ability to tag code components and libraries used within an application portfolio. Thus when new vulnerabilities are discovered, the financial institution can more easily identify the impact of remediation by understanding where all of the components exist within the application portfolio. This also applies to determining any legal liability for the use of open source libraries.

A new approach in the market is Component Lifecycle Management (CLM) which offers the ability to enforce policies in the development process. For example, if a development team inadvertently downloads obsolete software versions, CLM can apply a method of breaking the build when that library is submitted, enforcing the use of a more current version. CLM informs the developers and security staff which components have risky vulnerabilities and which ones do not. The benefits of this approach include:

- Enabling application architects to control versions of software.
- Accelerating the development process by encouraging the consumption of open source libraries that are resilient.
- Reduce operating costs since the cost of ripping out obsolete components from existing applications is high assuming the older versions can be identified in the first place.

Financial institutions should consider options in this control type to apply policies to the consumption of open source components and to specify methods for creating and managing an inventory of open source libraries in use within the application portfolio. There are manual options and automated options that should be considered to improve the resiliency of the most commonly used open source components. The controls applied to the consumption of open source are less expensive to implement than fixing defects after they are deployed in production throughout the application portfolio for the financial institution. An analogy that may apply is the delivery of pure water through our water systems, regardless of geography, is easier to implement when purification is applied at the reservoir rather than the downstream canals, pipes and distribution method.

The screenshot to the right comes from Sonatype CLM and provides a clear indicator of risk and impact for an application portfolio.

The supply chain for software offers a great deal of choice and options for improving time to market for software development today. Financial institutions should consider their options for influencing the consumption of open source libraries and provide scanning capability to identify the most resilient choices.
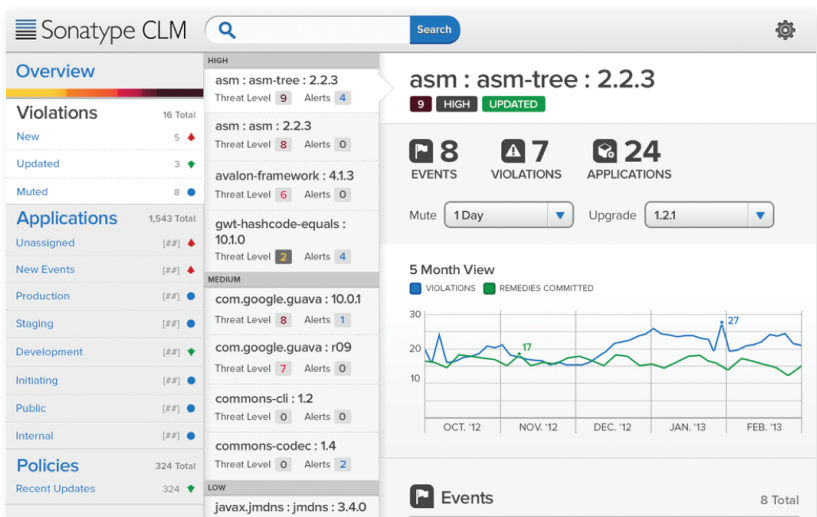

Figure 5: *Dashboard from a CLM product offered by Sonatype*

Firms should also encourage use of mature versions of software that are patched and not yet obsolete by applying policies and enforcing them using the best methods available. The large consumption rate of open source libraries for web and mobile applications offers compelling evidence of how time to market has been realized.

It is time to apply resiliency controls to the consumption process that will reduce the requirements to fix old versions with vulnerabilities after they have been deployed. Controls should encourage deployment of current versions that have been determined to be resilient. Providing more information to architects and developers is the responsibility of the information security staff. The information should improve the understanding that policy management applied early in the lifecycle will both cost less effort and speed up time to market in the long run.

Control Type 3B: A Bill of Materials (BOM) for Commercial Software to Identify Open Source Libraries Used

The Working Group recommends the following naming convention for OSS Bill of Materials to facilitate the controls for third party applications delivered to financial services institutions.

There are common use cases where an institution may mix and match internally developed solutions with vendor third party solutions. The idea is to be able to assess risk of open source software components regardless of the source. The Bill of Materials Naming Standards may be used by internal development teams and third party software providers to provide a common reference point for consumption of open source libraries and components.

This naming convention will leverage and extend the work done by Mitre and NIST on the CPE (Common Platform Enumeration). The four general use cases identified for using CPE include:

- Software Inventory
- Network-Based Discovery
- Forensic Analysis/System Architecture
- IT Management

The primary focus of CPE is software inventory. The other use cases are alternate cases where if CPE provides value, there is added benefit.

Use of CPE also aligns with SCAP (Security Content Automation Protocol), which is a synthesis of interop specifications. The goal is to extend from CPE and SCAP and introduce a minimal number of new concepts and elements. The focus on Bill of Materials for FS-ISAC is to support Control Type 3 of the Appropriate Software Security Control Types for Third Party Services and Product Providers.

Control Type	Description
Policy management and enforcement for consumption of open source libraries and components	Level of support for systems integrators

The primary use case for Bill of Materials data is:

- List known security vulnerabilities: Reconcile against known sources that are public (specifically, the National Vulnerability Database), commercial, or internal vulnerability sources.

Secondary use cases include:

- Bill of Materials data could be centrally aggregated without each financial institution having to perform transformations to a lowest common denominator.
- Risk scoring: the ability to rank and tier software based on the contents of the Bill of Materials
- Prioritization of vulnerability management
- Infrastructure & lifecycle management planning
- Awareness of potential intellectual property issues
- Vendor evaluation criteria: If a vendor does not provide OSS Bill of Material information or allow scanning of its code through contract negotiations, the vendor rating may be impacted.

The Working Group does not endorse any of the vendors or products associated with a control type. The Working Group may discuss or work with vendors in the domain for feedback. Vendors currently offering products for OSS Management include:

- Black Duck
- Open Logic
- Palamida
- Sonatype
- Veracode

This standard format could also be utilized to support manually entered Bill of Materials.

Structure

The CPE Item type consists of a Wfn (Well-formed CPE Name) for the component. Wfn consists of part (type), vendor, product name, Version, Update, edition, and language.

The CPE Name may be an official entry that is part of the National Institute for Standards and Technology (NIST) dictionary or a provisional entry. A provisional entry may be used for internal and released software, but once a publicly declared vulnerability is found, the expectation is that the owners would follow the CPE Submission Process to make the CPE official under the U.S. National Vulnerability Database (NVD) program. The scan generator program may attempt to determine the Wfn fields (such as vendor, product name, version, etc.) from the file name, package name, metadata files or other aspects detected by the scan. The CPE Submission Process consists of sending a well-formed, CPE schema-compliant XML, with well-chosen Vendor, Product, and Version naming by email to cpe_dictionary@nist.gov. The CPE Dictionary is hosted and maintained by the NIST as part of the NVD program. NIST is responsible for ensuring that the CPE Dictionary conforms to the CPE Specifications, and for managing the content review and quality assurance processes.

Checksums are optional, but highly recommended to confirm file integrity. The SPDX (Software Package Data eXchange) specifies attributes for both the checksum algorithm and the checksum Value. The checksum or hash sum is a block of data for the purpose of detecting a match and to verify data integrity. By providing a unique identifier of the package, confusion over which version or modification of a specific package the SPDX file references should be eliminated.
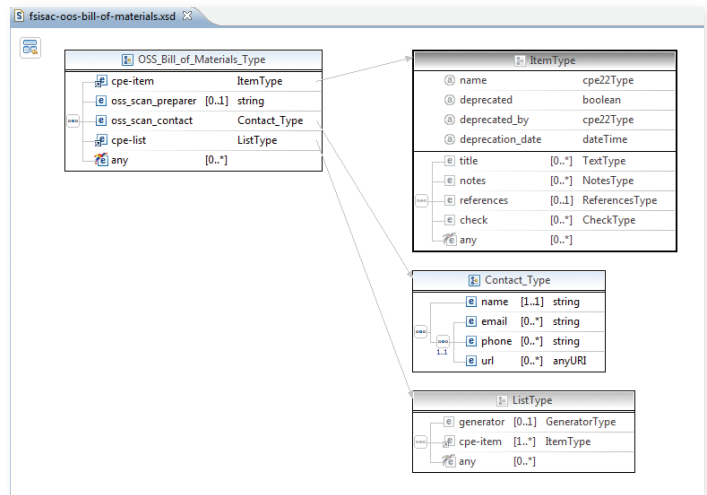

Figure 6: *Caption*

The following are RDF and tag examples:

```
<Checksum>
        <algorithm rdf:resource="checksumAlgorithm_sha1"/>
        <checksumValue>d6a770ba38583ed4bb4525bd96e50461655d2758
        </checksumValue>
</Checksum>
```

or

FileChecksum: SHA1: d6a770ba38583ed4bb4525bd96e50461655d2758

FileChecksum: MD5: 624c1abb3664f4b35547e7c73864ad24

The RDF syntax may be used in the ItemType any attribute. The tag syntax may be used in the ItemType notes attribute.

The Working Group has also noted that there may be some extensions that are more complex than can be supported by the URI provided by the cpe_dict:reference. In these scenarios, the notes attribute should be used. For example, with a Python package and version such as the arguments of a pip install command, the BOM note could include software that isn't in the CPE Dictionary, but is uniquely identifiable within the Python community on PyPI (Python package index). This information can be communicated along with the "Provisional CPE." This accounts for the fact that there may be other authoritative systems of record beyond the NVD that may be used. The note attribute will be utilized for external identifiers.

The OSS Bill of Materials XML file may be distributed embedded within the application archive or stand-alone. The expected exchange of the information is that the third party software provider will provide the OSS Bill of Materials to the entity where the software is delivered. The goal is that vendors could make this data file available for delivered software, and the results could be readily analyzed by the company acquiring the software.

The format would allow third party software providers to self-declare their ingredient list or rely on an external service to prepare the scan. For field-level documentation, refer to the schema documentation tags for more details.

Vulnerability Lookup

The OSS Bill of Materials does not contain the vulnerability information. This is for several reasons. First, the vulnerability lookup is a point-in-time event. There may have been no known vulnerabilities when the product shipped. Second, since the component is likely part of a larger system, the context of a lookup may change if a financial institution has more information available since the vulnerability may be associated with a single cpe artifact or a system of artifacts. Additionally, vulnerabilities may be associated with a dependency or composition of multiple elements.

Vendor Note

The OSS Bill of Material Naming Standard is intended to provide both an output format (in lieu of a report) and an input format to OSS Management Tools. The approach of using XML is to support both human-readable and machine-readable use and provide a mechanism to programmatically validate the structure.

Other Related Standards

The OSS Bill of Material Naming Standard may evolve over time. Other standards that were reviewed during the process include:

- ISO/IEC 19770-2:2009 provides a specification for tagging software to optimize its identification (www.iso.org/iso/catalogue_detail.htm?csnumber=53670).

- The Software Package Data Exchange® (SPDX®) operates under the auspices of the Linux Foundation and is a specification for a standard format to communicate the components, licenses and copyrights associated with a software package (spdx.org).

- Package External Identifier Proposal (docs.google.com/document/d/1j6LWnkh5GbMV9Xo5_zJ0wTNLROEIa4o1OU279Yuel90/edit)

- External Security and Asset Management Identifier Proposal (docs.google.com/document/d/1WfArS8_xR_CQ_5plOOMtj1y9ps5M-gXFjofUBXR8hyE/edit#heading=h.idwvi8enwar2)

- Open Vulnerability and Assessment Language (OVAL®) is an international, information-security community standard to promote open and publicly available security content, and to standardize the transfer of this information across the entire spectrum of security tools and services (oval.mitre.org/).

References

- NIST's Security Content Automation Protocol Page on CPE: scap.nist.gov/specifications/cpe/

- Common Platform Enumeration: Naming Specification Version 2.3: csrc.nist.gov/publications/nistir/ir7695/NISTIR-7695-CPE-Naming.pdf

- Common Platform Enumeration: Name Matching Specification Version 2.3: csrc.nist.gov/publications/nistir/ir7696/NISTIR-7696-CPE-Matching.pdf

- Common Platform Enumeration: Dictionary Specification Version 2.3: csrc.nist.gov/publications/nistir/ir7697/NISTIR-7697-CPE-Dictionary.pdf

- Common Platform Enumeration: Applicability Language Specification Version 2.3: csrc.nist.gov/publications/nistir/ir7698/NISTIR-7698-CPE-Language.pdf

- Official Common Platform Enumeration (CPE) Dictionary: nvd.nist.gov/cpe.cfm

- Mitre's CPE Specifications — Archive: cpe.mitre.org/specification/

Control Type 4: Contract Language

Based on the recommended controls in this whitepaper, acquiring firms should choose a combination of software security controls that work best for their business environment. No matter which controls an acquirer decides to implement with their third party vendors, acquiring firms should also work with their legal teams to incorporate vendor responsibility for software security into every applicable vendor contract. Formal service level agreements (SLAs) will ensure that the vendor will not jeopardize the acquiring firm's compliance or security posture. Contract language can also be written to specify penalties in the event that a vendor is found to be out of compliance with the required SLAs.

At a minimum, every new or renewed contract should contain a set of provisions requiring the vendor to deliver a product or service which is compatible with the acquiring firm's software security policy. Additionally, contract language may require software vendors to submit independently assessed evidence that their software security practices meet the level specified by the acquiring firm. These might take the form of reports from application security testing services, such as binary static analysis as described in Control 2.

Like choosing specific software security controls to implement as part of a broader vendor governance program, choosing the right contract language to include will be unique to every acquiring firm's security objectives and business environment. It is important for acquiring firms to work with their legal teams to write language that is appropriate to their specific situation. Ideally, the software security group works with their legal department to create standard boilerplate language for use in every contract with vendors and outsourced providers. A healthy relationship with a vendor cannot be guaranteed through contract language alone: it is important for the software security group to engage with the software vendor, discuss the vendor's security practice, and explain in concrete terms (rather than legalese) what the acquiring firm expects of the vendor.

Sample Contract Terms and Conditions

1. "Vendor Application Security Testing Code Scan" means a vulnerability scan of Supplier's software binaries (not source code) conducted by Veracode, Inc. on a specific application version for the purpose of identifying security vulnerabilities. Results are prioritized based on risk so that the Supplier may remediate deficiencies accordingly.

2. "Vendor BSIMM" (or "vBSIMM") means an assessment process, as defined and amended from time to time by www.bsimm.com and the Financial Services Information Sharing & Analysis Center 3rd Party Software Security Working Group whitepaper that provides visibility into the maturity of a supplier's ability to deliver secure software by evaluating: (1) architecture analysis activity, (2) code review activity, (3) security testing activity, (4) penetration testing activity, (5) configuration management, (6) incident response and (7) vulnerability management.

   a. Development of Software for Use by Enterprise. To the extent Supplier is engaged by Enterprise to develop, create, or maintain computer software for use by Enterprise, Supplier shall submit to the following security review procedures and/or provide GS the following Artifacts: vBSIMM and VAST.

   b. Development of Mobile Applications for Use by Enterprise or Enterprise Customers. To the extent Supplier is engaged by Enterprise to develop mobile applications for use by Enterprise or Enterprise customers, Supplier shall submit to the following security review procedures and/or provide GS the following Artifacts: vBSIMM, VAST, Encryption Review Questionnaire and Authentication Review.

c. Provision of Commercial Off-The-Shelf (COTS) Software to Enterprise. To the extent Supplier is engaged by Enterprise to provide COTS Software to Enterprise, Supplier shall submit to the following security review procedures and/or provide GS the following Artifacts: VAST.

3. Supplier will fully cooperate with ENTERPRISE in connection with efforts to assess and remediate vulnerabilities that could compromise the data, systems, or critical functioning of the information technology infrastructure of ENTERPRISE or its clients or customers (each, a "Critical Vulnerability"). To that end, Supplier will (a) actively monitor industry resources (e.g. www.cert.org, pertinent software vendor mailing lists and websites and information from subscriptions to automated notification services) for applicable security alerts and immediately notify ENTERPRISE upon the discovery of a Critical Vulnerability in its external-facing, internal, subcontractor or partner environments or in the products or services Supplier provides to ENTERPRISE; (b) respond in writing within 48 hours to an inquiry made by ENTERPRISE about the impact of a known Critical Vulnerability on Supplier's external-facing, internal, subcontractor or partner environments or on the products or services Supplier provides to ENTERPRISE; (c) within 72 hours of either (i) Supplier's discovery of a Critical Vulnerability or (ii) receipt of a ENTERPRISE inquiry about a Critical Vulnerability that impacts Supplier's external-facing, internal or partner environments or the products or services Supplier provides to ENTERPRISE, provide ENTERPRISE with a written and detailed plan to remediate such Critical Vulnerability appropriately and urgently; and (d) provide ENTERPRISE with written confirmation as soon as each such Critical Vulnerability has been remediated. Without notice or other advice to Supplier and in addition to the above Supplier obligations, ENTERPRISE is entitled to conduct Critical Vulnerability scans of Supplier's external-facing, internal, subcontractor or partner environments.

4. To the extent that Supplier is providing software, including software development services for ENTERPRISE, Supplier shall demonstrate the maturity of controls in its development process. In conjunction with delivery of the software, Supplier agrees to complete a vBSIMM assessment and provide to ENTERPRISE applicable documentation and/or artifacts which substantiate that the following software development controls are in place for the scope of the Deliverables being provided to ENTERPRISE hereunder: (i) security requirements documented during the requirements phase of the software development life cycle; (ii) secure architecture design; (iii) static code analysis during development (secure code review of the entire code base based on, at a minimum, the Open Web Application Security Project (OWASP) Top 10 and SysAdmin, Audit, Networking, and Security Institute (SANS) Top 25 software security risks or comparable replacement); (iv) dynamic scanning of web-facing applications and penetration testing of internal applications, using industry standard testing methodologies during the build process or quality assurance phase; (vi) open source code used in TPP-provided applications must be appropriately licensed, inventoried and evaluated for security defects; and (vii) security vulnerability management. If Supplier is unable to substantiate that the software is free of material security defects (i.e., no critical or high risk defects) through the above assessment, Supplier will conduct a software vulnerability scan (using an industry standard tool, e.g., Veracode) or submit to application scanning from a ENTERPRISE-approved vendor, and (x) share the results of that scan with ENTERPRISE; (y) to the extent that scan identifies any critical or high risk vulnerabilities, Supplier will remediate those vulnerabilities before implementation of the software into production (whether the software is hosted by ENTERPRISE, Supplier or a third party on behalf of either); and (z) develop remediation plan(s) to address any other vulnerabilities to ENTERPRISE's reasonable satisfaction (such plan to be included in the applicable Schedule as an obligation of Supplier) with the remediation occurring as soon as reasonably practicable, not to exceed six months of the effective date of the applicable Schedule.

In addition, there are terms and conditions offered on the SANS website that may be useful. software-security.sans.org/appseccontract

Conclusion

Financial institutions must determine their own path for addressing third party software security. While implementing all three recommended controls or even just one will significantly improve the resiliency of the application portfolio, these controls must be incorporated within existing vendor governance programs in order to achieve the maximum level of efficacy. When executed correctly, these recommended approaches will increase the effectiveness of the risk management practices and enable avoidance of expensive remediation post-production.

As member firms better understand the risks associated with sharing critical data and systems with third parties, the FS-ISAC Product & Services Committee will continue to refine third party software security control types either through the Third Party Software Security Working Group or another effort.

Appendix 1: Control Type 1 Through a vBSIMM Case Study

The Working Group looked closely at an example from a leading multinational bank that implemented the vBSIMM process for vendors. The bank performed vBSIMM assessments for over 18 months and had completed close to 50 vBSIMM artifacts. The firm evolved its implementation approach by incorporating vBSIMM questions into the vendor questionnaire that was already in use and educating its vendors on how to provide information in the five distinct practice areas within the vBSIMM. The results from the questionnaires were scored based on the answers to make it easier for vendor management staff to administer.

The Working Group noted the need to engage information security professionals with experience conducting software security assessments in addition to the vendor management staff, who could interpret the results from both the questionnaire responses and the artifacts shared by the vendor during the assessment process. The bank published a set of principles to apply to the assessment process. Financial institutions that have successfully implemented software security programs agree that integrating effective controls into the development process is an iterative process; onboarding a software vendor into the collaborative process initially outweighed a rigorous compliance check.

This bank implemented the vBSIMM for the five largest providers of off-shore development services as part of an initial project. The five firms were asked to volunteer to participate in vBSIMM assessments. Four of the firms agreed to allow the bank to conduct a vBSIMM assessment. The fifth one decided that the services provided by their firm to the bank did not qualify for a vBSIMM assessment since they were not responsible for determining the software development process and instead followed the bank's software development lifecycle (SDLC). The four vendors acknowledged that they were aware of software security practices but the practices were not included in their current projects. Their perception was that the bank would view the practices as additional work and would be unwilling to pay for the increase in labor to implement the security controls. The vendors agreed to review each of the five practices and consider methods for implementing controls within each practice area within a few months and then ask the bank to begin the vBSIMM assessment. The bank approved of this approach and conducted the vBSIMM assessments by introducing candidate activities for each practice area with each vendor so they understood the activities available based on the BSIMM framework. The vendors then worked on implementation of the activities they selected.

The bank conducted the assessment sessions reviewing information prepared by each vendor that included a description of the activities and selected artifacts from the activities. The review session took an average of 60 minutes to complete and the bank provided the risk score (maturity level of activities in each practice area) to the vendor by practice area. The vendors provided the artifacts prior to the session so the assessor was able to review the artifacts.

The four vendors acknowledged that they were aware of software security practices but the practices were not included in their current projects. Their perception was that the bank would view the practices as additional work and would be unwilling to pay for the increase in labor to implement the security controls.

The sessions themselves were often conducted as presentations of the activities in each respective practice area with the assessor asking questions of the vendor. The vendor selected product development or development leaders and architects to participate in the vBSIMM assessment. In a few cases the vendor assigned an information security officer to the project to oversee the implementation of the controls and/or review the assessment results. Each vendor scored at least the minimum score of level 1 for each practice area that applied. A few scored level 2 and level 3 for select practice areas. All of the vendors identified the appropriate controls by practice area and implemented the controls on current development projects for the bank by the time the initial vBSIMM project was completed. The vendors found they could implement the controls without changing their billing or rates of the current projects and they made a commitment to the bank to incorporate these practices in all future assignments for the bank.

The bank included additional types of vendors to the initial project work to evaluate the effectiveness of the vBSIMM and use the lessons learned to determine if modifications in the approach or techniques was necessary. The bank selected vendors from two more categories of vendors, commercial off the shelf software (COTS) vendors and providers that host Software as a Service (SaaS) offerings. They requested volunteers from both COTS and SaaS vendors and included approximately ten vendors from each category.

The results from the COTS category of vendors were quite diverse. This category included vendors that were familiar with and had adopted the BSIMM model so their knowledge of software security was significantly more mature. However, several larger software vendors chose not to provide assessment artifacts based on advice from their legal departments. In addition, several smaller COTS vendors were introduced to the BSIMM model through this initial project and the assessment results indicated very low maturity.

The SaaS providers also demonstrated diverse results with several measuring at very high maturity in the vBSIMM assessment while others had little or no security controls in their development process and limited understanding of the activities in the each of the respective practices. Several of these providers found that participation in this process provided additional value as they learned a great deal about techniques, tools and practices related to developing secure software.

The bank's information security and vendor management leaders reviewed the results of the initial projects very carefully. Each category of vendors offered interesting learning opportunities for considering full rollout of the vBSIMM process. The bank recognized that additional training for the assessors was essential and it was clear vendor governance staff would likely struggle with the responsibility of conducting the assessment. In all of the initial projects a highly experienced software security professional led the vBSIMM process for the vendors with vendor governance staff observing. One of the lessons that came out of the initial projects was the need to engage the right resources from the vendors. Product development or application development leaders are essential in making the vBSIMM assessment process work effectively. The vendor account manager can play an important role in facilitating the introductions to the right technical resources but are not able to provide useful information to propel the assessment process forward.

> Product development or application development leaders are essential in making the vBSIMM assessment process work effectively.

The next most significant learning was the need to gather more information from the vendor prior to the assessment session and to make it easier for vendor governance teams to both collect and interpret the information provided by the vendors. The bank decided to add specific vBSIMM questions by practice area into the vendor questionnaire[11] (often referred to as a Standard Information Gathering tool or SIG) and to score the results from the responses making it easier for vendor governance professionals to understand how to assess maturity. The bank anticipated having to include information security professionals that understood software security in the process but wanted to rely on the vendor governance staff for most of the support work required.

The bank has implemented the vBSIMM process for selected vendors based on the types of services they offer and the application risk. For example they apply the vBSIMM for all hosting vendors, for selected COTS products based on an application risk classification and for service providers that manage the development process following their own SDLC. The bank provides specific guidance to vendor governance professionals on how to determine the most appropriate way to apply the vBSIMM when a vendor has different SDLCs by product or where there are different development teams.

---

[11] A sample of the questionnaire used by the bank is available in the Appendix 2. For an Excel version of this questionnaire, please contact a Working Group member.
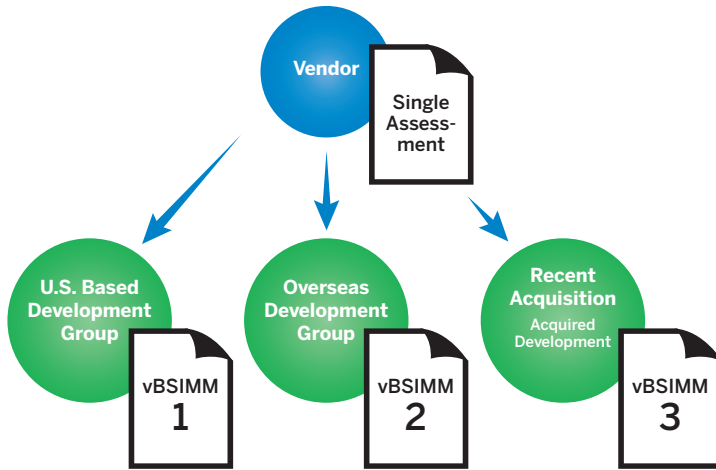
Figure 6: *Diagram of approach using vBSIMM assessment for a software supplier with multiple products, development regions, and M&A activity.*

Some COTS vendors will acquire other products and over time will migrate the development processes for the acquired company to the acquirer's core development process. Often the original product development process is in place for a year or longer post acquisition creating a need for vBSIMM assessment for each product unless the product development process is identical (same phases, techniques, controls, etc.). Also a services vendor may use different development centers around the world taking advantage of the market for technical talent (the same way a bank would) and therefore follow different development practices. In this case a vBSIMM would be done for each geographic development center since the development practices vary by center.

The lessons learned from the bank's initial projects were significant in influencing adjustments to the implementation approach going forward. This included an acknowledgement that a gulf existed between software vendors' interpretation their responsibilities and the bank's. The bank has continued the iterative process of refining the vBSIMM assessment process within their organization and as it applies to their software suppliers.

Appendix 2: Sample vBSIMM Questionnaire

TPRM Summary

Application Development Overview (vBSIMM)	
Name of the application	
What does the application do for our firm?	
Application inventory ID	
Risk classification (H, M, L)	
Type of application (Web, Mobile, Client/Server, etc.)	
Application development language(s) (Java, .NET, iOS, etc.)	

Summary	Min Score	Maturity Score
Architecture		
Development		
QA/UAT		
Penetration Testing		
Production		

	Vendor Response	Results Interpretation	Initial Score	Highest Possible Score	Assessor Comments
Architecture					
How do you identify the most critical applications/products for identifying risk?		A formalized process is more consistent than an arbitrary approach. Validate the approach to ensure that high risk apps are identified using sound methodology (are there high risk apps not being identified?)			
Do you perform a security feature review (authentication, access controls, use of cryptography, etc.)?		Security evaluations for every major release demonstrates a high level of maturity. Combined with additional security monitoring may be effective at mitigating risk.			

	Vendor Response	Results Interpretation	Initial Score	Highest Possible Score	Assessor Comments
Architecture					
Do perform a secure architecture design review for high risk applications?		Security evaluations for every major release demonstrates a high level of maturity. Combined with additional security monitoring may be effective at mitigating risk.			
Do you incorporate threat modeling into the business requirements/design process of your SDLC?		Failure to assess an application from a security perspective should not be an acceptable approach.			
		Less than 6 is poor; higher than 10 is an indication of maturity.			
Development					
Do you have a list of the most common vulnerabilities/bugs that need to be eliminated?		An affirmative response is indicative of some level of maturity. Ask questions to understand what they do and assign a level of maturity.			
Do you perform secure code reviews against the entire code base in the development phase?		Security evaluations for every major release demonstrates a high level of maturity. Combined with additional security monitoring may be effective at mitigating risk.			
Is there a security expert who performs the review? (Describe who conducts the code review.)		An affirmative response is indicative of some level of maturity. Ask questions to understand what they do and assign a level of maturity.			
Do you use automated code review tools?		An affirmative response is indicative of some level of maturity. Ask questions to understand what they do and assign a level of maturity.			
Do you remediate the findings?		This demonstrates a risk based approach by focusing on the highest risk issues. It assumes that the tools are robust enough to identify the critical security defects. Verify and ask questions related to updates for the process of identifying security issues.			
Do all developers receive formal software security training?		Formal secure development training is an indication of some level of maturity. Ask for details about the program; i.e. what coursework is required vs. optional and the frequency for this training.			
Do you have security experts that work with developers for every application?		An affirmative response is indicative of some level of maturity. Ask questions to understand what they do and assign a level of maturity.			

	Vendor Response	Results Interpretation	Initial Score	Highest Possible Score	Assessor Comments
Development					
How many applications do you perform secure code review for annually?		Commercially available code review analyzers or a 3rd party evaluation service should be used as part of a comprehensive software security practice. A dedicated software security group should be considered to drive/manage the process. Understand the process for re-evaluation once initially identified issues are remediated. Manual code reviews are not sustainable for a portfolio this size. Low developer counts (less than 100) could indicate outsourced development.			
Do you outsource any development? (Provide the name of the company and geographic location.)		This reduces the risk of external development introducing vulnerabilities.			
How many developers follow the SDLC under this review?		Large development shops should use commercially available code review analyzers or a third party evaluation service as part of a comprehensive software security practice. A dedicated software security group should be considered to drive/manage the process. Understand the process for re-evaluation once initially identified issues are remediated. Manual code reviews are not sustainable for a portfolio this size.			
Are the security defects identified being shared with the developers to prevent reoccurrence?		An affirmative response is indicative of some level of maturity. Ask questions to understand what they do and assign a level of maturity.			
		Less than 6 is poor; higher than 14 is an indication of maturity.			
QA/UAT					
Does your QA function execute edge/boundary value condition testing?		A negative response is indicative of a control gap. Ask questions to understand what level the vendor does do in this space and create an RP if necessary.			
Are testing procedures in place to determine whether security features are effective?		An affirmative response is indicative of some level of maturity. Ask questions to understand what they do and assign a level of maturity.			

	Vendor Response	Results Interpretation	Initial Score	Highest Possible Score	Assessor Comments
QA/UAT					
Do you use dynamic scanning against web apps while in the QA phase?		Security evaluations for every major release demonstrates a high level of maturity. Combined with additional security monitoring may be effective at mitigating risk.			
If no, is there any form of black box testing or are there scripts specific to abuse cases that are used?		A negative response is indicative of a control gap. Ask questions to understand what level the vendor does do in this space and create an RP if necessary.			
Do you remediate security vulnerabilities identified?		This demonstrates a risk based approach by focusing on the highest risk issues. It assumes that the tools are robust enough to identify the critical security defects. Verify and ask questions related to updates for the process of identifying security issues.			
Does your QA process involve fuzz testing (small numbers large numbers, negative values, binary sequences, command line inputs random values, etc.) ?		An affirmative response is indicative of some level of maturity. Ask questions to understand what they do and assign a level of maturity.			
How many releases of the this application occur in a calendar year?		Applications should be evaluated every release. The fewer the releases the less likelihood of new security issues being introduced.			
	Less than 6 is poor; higher than 9 is an indication of maturity.				
Penetration Testing					
How often do you perform pen testing of applications (not perimeter pen testing)?		Security evaluations for every major release demonstrates a high level of maturity. Combined with additional security monitoring may be effective at mitigating risk.			
Who performs the pen tests?		Using an external vendor is acceptable assuming the vendor is reputable. Focus on ensuring the approach the vendor uses (uses commercial tools combined with skilled pen testers). External firms typically keep their testers more current on emerging threats.			

	Vendor Response	Results Interpretation	Initial Score	Highest Possible Score	Assessor Comments
Penetration Testing					
If internal, are the pen testers part of the development group?		A negative response is indicative of a control gap. Ask questions to understand what level the vendor does do in this space and create an RP if necessary.			
Do you use the same approach (tools, methods, time spent, etc.) on each application pen test?		An affirmative response is indicative of some level of maturity. Ask questions to understand what they do and assign a level of maturity.			
Which applications receive penetration testing?		This demonstrates a risk based approach by focusing on the highest risk apps. Ignoring medium and low apps could introduce risk as variations in the risk ranking methodology may exist.			
Are pen testing results managed through a defect or vulnerability management system where results are assigned for remediation?		A negative response is indicative of a control gap. Ask questions to understand what level the vendor does do in this space and create an RP if necessary.			
Do you test the complete production version of the application (not just certain components)?		An affirmative response is indicative of some level of maturity. Ask questions to understand what they do and assign a level of maturity.			
Do you currently have any unremediated Pen Test issues in the application under review?		Verify/ensure that the production version was tested and that no vulnerabilities exist.			
Do you pen test applications while authenticated?		An affirmative response is indicative of some level of maturity. Ask questions to understand what they do and assign a level of maturity.			
Is the pen testing environment production or production like?		An affirmative response is indicative of some level of maturity. Ask questions to understand what they do and assign a level of maturity.			
		Less than 12 is poor; higher than 15 is an indication of maturity.			

	Vendor Response	Results Interpretation	Initial Score	Highest Possible Score	Assessor Comments
Production					
Is vulnerability/security information found in operations or production shared with developers? Describe how.		An affirmative response is indicative of some level of maturity. Ask questions to understand what they do and assign a level of maturity.			
If a security breach occurs, which groups have mandated involvement?		Operations will be focused on correcting the issue from an operational perspective and while helpful from an RTO perspective, it is important for Incident response and AppSec be involved to manage remediation as an incident and prevent the issue from reoccurring.			
Does the incident response process include steps to identify root/cause and prevent reoccurrence? Describe what.		An affirmative response is indicative of some level of maturity. Ask questions to understand what they do and assign a level of maturity.			
If you host applications for the enterprise, is there a service in place to monitor production applications for vulnerabilities?		A negative response is indicative of a control gap. Ask questions to understand what level the vendor does do in this space and create an RP if necessary.			
		Less than 6 is poor; higher than 7 is an indication of maturity.			

Earlier in the lifecycle, preventative controls are most effective. As any application migrates to production, detective controls become more important. Some vendors may rely entirely on Pen Testing as a SDLC control. While this can be effective in the detection of vulnerabilities, it does nothing to prevent issues from being reintroduced (unless shared with the developers who introduced the issue). Understanding who performs the pen test is important (are they qualified?). A higher number of releases amplify the need for detective controls. Mature programs will contain a mixture of preventative and detective controls for every release ensuring that developer education is addressed. Vendor attestation is never enough; always verify artifacts that support vendor responses. Lack of artifacts or practices may require a "point in time assessment" to measure the security posture of a given application. Architecture focuses primarily on preventative controls. Applications from those vendors that score less than the min mature score in development, QA/UAT, and Pen Testing may require RPs and/or point in time assessments (such as Binary, Dynamic or a Pen Test).

A Word About AGILE Development

AGILE development focuses on short "sprints" of development which usually last several weeks. In this method of development, engraining detective controls becomes more difficult (but not impossible). Preventative controls are of greater importance when using this method. Ensure that detective software security controls are in place and are used prior to the application migrating toward development. For example, not all "sprints" are released to production immediately. In this case static analysis can be integrated into the development cycle and dynamic may be used in the last release before production. AGILE requires more flexibility but understand that the use of AGILE is not an excuse not to apply controls.

Architecture

AA1.1	Activity	Response	IRM Comment
Threat modeling allows you to systematically identify and rate the threats that are most likely to affect your system. By identifying and rating threats based on a solid understanding of the architecture and implementation of your application, you can address threats with appropriate countermeasures in a logical order, starting with the threats that present the greatest risk. Threat modeling has a structured approach that is far more cost efficient and effective than applying security features in a haphazard manner without knowing precisely what threats each feature is supposed to address.	Architecture Analysis Activity *Perform security design/architecture/ feature review*		

	Vendor Response	Describe the procedure, process and tools used	If your process is not in the list, describe what you do	Additional Comments
Architecture				
How do you identify the most critical applications/products for identifying risk?				
Do you perform a security feature review (authentication, access controls, use of cryptography, etc.)?				
Do perform a secure architecture design review for high risk applications?				
Do you incorporate threat modeling into the business requirements/design process of your SDLC?				

Development

CR1.4	Activity	Response	IRM Comment
Source code review is one of the critical controls. Security code reviews focus on identifying insecure coding techniques and vulnerabilities that could lead to security issues. The cost and effort of fixing security flaws at development time is far less than fixing them later in the product deployment cycle. The use of an automated tool demonstrates maturity in the practice since the tools are much more mature today and make the review process more consistent. Managing false/positives from a source code tool is necessary for large scale development work and requires expertise and effective practices. For example, using a process or function to interpret vulnerability information or reducing the number of rules in the baseline rule set are both techniques for managing false/positives. Using a manual code review process for a small team may be effective as long as there is an experienced software security professional conducting the review. Manual code review is required for platforms not covered through source code static analysis tools.	Code Review Activity *Use automated tools along with manual review*		

	Vendor Response	Describe the procedure, process and tools used	If your process is not in the list, describe what you do	Additional Comments
Development				
Do you have a list of the most common vulnerabilities/bugs that need to be eliminated?				
Do you perform secure code reviews against the entire code base in the development phase?				
Is there a security expert who performs the review? (Describe who conducts the code review.)				
Do you use automated code review tools?				
Do you remediate the findings?				
Do all developers receive formal software security training?				
Do you have security experts that work with developers for every application?				

	Vendor Response	Describe the procedure, process and tools used	If your process is not in the list, describe what you do	Additional Comments
Development				
How many applications do you perform secure code review for annually?				
Do you outsource any development? (Provide the name of the company and geographic location.)				
How many developers follow the SDLC under this review?				
Are the security defects identified being shared with the developers to prevent reoccurrence?				

QA/UAT

ST1.1		Activity	Response	IRM Comment
The QA team goes beyond functional testing to perform basic adversarial tests. They probe simple edge cases and boundary conditions and no attacker skills required to do this. A minimalistic practice is to conduct specific tests designed to uncover potential input/output vulnerabilities in an application. Test scripts used or output of tests designed to do edge/boundary condition testing may be considered.		Security Testing Activity *Ensure QA supports edge/boundary value condition testing.*		

	Vendor Response	Describe the procedure, process and tools used	If your process is not in the list, describe what you do	Additional Comments
QA/UAT				
Does your QA function execute edge/ boundary value condition testing?				
Are testing procedures in place to determine whether security features are effective?				
If yes, are the procedures derived by obtaining a list of security features implemented by the architecture group?				
Do you use dynamic scanning against web apps while in the QA phase?				

	Vendor Response	Describe the procedure, process and tools used	If your process is not in the list, describe what you do	Additional Comments
QA/UAT				
If no, is there any form of black box testing or are there scripts specific to abuse cases that are used?				
Do you remediate security vulnerabilities identified?				
Does your QA process involve fuzz testing (small numbers large numbers, negative values, binary sequences, command line inputs random values, etc.)?				
How many releases of the application occur in a calendar year?				

Penetration Testing

PT1.1		Activity	Response	IRM Comment
Penetration Testing is a conventional security control and the one most widely used by software vendors.		Penetration Testing Activity *Use penetration testers to find problems.*		

	Vendor Response	Describe the procedure, process and tools used	If your process is not in the list, describe what you do	Additional Comments
Penetration Testing				
How often do you perform pen testing of applications (not perimeter pen testing)?				
Who performs the pen tests?				
If internal, are the pen testers part of the development group?				
Do you use the same approach (tools, methods, time spent, etc.) on each application pen test?				
Which applications receive penetration testing?				
Are pen testing results managed through a defect or vulnerability management system where results are assigned for remediation?				

	Vendor Response	Describe the procedure, process and tools used	If your process is not in the list, describe what you do	Additional Comments
Penetration Testing				
Do you test the complete production version of the application (not just certain components)?				
Do you currently have any unremediated Pen Test issues in the application under review?				
Do you pen test applications while authenticated?				
Is the pen testing environment production or production like?				

Production

CMVM1.1	Activity	Response	IRM Comment
This is often an initial point of identification of software vulnerabilities for less mature software security programs. When an incident is identified, what process is used to address the incident and what is the notification process with clients. Does the incident response process drive prevention activities?	Configuration Management—Incident Response/ Vulnerability Management		

	Vendor Response	Describe the procedure, process and tools used	If your process is not in the list, describe what you do	Additional Comments
Production				
Is vulnerability/security information found in operations or production shared with developers? (Describe how.)				
If a security breach occurs, which groups have mandated involvement?				
Does the incident response process include steps to identify root/cause and prevent reoccurrence? (Describe what.)				
If you host applications for the enterprise, is there a service in place to monitor production applications for vulnerabilities?				

Conclusion

Application Development Overview (vBSIMM)	
Name of the application	
What does the application do for our firm?	
ID	
Risk Ranking (H, M, L)	
Type of application (Web, Mobile, Client/Server, etc.)	
Application development language(s) (Java, .NET, iOS, etc.)	

Assessment Date/Location

Enterprise Assessor(s)

Conclusion

Artifacts Reviewed

RPs to be Created