Building and Breaking the Browser

How to measure security

Window Snyder – Mozilla Corporation

Mozilla creates the Firefox web browser together with a community of thousands of volunteers. Everything we do at Mozilla supports our mission to promote choice and innovation on the Internet. This includes our efforts to make Firefox as secure as possible.

Most traditional vendors do not share the details of what they are doing to make their products more secure. If users only notice security when it fails, it is hard to justify talking openly about security. If a vendor ships a patch that fixes eight bugs and the headlines read “Product has eight vulnerabilities!” then this vendor is not likely to want to add to the negative story by contributing more information. This is unfortunate because more information helps the security community recognize when the vendor is doing reasonable things to protect users.

All software has bugs. When evaluating which vendor is doing more to secure its products, it is more productive to consider in this scenario that this vendor *fixed* eight vulnerabilities, not just that they had them in the first place.

When talking about how a vendor is making the products more secure, it is not enough to say that this new version has increased security. In order to evaluate whether the product is more secure we need to know exactly what the vendor did to increase security.

Fixing bugs is not enough. Adding security features is not enough. We need to know whether the product was designed and built with security as an integral part of each step of the development process. We need meaningful metrics to measure progress. If there are no metrics released by the vendor, it is very difficult to judge whether those products are doing better over time. There are a few metrics we can see from the outside.

Counting Bugs

The first of these is the number of vulnerabilities. There are many problems with relying on this as a measure of security. The number of vulnerabilities identified is a factor of many things, including not just how many vulnerabilities are present, but also how many people are looking, how much time they spend, and how good they are at looking. The presence of more vulnerabilities may indicate that someone is looking really hard or has superior bug finding skills.

More importantly, the number of vulnerabilities we can identify from the outside is not the whole picture. Most software vendors ship security updates for security vulnerabilities that are reported externally. Vulnerabilities that are found internally by QA or contractors and consultants hired to do security analysis are usually shipped in major releases or in service packs. For most vendors this makes some sense since these fixes will get the benefit of a longer test pass required for a major release or a service pack. But it also means that the number of vulnerabilities fixed regularly in security updates are a small percentage of the total number of vulnerabilities fixed.

Additionally, not all vulnerabilities are fixed. There is always a trade-off between fixing vulnerabilities and shipping on schedule. In most environments the same people required to fix security bugs are the same people that would otherwise be working on new features. There is also a significant cost associated with regressions. In some environments regression rates for security bugs can be as high as 25 percent. For these reasons, most development environments set a bar for bugs that will be fixed and bugs that will not be fixed. Those bugs that are not fixed are not necessarily moved to the next release. These are bugs that do not meet a criticality threshold that justifies the cost of fixing them. The resolution is to tolerate the risk of these security vulnerabilities and leave them unpatched.

At Mozilla, the security updates we ship are comprised of fixes for security issues found externally and through our internal testing. We are constantly testing our software and improving our analysis tools to get better at identifying security issues so that we can fix them. We do not wait for major updates to get security fixes to users. Security updates are our vehicle to continuously make the security work we are doing available to users.

If we tried to compare the number of vulnerabilities identified in Mozilla products with the number of vulnerabilities identified in other products we would never get an accurate comparison. The whole world can see all the moving parts in the Mozilla security processes. The vulnerabilities fixed in a Mozilla security update include both internal and externally found vulnerabilities. For most vendors, the security updates contain only externally identified vulnerabilities and from the outside, we cannot see the whole picture.

Over time we learn that a security bug that today might warrant a low severity rating would get a higher severity rating in the future when new information is identified. There was a time when we believed crashes from heap overflows were not exploitable and that memory retrieval vulnerabilities were low risk. We later learned that both can result in serious security issues. We do not dismiss bugs with low severity for this reason. Security bugs with the highest severity rating are fixed first, but Mozilla fixes all security bugs with any level of security risk. Often, fixing lower severity security bugs (like some denial-of-service issues) ends up improving performance and reliability.

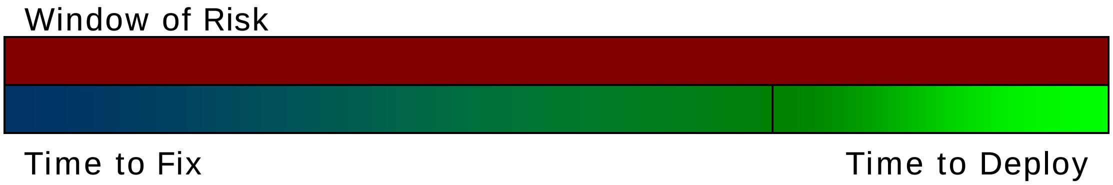
Development environments should evaluate whether this kind of policy is feasible for their operations and consider how to lower the bar for security issues and increase the number of security issues fixed.

Window of Risk

A better measure of overall risk to users is evaluating how long it takes for a user to get a security patch. This window is made up of two parts.

The first is the time it takes for a vendor to create a patch, or the Time to Fix. This includes the time to investigate a security issue, develop and test a fix, and finally ship the update. This is a better measure for understanding how safe a user is going to be than simply counting bugs.

The second is Time to Deploy. This is how long it takes for users to get a patch installed once the fix is available from the vendor.



Time to Fix

Time to Fix can be minimized by improving processes around response and investigation. It is more difficult to improve on the time it takes to develop a fix. Mozilla has been able to significantly reduce the time it takes to test patches because of the help of over ten thousand people from the Mozilla community who download and test nightly builds. These people run pre-release software on their machines with over ten thousand different host configurations, including installed drivers, applications, add-ons, and use these extremely varied systems to visit more different websites than we could possibly ever emulate through automated testing. This incredible breadth allows us to get a lot of testing done in a very short time. Reducing the time it takes to test a security update means our users get a fix sooner.

Time to Deploy

Time to Deploy may be minimized through mechanisms such as the automated update feature in Firefox. Our software update feature frees users from the burden of checking for security updates and automatically notifies them when an update is ready to download. Mozilla has been able to further reduce the Time to Deploy security updates for Firefox through infrastructure improvements and through the support of partner organizations that host mirrors. These improvements enable more users to get security patches faster and make downloads of large updates more reliable.

Reducing the amount of time it takes to deliver patches reduces the amount of time users are vulnerable to attack. If the limiting factor for this metric is a physical resource like bandwidth or servers, it is important to realize this so that the cost of securing users more quickly can be accurately weighed against the financial cost of more resources.

Internal Metrics

A software vendor can assess for themselves whether its processes are working and whether the product is becoming more secure by looking at a few other metrics.

Find Rate

The find rate, or how many bugs are found over a period of time, can be used to evaluate whether the team is getting better at identifying security issues. Improvement may be due to added resources on the product security team, more effective tools, a greater amount of person hours devoted to investigation or other factors.

In some scenarios, improvement in the find rate may also be due to an increase in the number of vulnerabilities present. This may be the case if a number of developers who are poorly trained in security best practices have recently joined the development team through acquisition or otherwise. This may also be the case if a third-party technology or library was recently incorporated into the product and has a lot of vulnerabilities.

Generally speaking, the find rate goes up as the environment becomes more effective at finding vulnerabilities. Eventually, as developers are trained in security best practices and existing vulnerabilities are fixed the find rate will begin to decrease. This indicates that there are fewer of the kinds of vulnerabilities that the development team knows how to identify present in the product. It is important to continuously train the team on new categories of vulnerability and methods for identifying them to ensure that as the easier bugs are eliminated from the product that the testers are able to move up to the more sophisticated and subtle bugs that are more difficult to find.

Fix Rate

The fix rate, or the number of bugs fixed over a period of time, helps a software vendor evaluate whether the current commitment of resources is sufficient. If the fix rate is low and the find rate is high the software vendor will soon have a large and growing backlog of security bugs waiting for fixes. This indicates an insufficient allocation of resources to bug fixing. This problem could also be resolved by reducing the find rate, or telling people to stop looking, but that will not make the product more secure.

The fix rate may go up if the development team allocates more time to fixing bugs instead of, for example, working on new features. It may also go up if the team introduces architectural changes that address many vulnerabilities at once or potentially eliminate an entire class of vulnerabilities. It will also go up as the team sees similar bugs repeatedly and is able to replicate earlier work to address the new problem reducing the amount of time spent on each individual issue.

Tracking the fix rate helps a development environment evaluate whether developer time is being spent on tasks that are accurately aligned with security goals or if more headcount is justified to achieve these goals. It also acknowledges improvements in efficiency and design changes that greatly impact the security of the product.

Severity

Tracking the severity of security issues over time indicates what sorts of bugs are being identified and helps answer the inevitable management question of “How bad is it?”

If over time the percentage of critical bugs is decreasing, this may indicate that the worst of the worst have been identified and now the majority of what remains are lower severity bugs. It may also indicate that the remaining critical bugs are more subtle and more difficult to find. Both of these indicate an improvement and would give weight to what for most vendors is externally expressed as simply “improved security.”

An open and transparent vendor approach to security enables the security community to evaluate whether the vendor is serious about security. If a vendor is doing great work in security, sharing this information is the best way to let the world know.