Security and 'time'

Ben Tomhave wrote a decent post musing "Which came first: The Software or The Security?".

In particular, Ben asks whether the response an organization has to its security problems should possess a time component. "Yes", he answers his own question emphatically. I agree, and for a few reasons worth expounding on.

…the only difference between today and yesterday
If you're effective at assessment, you'll often leave an organization's apps in shambles. Somewhat older data I have indicate an expectation for 10 critical vulnerabilities / 100KLoC, with 30-40 high-risk vulnerabilities following suit(*1). Organizations holding their broken app at arms length panic. "We've got to pull it from production!", they exclaim. Others take a different approach, "We'll stop everything until those vulnerabilities are fixed–no more features."

This behavior is irrational and needs to be quelled. The only difference between "today" and "yesterday" in this situation is that the organization *knows* how screwed they are (*2).

Ben indicates that visible risks should be prioritized based on a risk management framework. Ostensibly, we're very much in agreement on that one. There may be vulnerabilities that deserve such "stop the press" kind of treatment, but there is often a normalization process that must occur before that set is culled from an assessment (*3).

After that triage, the rest (of the vulnerabilities), and I've said this before, should enter the change-management/bug-tracking process like everything else: features, customer complaints, and so forth. Dealing with security vulnerabilities in-band both sobers security analysts' dreams of the all-important 'sploit and raises the respectability of security requirements within the normal development process by treating them as first-class citizens of a release process.

Depending on whether an organization has rapid-turn around (as happens when development engages in weekly sprints, organized through SCRUM) or more monolithic develop cycles, these change requests can be weighed from the perspective of whether or not they should be:

1. Released out-of-band as a patch
2. Included in the next planned release, potentially:
    - At the expense of other functions/features
    - At increased expense,

   or:
3. Deferred until a future release

Sometimes, the ‘correct’ answer is to conduct a usability study, or revisit the SLA promised by a system/service, and get feedback as to what form the fix will take before any of the above can happen.

In summary, time continues to elapse at a constant rate. In this case visibility is increasing. Visibility should allow for a better risk management decision, not a knee-jerk reaction.

Moreover, it’s important to discern the difference between 1) the organization’s increased visibility into vulnerabilities, not 2) an increased *discoverability* of the vulnerability itself. Yes, if the assessor used an automated pen-testing tool to find the vuln., it is probably pretty discoverable, but the organization’s knowledge of the vuln. does not *increase* it’s discoverability(*4).

Onto the next element of time…

0 to 60mph in?
Ben also rightly indicates (paraphrasing) knowing there’s a gap between you and the security posture you’d like to have, you’re not going to get there instantaneously. Very true. This, in my mind, is where organizations could improve the most. Most organizations I talk to have dashboards showing quantities like

- Percentage of applications in compliance with assessment policies;
- Number of outstanding critical + high vulnerabilities;
- Time to remediate critical + high vulnerabilities;
- <Metric du jour>

However, when asked-from the simplest risk perspective-”Are you any less vulnerable to phishing this year?”, or, “Can you provide a greater assurance case around the [confidentiality] of [this key asset]?”, the answer is <blink> <blink>. Yes, there are some exceptions to this.

I’d like to see organizations begin managing to security goals like:

> …Next year, we expect no application providing access to [asset X] to be vulnerable to discoverable by an external Threat, be he/she authorized or not, accessing our systems only through the web;
>
> …in three years, we expect no vulnerability that would provide access to [asset X] with any Internet-facing software, regardless of immediate discoverability or exploitability (by an external threat);

> …in that same three-year time frame, we also expect no Internet-facing apps. to fall prey to [common web attacks, listed in the SANS Top 25(*5) (aka: CSRF et. al.)] that would result in impersonation and therefore allow for inappropriate access to [asset X] on behalf of an unwitting user.

I'm attempting to extend Ben's call for a time-based (I'll call it iterative) approach to responding to security gaps. The previous paragraph proxies for what a more advanced risk management framework would incorporate in the form of probability (discoverability x exploitability x [other factors]). Remember, however, that the [asset X] clauses of these statements proxy for different but important factors in your risk model: intrinsic value of the information asset.

So, whether you're using a formal risk management framework, or doing it more informally, you can respond iteratively and get time back on your side: having a story about how you're measurably reducing organizational risk in a meaningful fashion.

Compare this with the alternatives:

1. A straight linear approach – fixing 80% of XSS bugs within an application has what effect on the overall security posture?
2. The bug-of-the-month approach – 'fixing' 100% of the CSRF problems within an app, but leaving all the XSS has what effect on the overall security posture?
3. Stopping the presses – fixing 100% of the bugs, and not allowing business functionality to evolve in the balance does what to your business opportunity/revenue stream?

And I think you'll be satisfied with the trade-off (through formal risk modeling or informally).

Finally,

Within a single vulnerability's remediation, I've got trade offs
Yes, within the scope of a single killer finding, organizations have to decide how to trade off the time it takes to field a mitigation and the effectiveness of that mitigation on the security posture (queue the "dynamic patching" vs. "fix it in the code and re-deploy" flame).

This, just like our previous two topics, is a matter of risk management. I implore organizations to consider what the capabilities and motivations of their opponent are in this case. If you're considering protecting against a web-vulnerability delivered through a URL parameter, a WAF rule will prevent a vulnerability scan from finding the 'fixed' problem but is unlikely to thwart a skilled penetration tester from manipulating the order of the innocuous and malicious parameters. WAF rules, then, are likely to provide a rapid response to failed scans and simpleton attackers but not concerted attacks of this sort. This same sort of analysis can be done on code-based fixes. When assessments find input validation problems, the development team will often respond by doing a simple (arbitrary) length check, or by black-listing particular SQL characters.

When I was quoted in a Darkreading article as saying costs to fix were [so low], it was these spot-fixes to which I referred. These fixes don't (often) fully remediate that particular vulnerability (even in that narrow locale) let alone improve the application's overall security posture. But, I'd say they're the most common response to an assessment finding. Like McGraw says regarding the BSIMM data, "we observed monkeys eat bananas; is that good? I don't know."

Having considered an attacker's capabilities and motivation, and the value of what you're protecting… an organization might in fact choose to engage in such cat-and-mouse play as the trivial fixes above imply, engineer a real fix, employ some combination of both, or off-load risk in an entirely different way.

Just like before, however, it’s unlikely the vulnerability was created yesterday, so the software *has been* vulnerable for some time. When risk analysis has occurred, and the full cost of a complete solution is resolved, complete with its own residual risk calculation, it’s very likely that a practical organization will tier its response, iterating in time.

Thanks for the post, Ben.
-jOHN

(*1) – Within a Java EE app found through tool-assisted code review with no preference for having been reviewed prior or not (be it prior code review or penetration test). Critical and High in these cases varied somewhat, but always having been seen by the client as exploitable and accepted for remediation by development.

(*2) – This is somewhat glib. More accurately, the organization has increased its visibility into what vulnerabilities its software possesses. It can not say these are the only vulnerabilities the software possesses, and, using purely code review, it can not always say with 100% confidence that detected vulnerabilities are exploitable; some statically determined findings are obviously exploitable whereas others will require dynamic verification.

(*3) – This normalization is even more important if the assessment was delivered by a 3rd party tool or assessor conducting their first run/engagement with an organization. Internal assessment teams often have it easier; they should know/conform to the business unit’s risk management measurement memes.

(*4) – Unless they have a really terrible vulnerability management process or a shady security vendor ;-)

(*5) – Indeed, I **have** suggested use of a Top N list. Here, I’m using it to indicate a priority implicit in industry opinion. Better to augment that clause with [the most common types of attacks observed in our production environment and the #1 emerging attack our security research team advises us on].

This entry was posted on Wednesday, May 20th, 2009 at 3:13 pm and is filed under Software Security. You can follow any responses to this entry through the RSS 2.0 feed. You can leave a response, or trackback from your own site.

Previous post: Twitter Security
Next post: New Security Reads

One Response to “Security and ‘time’”

1. *jOHN* Says:
   May 20th, 2009 at 6:04 pm

   1,000 apologies to BEN who I referred to as Tom in my entry’s outro. This is why you don’t blog on cold medication after a rough day. Textual fix in-bound…

   -jOHN

Leave a Reply

	Name (required)
	Mail (will not be published) (required)
	Website

Submit Comment

Resources

- Overview
- Silver Bullet Podcast
- Reality Check Podcast
- Justice League Blog
- Books
- Java Security Rulepack
- BSIMM
- Gaming
- Publications
- Security Articles
- White Papers

Join Our Mailing List

Email Address:

Subscribe

Contact Careers Account Privacy