Software Security Vulnerabilities: Baselining and Benchmarking*

Pete Rotella
Cisco Systems, Inc.
7200 Kit Creek Road
Res. Triangle Pk., NC, USA 27709
+1-919-392-3854
protella@cisco.com

ABSTRACT

The security of a company's software products is of paramount importance, of course, and arguably even more important than software reliability and the other key quality attributes. But companies are currently faced with a troublesome dilemma: Supplying customers with more features at greater speeds than in the past has become the norm; high feature velocity, fairly static engineering headcounts, and shorter release cycles are conspiring to threaten both software reliability and security. The work described in this paper is an attempt to baseline and (internally) benchmark the state of our company's software security, and also includes some data regarding the state of software reliability across the company's products. Of particular interest in this study is learning more about the extent of software vulnerabilities emanating from the open source software that we import and use in our commercial products. Prior evidence had been building that suggested that such 'third-party software' (TPS) is inherently more vulnerable to security (and reliability) problems. We have examined the software vulnerability occurrences across all the company's software, in the aggregate, and have found that the TPS used in our products, primarily open source software, initially contains more vulnerabilities than internally-produced software. Security and reliability problems, both in terms of bug counts and percentages of total code volume, correlate quite well, and examples of this are also shown, but we cannot rely on this concurrence in our study: Software security on its own has been examined in detail, and while some findings are documented here, many questions remain.

CCS CONCEPTS

• **Security and Privacy → Software and application security → Software security engineering**

* Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

SEAD'18, May 27, 2018, Gothenburg, Sweden
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5727-2/18/05…$15.00
https://doi.org/10.1145/3194707.3194708

KEYWORDS

Software security; vulnerabilities; CVSSv2; SIR ratings; third-party software; open source software; security baselines; security benchmarks; customer-found defects; software reliability.

1. INTRODUCTION

Without historical baselines and internal benchmarks (and, hopefully, industry benchmarks also), it is not feasible to reliably gauge how well engineering teams are controlling the levels of security vulnerabilities in the organization's commercial software products. We need to know if our security development lifecycle practices and processes are effective in keeping vulnerabilities at very low volume levels and severities. We also need to quantify vulnerability levels and severities in specific software products and in the specific releases of those products.

Using the baseline and benchmark data can enable us to:

- Reduce the number of high severity vulnerabilities in our software products.
- Reduce the total number of vulnerabilities of all severities.
- Identify engineering practices and processes that are effective in reducing vulnerabilities.
- Identify engineering practices and processes that are not very effective, with an eye to modification, replacement, or elimination.
- Improve on ways to promulgate security development lifecycle best practices.

2. ORIGIN OF VULNERABILITIES

In addition to comparing vulnerability rates to industry averages and best-in-class rates, we need historical baselines to track how well we are doing in controlling vulnerabilities, particularly Common Vulnerability Scoring System, Version 2 (CVSSv2) Critical (9-10) and High (7-9) severity vulnerabilities [1]. Baselines should go back several years, but further back in time than that, the rapid evolution of security practices and processes usually disrupts the integrity of the resulting trends. In this paper, we show history back to

2010 in some cases, but primarily back to 2013, to compare the current state with that of the past.

The security vulnerability bug rates are sometimes, in the analyses described, compared with reliability bug rates. The vulnerability and reliability trends in some cases follow the same pattern. We will show, in Section 3.5, that there is evidence that in several respects, security bugs are a fairly representative subset of what we typically catalogue as reliability problems. The distinctions between the two types of bugs are not well understood, but we do see high correlations and several other behavioral similarities between these two bug types. These similarities will be further investigated in future work (see Section 6, Next Steps).

3. SECURITY IMPACT RATINGS

Internally, we use a derived 'Security Impact Rating' (SIR) designation, which is essentially the CVSSv2 score condensed from 100 possible scores (from 0.0 to 10.0) down to five possible score categories, as shown in Table 1:

Table 1: SIR ranges, based on CVSSv2 scoring

Security Impact Rating		
SIR Category	CVSSv2 Equivalent	SIR Index
Critical	9.0-10.0	1
High	7.0-8.9	2
Medium	4.0-6.9	3
Low	0.1-3.9	4
No Impact	0.0	5

The SIR rating system is a company-specific method used to cluster the security vulnerability severities and thereby enable us to better clarify the aggregate trends and also simplify the overall analysis.

3.1 SIR Volumes: Internal and TPS Bugs

Across the company, for all the software products, there has been a fairly steady rise in the SIR *volumes* (for the Critical, High, Medium, and Low categories) throughout the past three years, except that over the past two years, the numbers of 'No-security-impact' SIRs have actually declined (see Figure 1). The sharp increase in No-security-impact bugs in 2014-2015 is attributable to increased security scrutiny that was the result of a high number of reliability and security bugs caused by the inclusion of a large amount of third party software (TPS) code in our commercial products. In 2016, modifications were made in the security screening method, for all code, which reduced the number of 'false positives,' particularly for TPS code, but also for internally-developed code. Despite this improvement in the security screening process, we continue to observe a steady rise in the numbers of vulnerabilities in the four key SIR categories, Low through Critical.

In Figure 1, the heightened 'No-security-impact' level seen in 2014-2015 is also partially attributable to increased scrutiny of internally-developed code, a side effect of the TPS focus. A secondary cause of the rise in the higher severity vulnerability populations is the security 'challenge' faced by the internal teams during that period. Security declines for the internally-developed code appear to be roughly proportional to the gradual rise in reliability problems for the internal code during that period. High feature velocity impacted both reliability and security, but not as much as the introduction of large amounts of third-party code. See Sections 3.2 and 3.3.

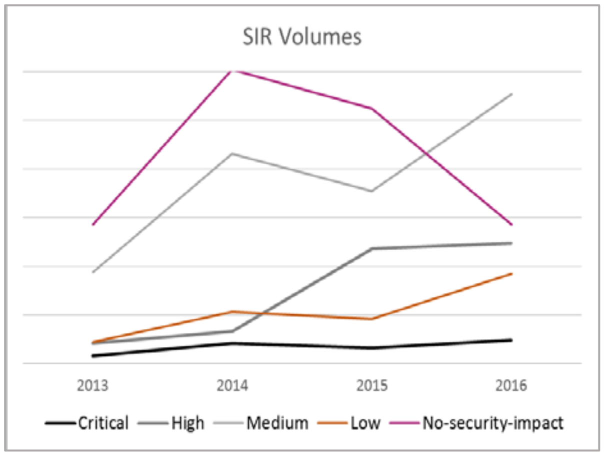

Figure 1: SIR-rated internal bugs volume, by year

From 2010 to 2014, we had been tracking the vulnerability levels per KLOC (thousands of lines of new source code) for internally-developed code, and the product-specific and aggregate SIR rates (for all SIR categories) had been observed to be fairly consistent during that period. But in 2014 and into 2015, we incorporated more TPS and also found more vulnerabilities in internal code. All vulnerabilities were remediated prior to software release, but we mobilized, of course, to understand and correct deficient development practices. Increased focus on adherence to the Cisco Security Development Lifecycle (CSDL) processes and practices was achieved.

Third-party software (TPS) is used in a wide range of the company's software products. Most (>90%) of this imported software is open source software (comprising ~5% of the company's product code), rather than commercial off-the-shelf software (COTS), which now comprises an additional ~1% of the company's product code.

'IFDs' in this paper refer to 'internally-found defects,' bugs found by the Engineering teams prior to the software's release to the field; CFDs refer to 'customer-found defects,' bugs found by the customers after the software is released (but not counting beta testing, early field trials, and field testing bugs – these have not been included in the analyses described).

We have anonymized the y axis values on nearly all the figures in this paper. This has been done to maintain the privacy of company confidential information. Except where explicitly noted, the base y level in each of the figures is at zero bug count or percentage, as applicable.

The Critical plus High SIR volume has steadily increased since 2013, increasing by a factor of four over the three years prior to 2017. Is the volume of Critical plus High SIRs increasing because of a higher volume of code being released to customers, or higher scrutiny of TPS and internal code, or is all new code becoming less secure, or all three? We address this question in Section 3.4, but whatever the cause(s), we see sharp volume increases over the past three years in the SIR Critical and High ratings for all code (Figure 2), internally-developed code (Figure 3), and TPS code (Figure 4).

Figure 2:

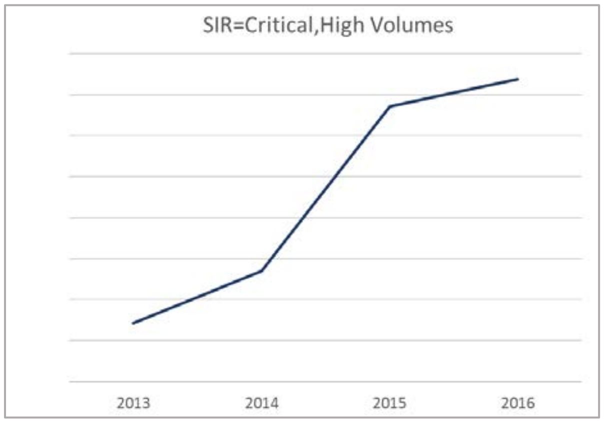

Figure 2: Critical+High SIR volume

Figure 3:

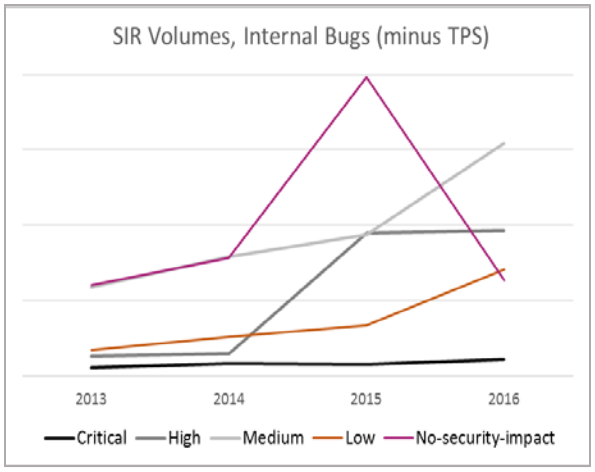

Figure 3: SIR volumes for non-TPS internally-found bugs

Figure 4:

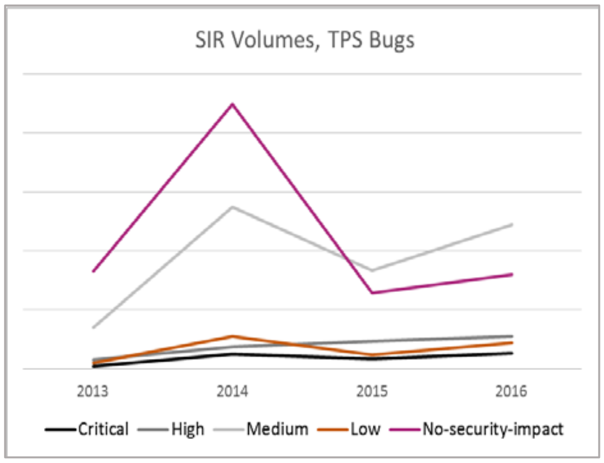

Figure 4: TPS SIR rates

As we see in Figure 3, the High and Medium SIR volumes for non-TPS internally-developed code have seen increases over the past two years for High SIRs, and over the past four years for Medium severity vulnerabilities. Critical levels have increased slightly, but remain low.

The rise in SIR volumes has partially been the result of improvements in inspection and testing, but the major cause is the poorer quality of the code imported. As a next step (see Section 6), we intend to better quantitatively separate out these and any other major influential factors. Also, will TPS SIR volumes continue to rise further because of increased use of TPS code in the company's products?

All internally-found vulnerabilities are remediated prior to the software's release to the field; all externally-found (i.e., CFD) vulnerabilities are given the highest remediation priority in Engineering, with our most aggressive mean time to repair standards and goals.

3.2 SIR Percentages: Internal and TPS Bugs

We see a steep rise in Critical plus High SIRs also as a *percentage* of all bugs, both internally-found and TPS bugs, over the past two years See Figure 5:

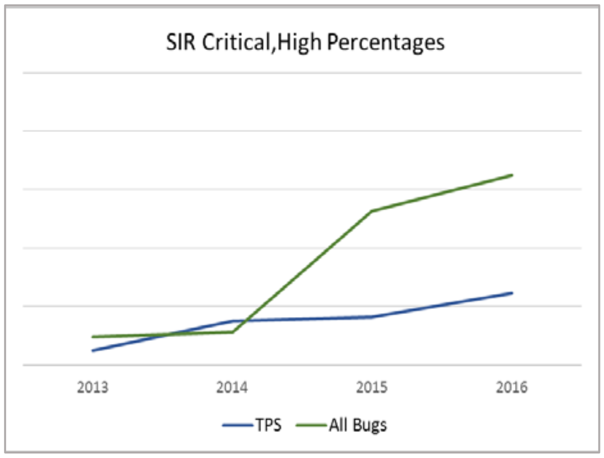

Figure 5: Critical plus High SIRs for TPS plus internal code ('all bugs'), as percentages of total SIR population

The percentage of Critical plus High SIRs has increased almost five (i.e., 4.7) times over the past two years. However, the Critical plus High percentage as a fraction of *all* SIRs has increased less, 2.4 times.

The SIR percentages for TPS bugs have also been substantially increasing. Figure 6 shows the steep increases for all key SIR categories. Recently, the ratios (not volumes or percentages) for TPS bugs are similar to those of all internally-found bugs. Figure 6:

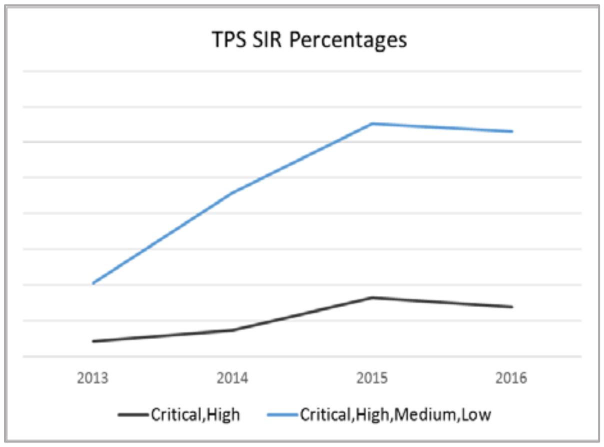

Figure 6: SIR percentages for TPS bugs

The TPS Critical plus High percentage of all bugs is lower than the percentage for internally-developed code (although both have increased over the past several years), but, as shown on Figure 7, the volume of TPS code is much lower than that of internally-produced code. TPS Critical plus High percentage of SIR bugs has increased 4.5 times over the past three years, and for internally-developed code the increase has been 1.3 times. TPS-caused bugs currently comprise roughly 6% or our total bug population. See Figure 7:

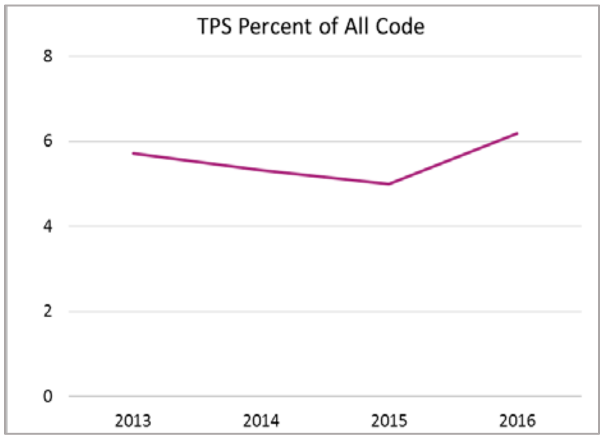

Figure 7: TPS-caused bugs

At this time, we cannot determine whether or not the bug content per KLOC of TPS code is or is not a problem. We normally use an internal 'aging wheel' to discount the defect density for internal code ported from other release trains, since we know how long the code has been subject to internal testing and how long it has been exposed to the field. For TPS, we would need to use such an 'aging wheel' approach to estimate defect density rates, and thereby ascertain whether or not the discovered rates substantiate a reliability, or a security, problem.

The total SIR population, as a percentage of all bugs, has increased by a factor of 3.4 over the past four years. See Figure 8:

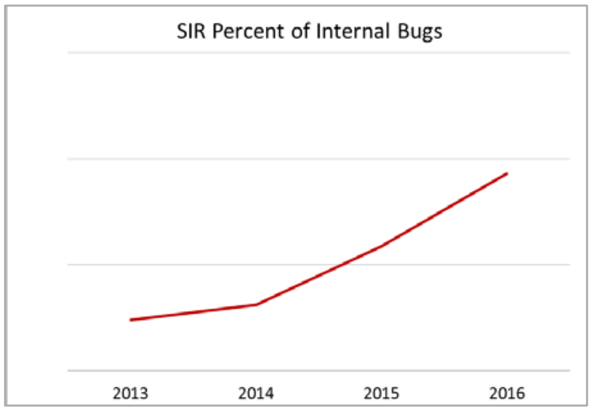

Figure 8: Total SIR (except SIR=No-security-impact) population

3.3 SIR Ratios: Internal and TPS Bugs

TPS code is currently 6.3 times more likely than internally-developed code to cause a Critical or High SIR-rated bug, factoring in the code quantities of each, as described earlier and shown in Figure 7. This rate had been higher in 2014 because of higher scrutiny of TPS code, but scrutiny (i.e., the percentage of all bugs evaluated to detect security vulnerabilities) has been at approximately at the same level as for internally-developed code since 2015. See Figure 9:

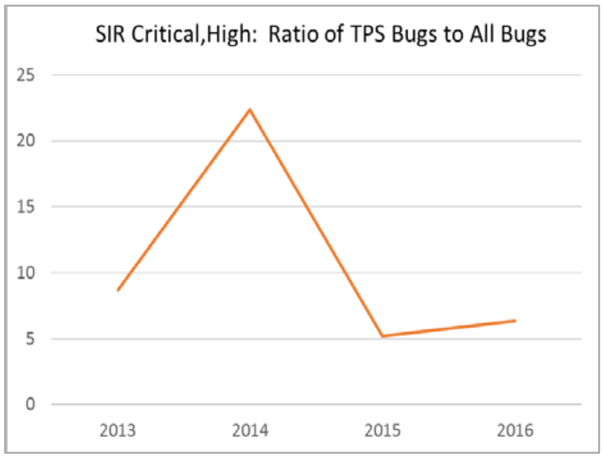

Figure 9: TPS to internal code ratio for Critical, High SIRs

Given this factor of ~6, we calculate that ~30% of the Critical plus High SIR content of the company's software is caused by the 6% of the code base (see Figure 7) that originated as TPS code. We currently use several reliability prediction mechanisms [2-6], in a formal manner, prior to importing TPS code, and these have confirmed a similar reliability problem with TPS. Several other studies have looked at the security of open source code, but there is little agreement on the state of open source security [7-13].

In Figure 10, we see that TPS code's Critical plus High rate, when compared with that of all SIRs, from both internal code and TPS, shows that TPS code is roughly twice as likely to cause a Critical or High SIR than internally-developed code. See Figure 10:

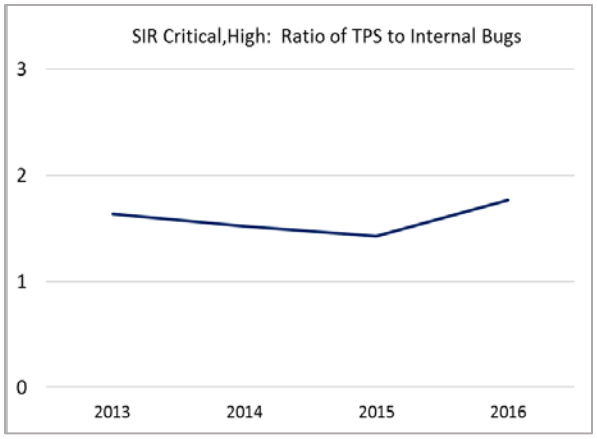

Figure 10: TPS code's Critical+High rate compared to all SIRs

3.4 CFD SIRs: Internally-developed and TPS Bugs

Figure 11 shows a representation of the numbers of yearly customer-found defects (CFDs) in each SIR category:

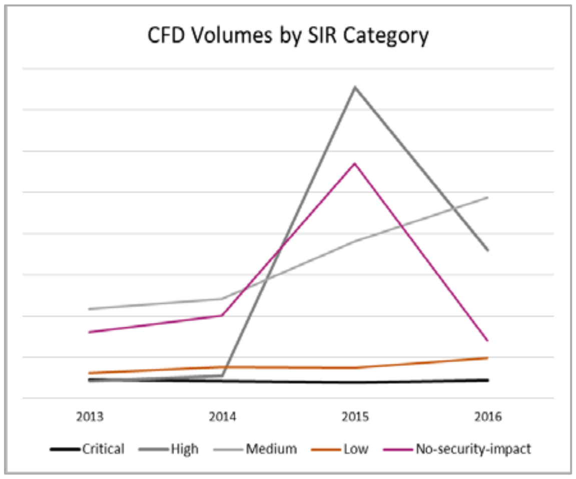

Figure 11: CFD volumes by SIR category

Unfortunately, CFDs are 4.1 times more susceptible to reaching a Critical or High vulnerability rating than internally-found defects (IFDs). (For all SIRs, including Medium, Low, and No-security-impact, the rate delta is 2.7 times). CFDs are 3.6 times more likely than IFDs to be evaluated for SIR rating, because of a much higher and closer scrutiny of customer-discovered bugs.

We are currently engaged in better quantifying the rates attributable to higher scrutiny alone, compared with the rates associated with more releases available to the customer base. And there are several other comparisons between TPS and internal-code that we are investigating. These are critical quantifications and may affect how we conduct our CSDL practices and processes in the future. This investigation is part of a more expansive one in which we are currently evaluating the effectiveness of CSDL practices, first concentrating on penetration testing and threat modeling practices.

As with internal bugs, CFD SIR volumes have been increasing, especially in the past two years. In this period, the Critical- and High-rated CFD rate is 5.1 times greater than that of the previous two years. Approximately 65% of this increase in 2015 is attributable to TPS vulnerabilities, and for 2016, the TPS portion is about 35% more likely to be rated Critical or High SIR than IFDs, and 2.7 times more likely to have any of the key vulnerability ratings. See Figure 12 for the internal and TPS combined effect on the customer space:

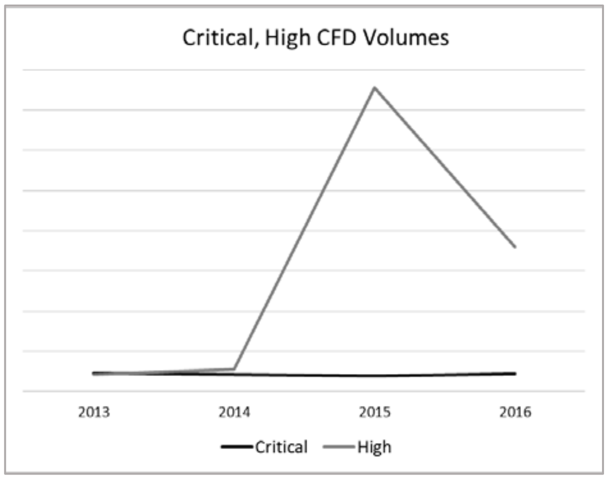

Figure 12: Critical, High CFD volumes

A fairly large percentage of the SIR population comes from CFDs, but the most disturbing aspect is that the Critical-rated SIRs have, on average over the past four years, comprised a higher percentage of the total SIR population than the Medium- and Low-rated CFD SIRs combined (see Figure 13). This phenomenon is being investigated: So far, it appears to be that Engineering, to be extra careful with customer-found bugs, assigns elevated rates to make sure adequate attention is paid to their remediation. Of a lower probability is that escapes to the customer tend to be more serious, and in these cases we are relying on our follow-on studies examining the effectiveness of several key CSDL practices and processes.

The high spikes seen in 2015 are the result of the large influx of problematic TPS security bugs, following the 6- to 9-month time lag from the high volume of 2014 TPS imports. The %CFD level has also been rising, and this is shown on Figure 13 also, but pertains to a separate, right-hand-side y axis:

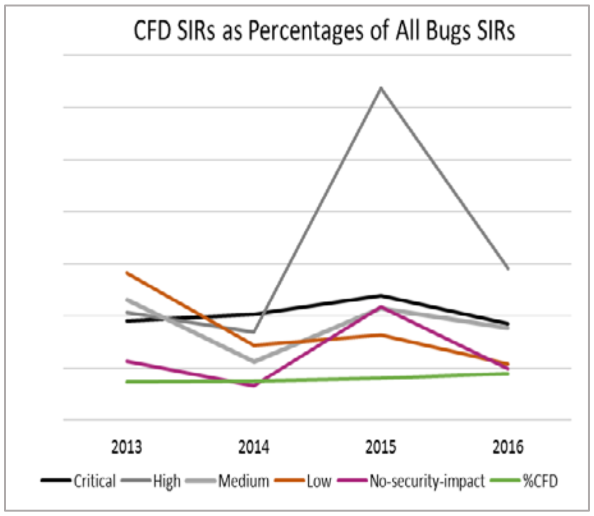

Figure 13: CFD SIRs, percent of all bugs SIRs

(%CFD is the percentage of all bugs, internally- and externally-found, that are found externally, by customers.)

In Figure 14 below, the trend line combines Critical and High populations: In 2015, the percentage of Critical plus High SIRs for CFDs is 3.1 times higher than the rate for IFDs. However, this rate is much better than the 7.3 rate seen in 2015, when a large number of TPS vulnerabilities entered the customer space.

The high rate of CFD SIRs, particularly the highest severity SIRs, may be occurring for one or more of a number of reasons, including:

- Higher scrutiny of CFDs
- Higher intrinsic vulnerability of the CFD bug population
- Higher vulnerability of TPS code.

Are typical high severity reliability escapes to the field (i.e., CFDs) more prone to also be high severity security escapes (i.e., CFDs having SIR=Critical or SIR=High)? Also, are we two to three times, as seen over the past four years, more attentive to CFD security concerns than to internal bug security concerns? These are key questions that we are attempting to answer.

We do know that over the past four years, CFDs contribute 3.7 times as many Critical plus High SIR ratings than IFDs; the rate in 2015 is 2.6. See Figure 14:

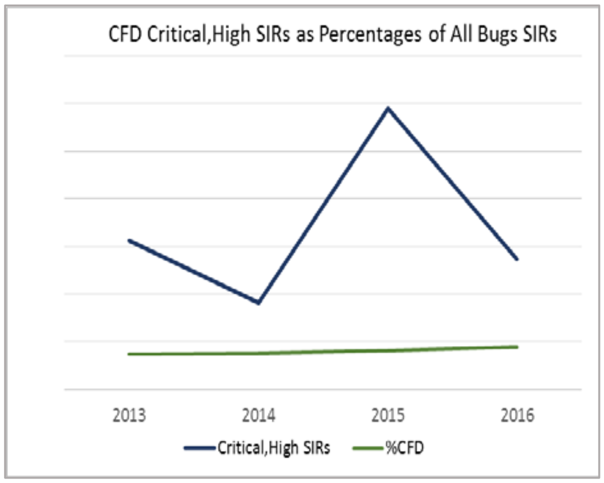

Figure 14: CFD Critical, High SIRs, percent of all SIRs

3.5 CFD and SIR Trends

The corporate trends, over the past six years for both CFD Incidents and security vulnerability bugs, are fairly similar (see Figure 15). (CFD 'incidents' are the number of times CFDs are encountered by customers; for example, 10 customers encountering a single CFD would cause 10 CFD Incidents.) Over this six year period, there has been a 36% increase in incidents, mostly attributable to an increased number of products and customers in this timeframe, and a 77% increase in the number of identified CFD vulnerabilities, partially attributable to enhanced testing and other CSDL processes applied to CFD bugs, but, equally important, to the large increase in added TPS functionality and its concomitant security bugs. Currently, the ratio of vulnerabilities to CFD Incidents is ~ 1:65, whereas that ratio, in 2011, was ~1:102. See Figure 15:

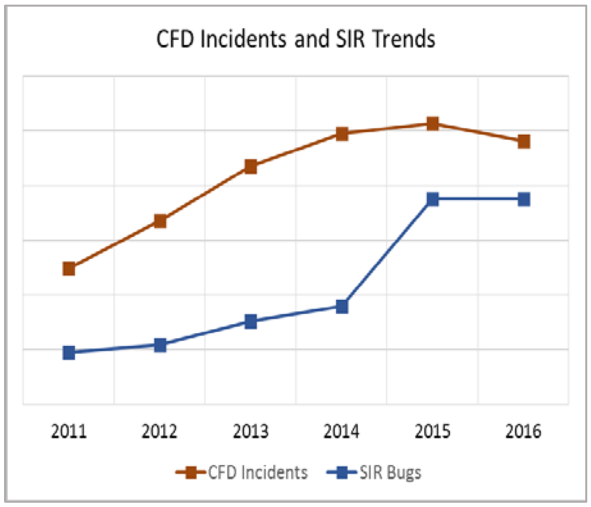

Figure 15: CFD Incidents, SIR bugs

Similarly, the corporate trends, over the past six years for both %CFD and %SIR, are fairly similar in shape. There has been an appreciable (but nevertheless still close to the best-in-class value) increase in the ratio of field bugs to all bugs (i.e., %CFD), and a similar increase in the ratio of SIR bugs to all bugs (i.e., %SIR). (The left-hand side y axis (orange/upper line), for %CFD, and right-hand side y axis (blue/lower line), for %SIR, have different scales in this graph.) Currently, the ratio of SIRs to CFDs is ~1:31, whereas that ratio, in 2010, was ~1:52. See Figure 16:

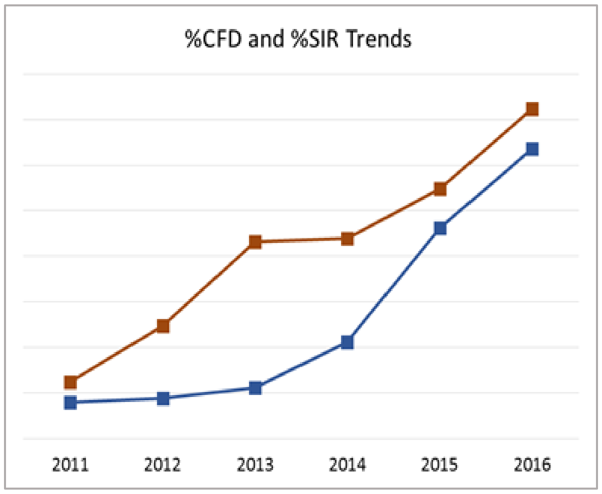

Figure 16: %CFD (orange/upper) and %SIR (blue/lower) trends

4. SUMMARY

Listed here are conclusions drawn from these recent investigations, and a brief summary:

- Critical plus High SIR-rated bugs have increased in volume by almost four times over the past two years.
- CFDs are three to four times more likely than IFDs to be SIR-rated Critical or High.
- All internally-found vulnerabilities are remediated prior to the software's release to the field; all CFD vulnerabilities are given the highest remediation priority in Engineering.
- TPS Critical plus High SIR bug volume has increased by ~130% over the past two years, but, more important, TPS code is now 6.3 times more likely to receive a Critical plus High SIR rating than internally-developed code.
- There has been a sharp volume increase (~2.6 times) over the past two years for all SIR ratings. These increases occur in both internal and TPS code.
- The percentage of Critical plus High SIR bugs has increased 4.7 times in the past two years. However, the fraction of all SIRs has increased less, but still 2.4 times.
- TPS Critical plus High SIR bugs have increased 3.5 times over the past three years, but all internal bugs have seen a 4.7 times increase.
- TPS code is 6.3 times more likely to receive a Critical or High SIR rating than internally-developed code. Although TPS code comprises only ~6% of all product code, the Critical plus High SIR ratings for TPS outnumber internal ratings by a factor of ~2.
- SIR vulnerability designations occur in <1% of all internal code (internally-developed plus TPS).
- As with internal bugs, CFD SIR volumes have increased substantially, especially in the past two years. CFDs are 4.1 times more likely to be rated Critical or High SIR than IFDs, and 2.7 times more likely to have any vulnerability rating.
- Over the past six years, CFDs contribute 3.7 times as many Critical plus High SIR ratings as IFDs have; the rate in 2015 is 3.1.

5. THREATS TO VALIDITY

We have not fully quantified the contribution of increased security scrutiny of all bugs, emanating from both internally-developed code and from third-party software, to the increases seen in recent years in vulnerability volumes and comparative percentages. There is considerable evidence that shows that internally-developed code is less of a security problem than third-party code, but the scrutiny question needs to be more thoroughly examined and better quantified.

Likewise, the higher vulnerability rates seen with customer-found bugs needs to be better understood. We know that the higher rates are largely the result of greater scrutiny, but we need to better quantify this. We also need to better understand the reason for higher percentage of vulnerability escapes to the customers? Likewise, higher scrutiny has been shown to be the major factor in this, but all contributions need to be better understood and quantified, and this effort is underway.

6. NEXT STEPS

These are our next steps in this investigation, several of which are already underway:

- As mentioned above, we need to continue to separate out the contribution of greater scrutiny from that of lower code reliability and security, particularly for third-party code and customer-found bugs.
- External benchmarking is underway to attempt to understand average and best-in-class SIR volumes and percentages, both for internally-developed code and for third-party software.
- Reliability and security bugs usually track fairly well together, and there is internal evidence that they are in many ways related – we are continuing to investigate the relationship between these two quality attributes.
- Does the ratio of reliability to security bugs have a profile that is specific to a given Engineering group? If so, does this suggest that different groups are better at certain CSDL practices than others? Maybe this is a reasonable 'benchmarking' approach to identify groups that have low practices effectiveness? This study is underway, and is part of a larger study of security practices effectiveness.
- How can we assess third-party vendors' software security before we allow its internal use, and how do

we set security thresholds for imported code? This work is also underway.

- Can we develop an 'aging wheel' approach for third-party code to gauge defect density levels for imported code? We have made a start with this work.

REFERENCES

[1] Mell, P., Scharfone, K., and Romansky, S. 2007. A Complete Guide to the Common Vulnerability Scoring System v 2.0; (June, 2007); https://www.first.org/cvss/cvss-v2-guide.pdf.

[2] P. Rotella and S. Chulani, "Predicting Release Quality," ISSRE 2014 (IEEE International Symposium on Software Reliability Engineering), Naples, Italy, November, 2014.

[3] P. Rotella, S. Chulani, and D. Goyal, "Predicting Software Field Reliability," ICSE 2015 (IEEE ACM SIGSOFT International Conference on Software Engineering), 2nd International Workshop on Software Engineering Research and Industrial Practice, " Florence, Italy, May, 2015.

[4] P. Rotella, S. Chulani, and D. Goyal, "Release Quality: Modeling and Predictions," ESEC–FSE 2015 (European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering), Bergamo, Italy, September, 2015.

[5] M. Gegick, P. Rotella, and L. Williams, "Predicting attack-prone components," in Proceedings of the International Conference on Software Testing Verification and Validation, Denver, Colorado, USA, April. 2009, pp. 181-190.

[6] M. Gegick, P. Rotella, and T. Xie, "Identifying security fault reports via text mining," in Proceedings of the International Conference on Software Engineering, Cape Town, South Africa, May, 2010 pp. 526-536. DOI: 10.1109/ MSR.2010. 5463340v.

[7] H. Plate, S. Ponta, A. Sabetta, "Impact assessment for vulnerabilities in open-source software libraries," ICSME 2015 (31st International Conference on Software Maintenance and Evolution), Bremen, Germany, September, 2015.

[8] J. Caballero, E. Bodden, and E. Athanasopoulos, "Engineering Secure Software Systems," ESSoS 2016 (8th International Symposium Engineering Secure Software and Systems), London, England, April, 2016.

[9] R. Clarke, D. Dorwin, and R. Nash, "Is open source software more secure?, https://courses.cs.washington.edu/courses/ csep590/05au/whitepaper_turnin/oss(10).pdf.

[10] E. Erturk, "A case study in open source software security and privacy: Android Adware," World CIS 2012 (World Congress on Internet Security), Guelph, Canada, June, 2012, pp. 189-193.

[11] A. Mahboob and J. Zubairi, "Securing SCADA systems with open source software," HONET-CNS 2013 (10th International Conference on High Capacity Optical Networks and Enabling Technologies), Magosa, Cyprus, December, 2013, pp. 192-198.

[12] R. Pandey and V. Tiwari, "Reliability issues in open source software," in International Journal of Computer Applications, Volume 34, No. 1, November, 2011, pp 34-38.

[13] L. Aversano and M. Tortorella, "Analysing the reliability of open source software projects," ICSOFT 2015 (10th International Joint Conference on Software Technologies), Colmar, France, July, 2015.