June 2012

V e c t o r

Perspectives on what's happening in the world of Information Security

Hiren Shah
President, Net-Square
hiren@net-square.com

Secure • Automate • Innovate

Net-Square Solutions

Is a niche Application and Network Security Service provider. Net-Square provides Consulting Services like Vulnerability Assessment, Penetration Testing, Code Review, Reverse Engineering and Security Architecture Consulting

Net-Square also offers Products like Server Defender Vulnerability Protection (SDVP), a web application Firewall for IIS applications and NS Webscan, an automated application vulnerability scanner

Last but not the least is the training programs. Net-Square offers a variety of customizable training programs for the benefit of end users and developers.

~~Breaking~~ Hacking News:

In what appears to be a series of hacking incidents, around 80 Bank account holders of State Bank of India and State Bank of Patiala near Yamunanagar, were robbed of Rs.20 lakhs.

The most recent incident comes from the main branch of State Bank of Patiala where hackers withdrew an amount of Rs.10.10 lakh from accounts of 23 customers. Police investigations are on.

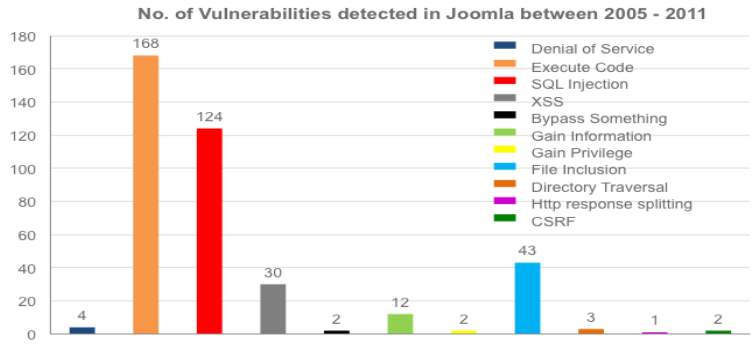
Source: The Times of India, 17th June 2012

To "Open Source" or "Not to Open Source"

In the IT World, the strategy "To Open Source or Not to Open Source" is a perennial debate. When I was in US last year, I came across many large Global Financial Institutions who are adopting Open Source as a strategy to implement all future solutions. Adoption of Open Source technology is a good strategy, especially in the complex licensing regimes practiced by many large software vendors. While security is an issue that bears upon the decision to go for it, not many fully understand how to take care of them when operationalizing the "Open Source Stack" strategy.

In recent times we have been called into test many applications, which are based on open source applications or a complete stack. Testing these applications have provided us some valuable insights to be considered while going the Open Source way.Before I discuss this, let me highlight that very rarely is an open source product used as-is. In most instances, the product undergoes heavy customization, including installation of many extensions. In light of this, our tests revealed two very important insights.

One, that many open source products have add-ons, extensions, plug-ins etc. which make them attractive in many ways. While the core application itself is mostly secure, it is these extensions and plug-ins contributed by many diverse developers and organizations that introduce vulnerabilities into the open source product as a whole. The graph below shows the number of vulnerabilities introduced in Joomla, a very popular open source CMS, between 2005 and 2011.



While the graph may shock you, it is actually not surprising since Joomla has more than 1700 extensions and add-on modules. While many of them may be fixed, what we recommend is to only select those that do not have any known vulnerabilities.

Two, all our tests have revealed that the customizations done during the implementation have ***always*** introduced new vulnerabilities. So expecting that there will be less number of vulnerabilities simply because there is limited coding due to customizations is a fallacy.

Conclusion: Conducting a thorough Vulnerability Assessment and Source Code Review is even more vital when implementing open source products to cover your bases against any vulnerability introduced or already present but unknown. But this should not deter you from taking a strategic call on adoption of open source technologies. With the right security partner, you should be able to get the strategic advantages of Open Source, whether that be cost savings or risk mitigation! Until next time, stay safe! - Hiren

Follow Hiren's views on Twitter @hiren_sh or on his blog http://1-thought.blogspot.in

Heavily Dependent on Automated Scanners? You may be missing out on vital clues!

A variety of automated web application scanning tools are available today, which can perform a vulnerability analysis of web applications really quickly and give you a list of vulnerabilities detected. This certainly reduces a lot of time for your Infosec team, which is already loaded with security issues. They can now certify and approve deployment of applications to production after getting a clean report. But, how good are these automated tools? Can we really rely on them?

Three Security experts, Adam Doupe, Marco Cova, and Giovanni Vigna of the University of California, Santa Barbara, put the best of automated and semi-automated scanners to test and have concluded that “while certain kinds of vulnerabilities are well-established and seem to work reliably, there are whole classes of vulnerabilities that are not well-understood and cannot be detected by the state-of-the-art scanners”.

Their study, aptly titled *"Why Johnny can't pentest"*, demonstrates how very well regarded automated scanners miss as many as 60% of the findings! And this is really not surprising. Even the best of scanners are limited in application "coverage". What you don't see, you don't report. Net-Square has a history in building some of the best automated scanners in the past, and we are well aware of problems with automated scanners. The second fundamental problem is that automated scanners can never perform vulnerability chaining. Critical findings discovered by Net-Square analysts sometimes take more than two or three bugs to exploit.

The other problem is an operational one. Customers and firms undertaking application testing are so focused on reducing the number of false positives that they weaken the actual premise of testing. When I talked to a client about Net-Square's automated scanner – NS-Webscan, the first question was about its rate of false positives! But wait, aren't we supposed to care about what it *finds* in the first place? Many firms performing application testing use automated scanners to generate the reports and have their analysts to filter out the false positives. How did we find this? While interviewing candidates from these firms.

Automated scanners aren't entirely useless. They are best utilized in reducing the load for manual testing. Obvious vulnerabilities get detected right away. Not all customers can engage a team of sharp penetration testers throughout the year. Automated scanners, like NS-Webscan, provide an intermediate solution for Infosec Management to certify rollout of minor releases and changes between two cycles of manual penetration testing. However, being entirely dependent upon automated testing is like an ostrich sticking its head into the sand.

We use NS-Webscan to initially check to see how vulnerable the application is. If we find many issues in the first round of testing then our analysts know that they have a long road ahead on that particular application test, with many interesting vulnerabilities to be discovered! The bottom line is, no amount of automation can match the skill and cunning of a hacker's brain!

- Hardik Kothari,
Business Development Manager
Net-Square Solutions

Denial of Service by Regular Expressions - ReDOS!

A Regular Expression or "regex" is heavily used in pattern matching. Applications use regular expressions to verify inputs of all sorts – dates, zip codes, email addresses, transaction amounts, etc. Little is known about the dangers in poor implementation of regular expressions. Regex search patterns, if not implemented properly, can cause havoc. That havoc is named ReDOS!

The intrinsic nature of regular expressions involves finding the best and tightest match. This involves recursion, a nested operation that can be time consuming if not handled properly.

Example: The regex "^(\d+)+$" matches any groups of digits. An input of "1234" matches the regex on the first pass itself. An input of "123X" will fail on the first pass. Since the regex involves grouping, the engine will try another pass to see if a different grouping works. Before it declares the input to be non-matching, the regex engine would have tried 2^4 = 16 passes. An input of "123456789X" will require 2^10 i.e. 1024 passes before it fails the matching. A single character added to the input *doubles the number of passes* for checking! In our tests, evaluating a 10 character mismatched input took 0.001 seconds, whereas a 25 character mismatched input took 15 seconds!

Adding more characters in the evaluation string increases the number of paths evaluated exponentially and forces the regex engine to evaluate millions of paths, eating up CPU time and leading to ReDOS!

At the heart of a regex engine is a Finite Automata machine. Some regex engines use NFA Usually when matching search patterns, either an NFA – nondeterministic finite automata – which involve backtracking when it comes to mismatches or a DFA engine is used. Backtracking matches the positive input fairly quickly, it is the negative input which takes a bit longer. The NFA must confirm that none of the possible paths through the input string match the regex, which means that all paths have to be tested.

ReDOS is more common than you think it to be. Constructing a regex is a complex task, and if it isn't thought through properly, you have a ReDOS situation in your application.

More on ReDoS can be found at
https://www.owasp.org/index.php/Regular_expression_Denial_of_Service_-_ReDoS

- Mayur Singru, Net-Square Solutions