What Measures Do Vendors Use for Software Assurance?

Jeremy Epstein

February 2009

ABSTRACT: Books and articles frequently exhort developers to build secure software by designing security in. A few large companies (most notably Microsoft) have completely reengineered their development process to include a focus on security. However, for all except the largest vendors, software security (or software assurance) is a relatively recent phenomenon, and one with an uncertain payoff. In this paper, we examine what real vendors do to ensure that their products are reasonably secure. Our conclusion is that software vendors put significant energy into software security, but there is significant variation in where they invest their money.

INTRODUCTION

Concern that software products are (in)secure has been around for more than three decades, but until relatively recently was given little attention by the vendor community. The never-ending series of vulnerabilities in Microsoft software galvanized Microsoft, and resulted in a security focused lifecycle [8]. Numerous other texts have described the risks of insecure software, including [1], [10] and [17]. More recently, an industry consortium has been formed by some of the larger software companies to define best practices for building secure software [14].

Building on the demand, start-up companies[1] have developed tools to help identify security flaws using techniques such as source code analysis (e.g., Fortify Software, Coverity), binary code analysis (e.g., Veracode), dynamic testing (e.g., SPI Dynamics[2], Watchfire[3], NT Objectives, Cenzic), as well as service-focused

[1] Inclusion in this non-comprehensive list here should not be interpreted as an endorsement by the author or his employer. Some of the vendors listed here offer products and/or services in addition to those in the list.

[2] Acquired by Hewlett-Packard Company.

[3] Acquired by IBM Corporation.

Software Engineering Institute
Carnegie Mellon University
4500 Fifth Avenue
Pittsburgh, PA 15213-2612

Phone: 412-268-5800
Toll-free: 1-888-201-4479

www.sei.cmu.edu

companies that perform scheduled scans (e.g., Qualys, White Hat Security), education and engineering analysis (e.g., Aspect Security, Cigital), or penetration testing (e.g., Matasano Security).

In addition to the commercial companies, numerous research groups have developed tools to provide similar capabilities, sometimes more advanced than the commercial products, and other times less sophisticated. Given the choices, software vendors, especially those whose primary focus is not security, have difficulty determining where to invest their resources. Additionally, for vendors whose primary products are not security technology, there may be relatively little explicit interest (or understanding) from customers, thus reducing the perceived demand [4].

In order to determine what the "best practices" are that we should follow, we did an informal survey of software vendors to determine how they achieve software security, what motivated them to put energy into software security, and related topics. This determines the "typical practices"; by learning from the typical practices we hoped to determine best practices (i.e., as the least upper bound of typical practices). This paper presents the results of this study, along with its limitations. The paper does not make recommendations of what any particular vendor should do, but rather establishes the typical practices at this writing.

We make no claims in this paper that our survey was complete or unbiased. Rather, it was intended to provide a sampling of the *status quo* (as of early 2008).

The terms "software security" and "software assurance" are used interchangeably in this paper. Jelen and Williams argue [9] that "assurance" is the confidence in the correctness of an assessment (not the results of the assessment itself), so that a piece of software could have low assurance and high security ("we think it's secure but don't have proof") or high assurance and low security ("we know it's insecure"). However, common usage is that "software assurance" refers to methods for improving the security of software, not for measuring the accuracy of an assessment.

The remainder of this paper is organized as follows. Section II of this paper describes the topics investigated in the survey, and Section III describes the study results. Section IV discusses questions arising from the survey and presents areas for future work. Section V summarizes the findings, and Section VI discusses the relationship of this study to security maturity models including OpenSAMM and BSIMM.

II. STUDY TOPICS AND LIMITATIONS

Our study addresses four basic questions: who, what, why, and when.

Who

Who in the organization is involved in software assurance? In particular, we wanted to know:

- Whether there is a centralized security person or team, or whether responsibility is distributed to each engineering team
- Who has overall responsibility for software security, and where that person reports in the organization
- Whether that person is part of the release decision process, and if so whether they have a veto (i.e., to prevent a product from being released if there are significant security flaws)

What

What does the organization do to gain software assurance? In particular, we wanted to know whether the organization:

- Performs threat modeling to determine the risk factors
- Performs security design reviews to try to avoid security problems
- Performs source code reviews (manual[4] or automated) to find implementation flaws
- Performs automated scans (including, but not limited to, input fuzzing) to find implementation flaws
- Uses penetration testing (either in-house or thirdparty) to search for more subtle design or implementation vulnerabilities
- Provides developer training (and if so, how much and how frequently) so developers can avoid introducing implementation flaws
- Has an indication (whether by gut feel or metrics) as to which technique(s) are most effective in reducing or eliminating software flaws

---

[4] Manual" code review was not defined to the participants, but based on comments during the interviews was interpreted as "not using a tool specifically designed to find security flaws". Thus, use of "grep" or Eclipse could be part of manual code review; "findbugs" would be automated review.

Why

Why does the organization have a software assurance program? For example:

- Is the interest in software assurance due to direct customer demand, avoiding notoriety, government regulation, etc?
- How often do customers ask about software assurance? Or do they just expect it's there?
- What words do customers use when asking about software assurance?
- Is the organization seeing procurement language that asks about security?
- Do customers or 3rd parties (e.g., self-styled “security researchers”)

When

When did the organization start to focus on software assurance, and how long did it take to see results?

Vendors Included and Excluded

Our study focused exclusively on vendors of shrinkwrapped software. We deliberately eliminated several other types of software developers that might be interesting:

- Custom software developers. Custom software is driven by specific customer requirements, and not by the need to find the common set of capabilities that meet the common needs of a large set of customers. As such, software assurance may be given more or less emphasis, depending on the particular customer. This category includes companies that primarily develop software for the government marketplace, including GOTS (Government Off The Shelf)[5]
- Systems integrators. Similar to the custom software developers, these vendors are driven by specific customer requirements, and not by the goal of offering shrink-wrapped software.
- Software as a service. While companies like Salesforce.com and WebEx.com have significant security concerns, they are not (generally) selling their software, but rather use of that software. This would be a logical area to extend the survey, as these vendors are most similar to the shrinkwrapped software market, and are most at-risk due to their products being publicly exposed.

---

[5] As distinguished from COTS, or Commercial Off The Shelf software, which is what the commercial software industry calls “shrink wrapped” software.

- E-commerce. E-commerce vendors such as Amazon.com have significant software investments, and are at significant risk. However, software is not their primary business, but rather a tool to accomplish their mission.
- Very small vendors. Unless they are specifically focused on security, there is little real motivation or ability for them to put energy into software assurance, although their products may be at risk.
- Embedded systems vendors (e.g., for medical instruments, cash registers). Because these are more likely to run in a constrained environment, and for some categories are more subject to regulation, we did not consider them a useful comparison to our environment.
- Direct competitors to the author's employer. We wouldn't expect cooperation from our competitors, as they might believe that we are gathering information to use against them.
- Open source projects. There is no technical reason why they could not be part of the study, but our business goal was to understand the shrink-wrapped commercial software market.

Of course, some companies fit in more than one category. For those, we made an arbitrary decision whether to include them in our survey.

Our emphasis was on medium to large software vendors. We specifically did not seek vendors who are primarily focused on selling security products such as firewalls, IDS, PKI, etc., although some of those vendors are in our sample.

The list of target vendors was selected by reviewing a list of the top 500 software vendors [16][6] and removing those who met one or more of the exclusions listed above. From the remaining list, the author focused initially on those vendors where he knew one or more employees. These employees were usually, but not always, security specialists.

In each case, the author asked his contacts for the name of the person or people responsible for software assurance. In most cases, the author was able to identify an appropriate person, and in most cases, the vendors supplied the information requested in the form of a telephone interview.

Because the author started with those vendors where he had contacts, the list of targeted vendors is somewhat skewed. Most of the author's professional peers are in the security business, and he knows many people in the industry. Thus, if the author does not have any contacts in a vendor, it may be an indication that

[6] This list is admittedly dated, but for purposes of this study was adequate.

the vendor does not have a focus on security. To reduce this bias, the author reviewed lists of attendees at security conferences to identify security specialists, and attempted to contact vendors through those security specialists. In some cases, targets were identified through social networks such as LinkedIn. These methods were less successful, as the personal contacts were more willing to be forthcoming than people who did not know the author and therefore, had no reason to trust him.

We specifically excluded Microsoft from this survey, because their security processes are well known and have been described in numerous presentations and books, especially [8]. Had we included them, their results would have shown that they use all of the techniques addressed in this paper, and have numerous motivations for practicing software assurance, most notably the impact on their reputation.

Why Not an Online Survey?

When starting this survey, we considered using an online survey with questions that could be rated objectively, rather than the interview-style described in this paper. However, there were several reasons we rejected an online survey.

First, it was not clear at the beginning what the questions should be, and what range of answers to allow. The scope of software assurance is broad, and we learned about techniques and involvement as we went through the interviews that we would not have predicted, and hence would not have been included in a questionnaire. For example, several companies included software security as part of the process of acquiring other companies, and several mentioned the importance of software security in third party and open source products.

Second, we wanted to have confidence that we were getting information from qualified participants, and in particular not from more than one person in an organization.

An open survey would have made this difficult; a closed survey (i.e., with a secret URL) might have allowed one vendor to submit multiple sets of answers.

Third, we wanted to assure the participants that their responses would be kept anonymous. Doing this by phone calls (which we did not record) gave greater confidence to the participants than putting their comments in writing. In most cases, participants did not have official authorization to participate in the study from their employers, so anonymity was critical.

Finally, when this effort was started, it was intended as an internal project to help make the case for greater investment in software assurance. It wasn’t until after several interviews had been completed and participants expressed interest in knowing the results that we considered writing a publishable report.

III. STUDY RESULTS

In this section, we discuss the results from each vendor in our sample.

The terms “small”, “medium”, and “large” in the section titles refer to the approximate size of the vendor. For purposes of this article, small means under US$100M in annual software sales, medium means US$100M to US$1B, and large means above US$1B. Sales volumes were estimated from [16].

Vendors were classified as “security” or “non-security” depending on the predominance of their sales. This distinction was useful because companies perceived as being security vendors have a higher expectation from the marketplace – customers assume that security vendors will be less likely to have security flaws than non-security vendors.

Company M: Small Security Vendor

As a small security vendor, M has both fewer resources to invest in software assurance, and a greater risk. “Small security companies get one chance” is the mantra – a serious security flaw in their product would ruin their reputation, and might well put them out of business. M does not have a dedicated software assurance team (although there is a substantial QA team), nor is there a single person with overall responsibility for software assurance other than the head of product development.

M uses several techniques to get software assurance:

- They use FIPS 140 validated cryptography (licensed from another vendor) in their products to reduce the risk of a cryptographic flaw affecting their product.
- Because their product is primarily shipped in appliance form, they expend considerable energy in stripping down the underlying operating system to reduce the risks of a flaw in the underlying product causing a vulnerability in M’s product.
- The QA process places significant emphasis on finding security flaws.
- When a new employee is hired, he/she is paired with a more experienced employee for informal security training, so that a culture of awareness is introduced. However, there is no formal training provided.

- The primary focus for software assurance is penetration testing, performed by third parties that M hires, and those hired by M's customers in the commercial and government sectors.
- M's product is written completely in Java, which insulates them from many of the common security security flaws found in commercial software (e.g., buffer overflows, integer overflows)[7].

Several of M's customers have performed detailed penetration tests, in addition to the testing that M does themselves. M does some manual source code reviews, but has found this to be expensive and relatively unproductive at finding flaws, compared to other techniques they used. There is no formal threat modeling; however the design review includes approval by product management, the chief architect, and the quality assurance director. M has considered using a honeypot to learn whether and how their product is attacked, but have not done so due to resource limitations. M does not use automated scanning or fuzzing tools to test its products, although they have used generic scanners to look for trivial flaws.

M believes that penetration testing gives them the greatest "bang for the buck" in reducing security flaws, followed by their emphasis on QA testing.

As noted above, M recognizes that bad publicity resulting from a security exploit might lead to their failure as a company, and are motivated by that more than regulation. While government customers ask about assurance provided by Common Criteria [3], M says that commercial customers rarely ask about software assurance, and then generally only to inquire about hardening of the underlying operating system. M believes that customers assume that as a security vendor, M will ensure that the product is secure.

Because M is a security vendor, software assurance has always been a concern. As the company has grown, it has been an increasing focus, with more emphasis on penetration testing.

Company W: Medium Security Vendor

As a medium security vendor, W knows its customers rely on their product to secure their systems.

[7] While integer overflows can occur in Java, to date they have not resulted in security vulnerabilities, since the JVM protects against their use to access protected memory areas.

There is no single person responsible for software assurance at W; each product team has an informal structure responsible for their own software assurance activities. Within each team, however, W has processes in place that prevent a product release if serious security vulnerabilities are found, and to promptly release fixes for any product releases that are in use by customers.

W uses the following techniques to gain software assurance:

- The QA team performs dynamic testing, source code analysis using proprietary tools, and penetration testing.
- They perform FIPS 140-2 and Common Criteria [3] testing using third-party testing labs. Unlike many vendors, W has found that CC is a useful exercise, and reports that security flaws have been found in the evaluation. As part of the CC evaluation, a US government-approved laboratory performed a source code review, and identified potential issues.
- Newly hired developers receive approximately two weeks of training in security, with much of that focused on software assurance (the remainder being on the security features of W's products). In addition, refresher courses are provided to update developers on new threats.

While significant emphasis is placed on the QA team, W does not have a formal process for threat modeling. Designs are reviewed to ensure that they meet security standards.

W finds that training is the most important factor in ensuring that their product is secure, and they put particular emphasis on hiring well-trained employees.

W's motivations are different from many of the other companies in this study. As a non-US company, their products are treated with some suspicion by the US government. Third-party testing, especially Common Criteria, help allay those fears. W is less concerned with bad publicity, and believes that investing in software assurance is the "right thing" to do for their customers. W's customers rarely ask about software assurance, probably because the brand name gives them confidence that W is doing a good job.

W's customers frequently perform their own security testing, including penetration testing, and can report results back to W. Independent third parties also test W's products, but the issues they identify are frequently due to misconfiguration.

As a security company, W has been concerned with software assurance since their earliest days.

Company F: Small Security Vendor

As a small security vendor that has been in business for a long time, F recognizes the risks of security failures. While there is not a single person responsible for software assurance (other than the head of engineering), software assurance has a significant influence at F, and can stop releases if significant security flaws are found.

F uses the following software assurance techniques:

- The primary focus of software assurance is penetration testing during the development process. F has two full-time software assurance specialists on staff (out of a total R&D staff of about 100).
- Both have worked for F for many years, and are familiar with all of the products. This allows them to rotate between penetration testing of existing products and new products, and to extend the penetration testing by building custom tools for use by QA and developers.
- The software assurance specialists also perform some manual source code review. Development teams perform their own design reviews because they're aware of the issues involved in building security products.
- A security review board that includes the software assurance specialists meets every two weeks to examine reported flaws and determine their severity. A separate bug-tracking system is used for security flaws, to ensure that access to vulnerability details is closely controlled.
- F has an incident response system that includes informing customers of problems without solutions, if the problem is severe. This allows customers to be on alert for attacks, even before a fix is available.
- To satisfy government customers, F performs FIPS 140-2 and Common Criteria evaluations of their products.
- The software assurance specialists offer training courses for developers and testers twice a year, designed to keep them up to date on threats. Much of the material in these courses comes from conferences and workshops that the software assurance specialists attend, such as BlackHat. These two-hour courses are videotaped for internal use, and F has considered selling the training as an offering. Whenever possible, these training sessions use flaws found in their own products to demonstrate the threats.

F has examined automated source code review tools, but found them to be generally inadequate due to the number of false positives, and the high per-user cost. However, they have developed some custom tools that look for particular problems that they find in their products.

F is a security company with a strong reputation, and their greatest fear is having their corporate nose bloodied by a security flaw. They want a clean reputation, “not like Microsoft”. Thus, software assurance is critical to maintaining their place in the market. Customers sometimes ask about software assurance in RFPs through questions such as “is your product ethically hacked” or “what is your vulnerability and patch management process” or “are there third-party security reports available on your product”.

Security flaws in embedded third-party and open source products have been a serious problem for F, as have been products that come in via corporate acquisitions. While F does not do source code reviews of these products, they perform a brief search for malware before incorporating third-party software, and subject it to penetration testing.

Company H: Large Security Vendor

Like many of the companies in the survey, H has grown through numerous acquisitions. A standardized software development process has been defined with security elements, but is not uniformly applied or enforced. A centralized team of security evangelists in the office of the CTO provides technical assistance to all of the product development organizations.

Software assurance techniques used at H include:

- Training throughout the company focused on architectural reviews, secure coding, and testing processes. The training materials were initially licensed from a major university, and have since been customized to their needs. H further customizes the training for product groups, to maximize relevance to the staff. While training is usually a one-time event, organizational turnover is high enough that the training is repeated in each location on a regular basis.
- In some cases, threat modeling as part of the design process.
- A company-wide license to use a source code analysis tool, along with training by the evangelist team on how to use the tool effectively.
- An in-house penetration testing team, coupled with third-party penetration testing when the need arises (e.g., because the in-house team is unavailable).
- Use of a third-party team to assess the security status of products being considered for OEM or acquisition, to minimize the risk of acquiring security vulnerabilities along with products. This review team currently operates after the OEM arrangement or acquisition has been completed. The evangelist team believes it would be more effective before the deal is signed, but that change has not occurred.

H finds that training is the most effective method for ensuring software assurance, as it reduces the problems before they occur.

H has performed several Common Criteria evaluations of their products, and has found that the paperwork-heavy process is largely orthogonal to their goals of improving software assurance.

A unique problem noted by H is geography. Like most large software vendors, H develops software worldwide, and most penetration testing is performed in one of their lower cost locations. H considered and rejected locating their penetration testing team in China due to concerns by US government customers that a Chinese national might discover a vulnerability in one of their products, conceal that fact and then later try to exploit the vulnerability. For some products, H performs penetration testing in the US, due to export limitations.

Several years ago H was hit by a string of product vulnerability disclosures, which resulted in phone calls from angry customers to corporate executives, and caused brand and image issues. Until the string of threats to H's products surfaced, executive management at H was not focused on software assurance. Thus, H's motivations are primarily to avoid a repetition of the bad publicity.

Some of H's customers test their products using open source and commercial products.

H's customers do not generally ask about software assurance, but expect that since H sells security products, the assurance is there. On occasion, RFPs will ask questions such as "what are the measures / processes used to avoid security defects". H sees more interest in software assurance from government customers.

H's customers expect the same level of software assurance whether the products were developed by H, or a third party (including open source). As a result, H must place as much emphasis on assurance of the software it does not develop as software developed in house.

Company B: Large Non-Security Vendor

B's development processes are variable through the company as a result of growing by acquisition. B offers over 100 products, some of which are focused on security and others in non-security areas. In recent years, B has begun standardizing its processes around security, including adding uniform security software assurance techniques, including:

- The entire organization uses automated source code analysis tools.
- QA teams in B are tasked to perform security testing as part of their QA efforts. QA teams use automated scanning tools to look for known vulnerabilities, as well as on-demand scans by the centralized security team when requested by QA teams. Some product teams also use fuzzing tools.

- Some groups within B use penetration testing, but
- B’s security team discourages relying on penetration testing, preferring to focus on preventing introduction of security flaws. B finds that penetration testing helps sell the need for software assurance, but doesn’t provide that assurance by itself.
- B provides a two-tier training system using outsourced training experts. The introductory training level has three modules: a one-hour overview of motivations and regulations for product management, a two-hour software development policies session for all engineers, and a three-hour class for developers and QA specialists that explains common vulnerabilities. The advanced level is two days each on Java, C/C++, and testing. All training materials are available online to encourage internal use. In addition, periodic “refresh” courses help keep developers up to date on the latest threats.
- B has an extensive product security policy which is a set of 80 criteria that define the security attributes of their products.

B finds that the most effective methods for software assurance are threat modeling (which is done as a workshop with both security experts and product specialists) and training. Several years ago B developed a “top 10” most serious problems, and focused on solving the identified problems. While B’s security team was reluctant to dictate priorities, this was a useful exercise.

B’s initial motivations were customer threats not to purchase their products if software assurance wasn’t improved. Once the initial threat passed, the motivation shifted to a customer expectation that B provided secure software as part of the quality expectation. Fear of publicity was not a significant issue.

B has performed Common Criteria evaluations, and has found that while it is a useful marketing exercise, it did little to improve the security of their products.

Customers do not frequently ask directly about software assurance, but it helps indirectly by allowing the customers to meet their internal security requirements.

B began their focus on software assurance about five years ago with internal evangelization, with a “product security office” officially starting about two years ago. As is typically the case, it took 12 to 24 months to see an impact in reduced vulnerabilities and fewer customer complaints as a result of the focus on software assurance. B’s focus on software assurance now allows them to use security as a sales differentiator.

One of the key difficulties in starting software assurance programs is corporate focus on “Return on Investment” (ROI). B notes that ROI arguments are very difficult for software assurance – security does not increase sales, but rather reduces lost accounts, unhappy customers, and bad publicity.

B’s security team reports at a very high level in the R&D organization, and is closely tied in to the QA organization. This gives the security team the influence to stop product shipment, in the same way that a serious QA problem would stop a shipment.

Company S: Large Non-Security Vendor

S’s business is organized into three divisions which are operated independently, with minimal coordination from a security perspective. The division that was interviewed reported that their focus on software assurance includes:

- Extensive focus on an architectural review panel that includes very senior staff, including a half dozen members with 30 years of security experience. In addition to considering architectural issues, this review panel ensures that security and cryptographic features are not reinvented or duplicated. The goal of the panel is depth of analysis, so reviews frequently last several days.
- Using customized versions of tools such as UNIX “lint” to search for unsafe library usage. S also uses third-party source code analysis tools on occasion, but not as a regular part of their software assurance process. However, S uses peer source code reviews before software check in.
- Any product change that creates interfaces which might have a security effect (i.e., increasing the “attack surface” of the product) goes through an architectural review that includes filing a multi-page security questionnaire. The goal of the questionnaire is to cause the designer to think about the implications of the change.
- S uses automated tests for every build which include some security tests. More extensive dynamic testing is under consideration, but is not performed today. Fuzz testing is performed on final versions of products before release.
- S does not perform penetration testing in the product development organization, but some penetration testing is done by the field staff.
- Developer training is limited to internal seminars.

S’s developers are required to use the latest version of their product to develop newer versions. This provides significant peer pressure on developers not to check in flawed software.

S finds that their most productive use of resources is peer code review and architectural reviews.

Like many vendors, S focuses on security assurance because of fear that news of security vulnerabilities will appear in the business press, which might bring bad publicity and depress their stock price. Unlike some other vendors, S found that “do the right thing” is sometimes not an adequate motivation, nor is saying “Microsoft does it”.

S is also affected by government regulation and procurement policies in the US and in several European countries. S's government customers ask about software assurance requiring Common Criteria and FIPS evaluations.

Some of S's government customers perform their own security testing in addition to that performed in the evaluation process. S finds that some of their commercial customers hire outside consultants to perform security testing.

One of the biggest issues at S is that the commitment to software assurance has slipped as the company has suffered financial setbacks and layoffs.

Organizationally, security at S is centralized, and the head of software assurance at S reports directly to the head of product development. This has given the security team the ability to stop a release if a serious security flaw is found, although management commitment to software assurance has been fairly weak.

Company K: Large Non-Security Vendor

K has grown by acquisition, and has many relatively independent product development organizations. Hence, K's processes are not standardized across the company.

K's software assurance efforts include:

- A single security expert responsible for overseeing software assurance across the company, assisted by a distributed team of engineers who answer internal questions on software assurance.
- Mandatory training for all developers (more than 75% of developers trained thus far). The training is a 90 minute company-developed course in "writing secure code" based on [1].
- Security reviews as part of the design process. A security expert reviews every product design document, which is required to have a section devoted to security considerations. However, the security expert can only provide recommendations (which are frequently assigned a low priority and sometimes ignored completely), and cannot force changes in products.
- While K's design process is designed to allow for threat modeling, no products have elected to perform that modeling thus far.
- Some products use commercial tools for automated static code analysis. Although these represent only a small fraction of the total code base, they are the most commonly used tools.
- Some product groups use commercial application scanners such as IBM/Watchfire AppScan to look for vulnerabilities. K finds these tools to be very helpful, but notes that since each application scanner finds different flaws, they sometimes create problems with customer communication; with-

out a common basis for comparison regardless of how many tools a vendor runs, a customer can run a scan using a tool the vendor does not own and find additional flaws (some of which might be false positives).

- K has performed Common Criteria evaluations of some products to meet US government requirements. They find that lower level evaluations (EAL2 and EAL3) have no impact on the security of the product because they are documentation-focused, but higher level evaluations (EAL4 and above) help improve the product by forcing a more comprehensive security review and significant security testing.
- A few products have undergone third-party penetration testing. K has no internal penetration testing capabilities.

K believes that their investments in application scanning provide the greatest payoff.

K notes that many of their products are “enterprise software”, which provides a measure of security by obscurity: while it’s easy for attackers to get access to retail software products, enterprise software, which sells for hundreds of thousands of dollars and requires custom installation and configuration, is mostly unavailable to the general public. However, the web interfaces on enterprise software products are more accessible, and hence considered riskier.

K’s motivations are primarily driven by customer expectation: just as customers expect that vendors perform quality assurance testing, K’s customers expect that the products they purchase will be secure. K believes that customers would be surprised at how little effort is invested in software assurance by vendors. In addition, K is concerned about bad publicity in case of a security flaw in their products. At K, as in many other vendors, bad press would get executive attention, and might cause K to invest more in software assurance.

K notes that non-liability clauses in software licenses protect vendors, and reduce the incentive to invest in software assurance. If K’s processes were found to be significantly worse than typical industry processes, they might be vulnerable to negligence claims. Ironically, this study, whose goal was to make a case for improving software assurance, may have the opposite effect: if the typical level for vendors is to provide minimal software assurance, then negligence would not be a risk, and hence there is little incentive to invest further.

Some of K’s customers perform application scanning using automated tools, and a few also perform penetration tests.

Organizationally, security at K is distributed, with the head of software assurance reporting to the CTO, who reports to the head of product development. The security team would probably have the ability to stop a release if a serious security flaw is found, although management commitment to software assurance has been quite weak.

Company R: Large Non-Security Vendor

R has grown by acquisition, as many of the companies in this study. However, R prides itself on integrating their acquisitions into a common software development process, including software assurance.

R provides minimal security training to its developers, and does not perform significant design reviews focused on security, makes minimal use of application testing tools, and performs minimal penetration testing. R does minimal threat modeling, as they consider that all of their products will be directly attacked by determined adversaries. Rather, R relies heavily on source code analysis to find flaws after products have been built and before they are shipped. R uses a combination of commercial analysis tools and manual analysis.

Some of R's products have undergone Common Criteria evaluation to meet US government requirements. They have not found the process to be particularly helpful as a way of assessing the security of the products.

R's main motivation for software assurance is customer expectation. Customers buy R's products because they expect them to be reliable, high performing, and secure – security isn't an option for R's customers due to their usage. At the same time, however, R only rarely receives explicit inquiries from customers about software assurance; they assume that R knows how to ensure that their products are secure.

IV. QUESTIONS AND FUTURE WORK

This survey in some respects opens as many questions as it answers. How will vendors respond to knowledge of the "industry norms"? Will those doing less than the average be encouraged to invest more and catch up? Will those at the midpoint be encouraged to continue their investments? Will those investing more be likely to reduce their investments, since there's little motivation to be "better" than average when there's little customer demand? Vendors can use "industry norms" as a defense against customer requests for better assurance – will this study, which was started to justify increasing investment in software assurance thereby (paradoxically) reduce the motivation to do more than the average? In short, will this survey motivate vendors to raise the bar, or simply to establish the *status quo*?

As was noted in section II.E, the study deliberately focused on the segment of shrink-wrapped software vendors. Future extensions to this survey could include surveys of:

- Embedded systems, such as medical instruments, automobile systems, etc. As these systems become are heavily software-based, they are increasingly at risk. For example [7] demonstrated a potential vulnerability in heart pacemakers. Hence, an examination of the software assurance methodologies used by medical instruments vendors would be helpful to understand the level of analysis being performed.
- Financial institutions, which are increasingly under attack, and are subject to regulations such as PCI [13] and OCC [11].
- Online merchants (e.g., Amazon), which are generally subject to PCI [13] requirements, and are regularly attacked.
- Software as a Service (SaaS) vendors, which may be subject to requirements including PCI [13] or HIPAA, depending on the type of information they manage.
- Systems integrators, who have the unenviable task of tying together products from different vendors into complete systems.

V. Analysis and Conclusion

Table 1 and Table 2 summarize our key findings. In Table 2, the terms "primary" and "secondary" mean that these were the driving forces, while "yes" means that it was a consideration to the vendor but not a driving force.

Table 1. Motivations for Investment

Vendor	Customer expectations	Fear of publicity	Explicit requests
M	Primary	Yes	Minor
W	Primary	Minor	Govt customers only
F	Primary	Yes	Occasional
H	Secondary	Primary	Govt customers only
B	Secondary	Minor	Primary
S	Secondary	Primary	Minor
K	Primary	Secondary	Minor
R	Primary	Minor	Govt customers only

The columns in Table 1 should be interpreted as follows:

- Customer expectations: Are customer expectations the primary or secondary reason for investing in software security? All vendors said it was at least a secondary motivation.
- Fear of publicity: Is fear of publicity (i.e., appearing in the media as the cause of a security failure) a primary, secondary, or minor consideration?
- Explicit requests: Are explicit requests from customers a primary or minor consideration (no vendor said it was a secondary consideration). Several vendors said the only explicit requests for software security come from government customers.

Table 2. Assurance Methods Used

Vendor	Training?	Design reviews?	Pentesting?	Source analysis?	Dynamic testing?
M	Informal	Informal	Internal & external	Manual	Yes
W	Formal & refresher	Not a focus	Internal, external, & customers	Proprietary tools	Yes
F	Informal & seminars	Performed by developers	Extensive internal, some external	Manual & proprietary tools	Yes
H	Formal	Informal	Internal, external & customers	Company-wide automated	Yes
B	Formal, extensive	Workshop with experts	Internal but discouraged	Company-wide automated	Yes
S	Seminars	Workshop with experts	Field only	Manual, simple tools	Minimal
K	Formal, mandatory	Performed by security expert	Varies by product	Varies by product; some automated	Yes
R	Minimal	Minimal	Minimal	Primary focus	Minimal

From this limited survey, we conclude that:

- Software vendors are aware of the risks of insecure software, and are generally motivated by fear of bad publicity to minimize the security vulnerabilities in their products.
- Few non-government customers explicitly ask for software assurance, but vendors believe that it’s an unspoken expectation.

- Most organizations have centralized security organizations that hold the expertise, with outreach into the product development teams to provide software assurance. The head of software assurance typically reports directly to the head of product development, and has a reasonable degree of influence that allows him/her to prevent product release in case of serious security flaws.
- The techniques used to gain software assurance vary among vendors, but nearly all agree that developer training is one of the most valuable uses of limited resources. While everyone agrees that penetration testing has its limitations, it is still helpful as a way to know how good or bad a product is.
- Source code analysis is still early in the acceptance phase, both because tools are expensive and difficult to use effectively. Dynamic testing, including fuzzing, seems to be more cost-effective.
- Common Criteria was mentioned by nearly all vendors, and all but one felt it was a paperwork exercise that had almost no impact on the security of their products.
- Most organizations started focusing on software assurance several years ago (perhaps influenced by the famous “Trustworthy Computing” memo [6]), and took several years to see results.

Security engineers frequently ask why vendors sell software that has significant security problems. This survey is a step towards answering that question – customers rarely ask about software assurance, but despite that, vendors are making significant strides in improving the security of their software.

VI. Related Work

The idea of surveying companies to find out about practices was also used by the developers of the Build

Security In Maturity Model [2]. BSIMM was built by synthesizing nine companies’ processes to determine “typical practices” (sometimes given the misnomer “best practices”). This study and BSIMM used similar methodologies of having a set of common questions that were asked of all participants, but also allowing open-ended answers. While this study was explicitly limited to the software development industry, BSIMM also included companies from the financial and other sectors. Thus,

BSIMM relies on a smaller set of company practices in each of several sectors, while this paper describes a slightly deeper analysis of the ISV market. BSIMM has released the names of some of the companies that participated in their study; this study included some of the same companies.

The Open Security Maturity Model [12] was developed by assessing the operation of real organizations and building a hierarchy of recommendations. However, OpenSAMM did not use formal interviews but rather makes recommendations based on the author's experiences as a security consultant.

As for timelines, this study was performed prior to either the BSIMM or OpenSAMM efforts. An earlier version of this study was published as [5].

The SAFECode group, composed of major software companies, defines a set of recommendations based on their survey of member companies' software development practices [14, 15]. Their recommendations in [14] are similar in breadth to this study; [15] is more focused on detailed coding recommendations when using C and C++.

VII. Acknowledgments

The author thanks his contacts in each of the vendors. As each of the vendors provided information about their processes on a non-attribution basis, he regrets that he is unable to thank them by name. He also thanks the anonymous reviewers, who provided useful suggestions.

VIII. REFERENCES

[1] *19 Deadly Sins of Software Security*, Michael Howard, David LeBlanc, and John Viega, McGraw-Hill, 2005

[2] *Building Security In Maturity Model*, http://www.bsi-mm.com/

[3] *Common Criteria for Information Technology Security Evaluation*, ISO/IEC 15408.

[4] J. Epstein, S. Matsumoto, and G. McGraw, "Software Security and SOA, Danger Will Robinson!", IEEE Security & Privacy magazine, February 2006.

[5] J. Epstein, "What Measures do Vendors Use for Software Assurance?", in *Making the Business Case for Software Assurance Workshop*, Carnegie Mellon University Software Engineering Institute, September 2008.

[6] Trustworthy Computing (memo), Bill Gates, Microsoft, January 15 2002, http://www.microsoft.com/mscorp/execmail/2002/07-18twc.mspx

[7] D. Halperin et al, “Pacemakers and Implantable Cardiac Defibrillators: Software Radio Attacks and Zero-Power Defenses”, in *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland CA, May 2008.

[8] *The Security Development Lifecycle*, Michael Howard and Steve Lipner, Microsoft Press, 2006.

[9] G. Jelen and J. Williams, “A Practical Approach to Measuring Assurance”, in *Proceedings of the 14th Annual Computer Security Applications Conference*, Phoenix AZ, December 1998.

[10] *Software Security: Building Security In*, Gary McGraw, Addison-Wesley, 2006.

[11] *Application Security*, OCC Bulletin 2008-16, May 2008, http://www.occ.treas.gov/ftp/bulletin/2008-16.html

[12] *Software Assurance Maturity Model*, www.opensamm.org

[13] *Payment Card Industry Data Security Standards*, version 1.1, September 2006, https://www.pcisecuritystandards.org/security_standards/pci_dss.shtml

[14] *Software Assurance: An Overview of Current Industry Best Practices*, February 2008, www.safecode.org

[15] *Fundamental Practices for Secure Software Development: A Guide to the Most Effective Secure Development Practices in Use Today*, October 2008, www.safecode.org

[16] *The Complete Searchable 2007 Software 500 Database*, Software Magazine, http://www.softwaremag.com/S_FocusAreas.cfm?Doc=The500

[17] *Building Secure Software: How to Avoid Security Problems the Right Way*, John Viega and Gary McGraw, Addison-Wesley, 2001