Beyond Performance Testing

by:
R. Scott Barber

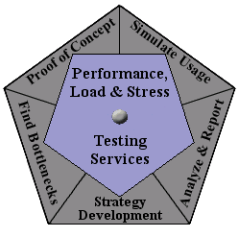
Part 14: Testing and Tuning Security

This final installment of the "Beyond Performance Testing" series is also the last article on the theme I call "the performance testing and tuning team." Here I'll discuss testing and tuning system security features. Over the last few years, security has become a major focus of nearly every application, but security features degrade the overall performance of most applications more than any other factor. In short, increasing the security of an application can simply devastate performance.

This article aims to make you aware of which security features may be responsible for your site's poor performance and to help you determine conclusively whether security features are indeed degrading performance. It describes how to test security features with the intent to tune and how you can provide the information that stakeholders need in order to make informed decisions. What it doesn't cover is how to use Rational tools to test the effectiveness of security features. For information about that, see the Rational User Conference 2002 presentation by Chris Walters and me entitled "Security Testing: Step by Step System Audit with Rational Tools," on my Web site under Presentations.

So far, this is what we've covered in this series:

This article is intended for mid- to senior-level performance testers and members of the development team who work closely with performance test engineers. You should have read at least Parts 9 and 11 before reading this article; it would be helpful to have read Parts 5, 6, 7, 8 and Part 10 as well.



Security and Performance

It's common knowledge that the more secure you make an application, the more that application's performance degrades unless performance concerns are actively addressed. I've run across dozens of quotes that acknowledge the trade-off between security and performance in applications. Here are just a couple:

- "The need for security is obvious, but too often comprehensive security degrades network performance to unacceptable levels." — Mark Tharby, vice president, Metro Networks, Nortel Networks
- "Secure Sockets Layer provides a secure way to exchange information between clients and servers. However, the CPU has to perform intensive cryptography, which degrades performance." — Microsoft TechNet IIS 6.0 Features

SSL has become so common that sometimes stakeholders fail to realize just how much performance degradation can be directly attributed to "simply" enabling SSL on an e-commerce Web site. A study by the Courant Institute of Mathematical Science at New York University entitled "A Comparison of HTTP and HTTPS Performance" shows that just turning on SSL can degrade system performance by as much as 22%. The authors of yet another academic paper, "Security versus Performance Tradeoffs in RPC Implementations for Safe Language Systems," summarize by saying essentially that designing for performance and security are risk-based trade-offs that must be taken into consideration at design time: "[Testing various configurations] generates a spectrum of design points that can help application programmers to decide the level of security and performance that is more suitable."

Performance tuning highly secure applications normally involves either reducing security (increasing risk) or significantly increasing network and/or hardware resources (increasing expense). In this struggle among security, cost, and performance, performance often loses. The best we can do in these cases is to help the stakeholders make informed decisions.

Testing Security

Testing security features for performance is mostly an exercise in what's commonly known as configuration testing. In configuration testing, you execute the same test against several different configurations of the system. Most people think of "configurations" as referring to hardware (for example, Web servers, CPU's, or RAM), but it can also refer to security features (for example, whether SSL is enabled or disabled or just used for login; whether session tracking is dynamic or static). These tests serve several purposes. They can be used to determine which tested configuration delivers the best performance, to provide input to an ROI analysis, or even to determine if the factor being varied has any noticeable effect on the system at all.

It would be impossible for me to list all the security features your application may have that you may want to test, so instead I'll outline an approach to testing security that applies to a common scenario — testing a site with SSL enabled and disabled. The same testing principles apply pretty well to the majority of other security features your site may employ. In the example scenario, the stakeholders have decided that security is of the utmost importance, so the developers have SSL enabled the entire site. When performance testing starts, it reveals that the site meets none of the performance goals, so

naturally, the developers want to know what the bottleneck is. My suggested approach in cases like this is as follows:

- Disable SSL for the entire site.
- Test and tune until all performance requirements are met or exceeded.
- Enable SSL for one feature (for example, login).
- Test the site with the same workload as with SSL disabled.
- Compare response times and resource utilization.
- Tune where necessary, if possible.
- Repeat steps 3–6 for each feature until the site is as secure as the stakeholders desire.
- After all tuning is complete, reexecute the test at all stages of SSL enablement. Record and report on the response times and resource utilization (particularly CPU and memory) of each relevant tier.

I realize that sounds like a lot of work. Unfortunately, even that's a bit oversimplified. You can't just play back your SSL-disabled scripts against an SSL-enabled site. To use this method you'll have to choose from one of four options:

- Record new scripts for each test.
- Manually modify the sections of your scripts that change due to the SSL changes from test to test.
- Manually parameterize the URL, port, and `http_request` flags (that is, `HTTP_CONN_DIRECT` and `HTTP_CONN_SECURE`) so that you can make these adjustments in one place using variables, rather than having to change them by hand globally each time.
- Use the custom VuC utility and function `URLParse`, which will parameterize these components for you.

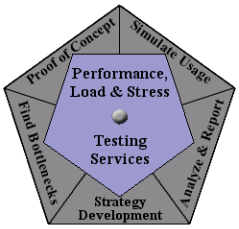
It goes without saying that I recommend the last option.

Once you've completed this testing cycle, or as much of it as you're able to accomplish, you can give stakeholders the information they need in order to make decisions. I typically only get time for SSL disabled, SSL login, and SSL entire site. Still, that's enough to give stakeholders a graph like the one shown in Figure 1. All of the metrics there come from an actual project that I recently did.

SSL Performance Comparison
(75% expected peak load)

	No SSL		SSL Login Only		SSL Whole Site	
Page	Response Time	Web Server	Response Time	Web Server	Response Time	Web Server
home page	2.57	CPU =	2.63	CPU =	4.33	CPU =
login	3.58	52%	5.62	63%	7.71	91%
search	4.47	RAM =	4.58	RAM =	6.11	RAM =
order	5.21	45%	5.33	61%	6.89	84%

Figure 1: Response times and resource utilization figures for three different security configurations



As you can see, this testing gave the stakeholders enough information to make decisions. It also allowed the testing and tuning team to realistically determine what portion of the performance metrics was attributable to security-related features (in this case, SSL), which helped them immensely with the tuning effort.

Tuning Security

Tuning security features for improved performance seems to be a five-step process:

- Tune the system as well as possible before applying security features.
- Retune the system after the security features have been implemented.
- If better performance is required and budget allows, add hardware.
- If better performance is required and hardware can't be added, limit, remove, or try different ways to implement security features.
- Either accept the resulting performance/security trade-offs or abandon the project.

As performance testers, our role in this process is to provide as much information as possible to (1) the developers, architects, and administrators, to help them tune or redesign the system, and (2) the stakeholders, to help them decide which security features stay, which go, and what price they're willing to pay for improved performance. All of the principles of collaboration that we've discussed in previous articles apply here as well. This time we've just added a few extra steps to the strategy.

Summing It Up

This article has barely scratched the surface of the topic of security, but I hope it's given you some insight into how security and performance are almost always inversely related and taught you a strategy to conclusively determine when security features are the cause of poor performance. By their very nature, security features degrade performance, often significantly. When those features degrade performance too far, what we commonly refer to as tuning isn't enough. In those cases the team must either redesign some or all of the system, add hardware, limit the security features, or accept degraded performance. Our job is to collect and present the data required for the appropriate members of the team to make those decisions.

Wrapping It Up and Looking Ahead

Over the course of 27+ articles (the "User Experience, Not Metrics" series, the "Beyond Performance Testing" series, and several other separate articles), we've explored some of the key areas of performance testing. We've looked at how to solve complex problems using the Rational tools, we've examined a performance engineering methodology and various ways to document performance data, and we've even investigated ways to build more effective performance testing and tuning teams. Never once did we lose sight of our ultimate goal, improving the end user's experience. Believe it or not, there's still a lot more to be said about performance engineering, but I hope these articles have helped you in some small way and piqued your interest in this quickly maturing area of software development.

It's been a pleasure talking to all of you I've met at conferences, exchanged e-mails with, or run into on the forums here at RDN or elsewhere. I've gained many insights from these conversations, improved

myself as a performance engineer, and even made what I honestly believe will be lifelong friends.

As you may already be aware, a performance testing community is starting to form. Top performance testers and engineers from around the world are starting to have regular conversations with one another, and a biannual workshop (the Workshop On Performance and Reliability) has been created to bring these people together. You can expect much more to come in this field in the near future. Keep an eye out for more articles, and maybe even books, in the coming years (and I don't mean all by me). A lot of energy is being directed toward bringing performance engineering up to the maturity level of other quality assurance fields such as systems testing. I hope you feel that energy, and I encourage you to become a part of the maturation process.

As I complete this article, I don't plan to write another series for RDN. I do have some plans for more articles to be written jointly with other noted software testing experts and performance testers. I'll continue to make myself available through forums, conferences, and my Web site. I'm sure I'll also periodically have a new article for RDN as well.

Thank you for reading these articles. I hope you've enjoyed reading them as much as I've enjoyed writing them.

References

- A Comparison of HTTP and HTTPS Performance by Arthur Goldberg, Robert Buff, and Andrew Schmitt (Computer Science Department, New York University, no date)
- "Security Testing: Step by Step System Audit with Rational Tools" by Chris Walters and Scott Barber (Rational User Conference 2002 presentation)
- Security versus Performance Tradeoffs in RPC Implementations for Safe Language Systems by Chi-Chao Chang, Grzegorz Czajkowski, Chris Hawblitzel, Deyu Hu, and Thorsten von Eicken (Department of Computer Science, Cornell University, no date)

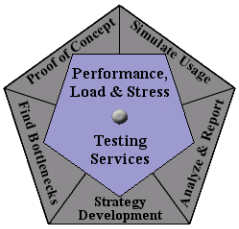
Acknowledgments

Extra special thanks go to Ron Stanley, Northeast Area Director, Noblestar; Rebecca Bence, Senior Editor, Rational Developer Network; Brian Bryson, technical editor for this series; and Lorraine Anderson, the copy editor for both this and the "User Experience, Not Metrics" series.

In 2001, Ron came to my office and asked me a simple question: "Hey Scott, Rational has this new program where it's looking for articles to be contributed by partners. You seem to have a lot to say about performance testing. Why don't you try writing?" That conversation is what started me writing. Before that day, I had never had a single word published. Thanks, Ron, for planting the seed.

After Ron planted the seed, I took my ideas to Rebecca. She made both of these series possible. Based on a few e-mails and a one-paragraph summary of the proposed articles, she encouraged me to take on the 13-part "User Experience, Not Metrics" series knowing that I had never written an article before. Thanks, Rebecca, for having faith in me before you had any reason to.

Brian wears several hats at IBM Rational; the one I want to thank him for is doing the technical editing of my articles. Brian was busy enough without my articles, but he never let up on his quality reviews.



He always found the spots I glossed over and suggested great ways to improve the content. Bri, thanks for taking the time to read these closely and never letting me down.

Lorraine has edited every article I've published to date. She continually amazes me with how she can take some of my barely formed thoughts and change just a few words or reorder a few paragraphs to turn them into these articles. She tells me that it's "no big deal," but I think it is. I've learned more about writing from her than I did from any of my English teachers in school. Thanks, Lorraine, for making my thoughts easy to read — I've enjoyed every moment of working with you.

One last time I'd like to thank (in alphabetical order) the other folks who contributed in some way to these two series with content, ideas, peer reviews, or original works that I derived from: James Bach, Ross Collard, Cem Kaner, Alberto Savoia, Roland Stens, Joe Strazzerre, Steve Tani, Chris Walters, Nathan White, and all the members of the discussion forums here at RDN and on QAForums.

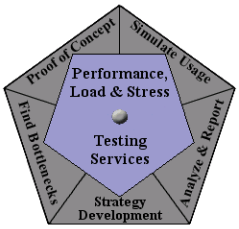
Thanks, of course, go to the readers. As a result of the interest you've shown in these articles, I've had numerous great opportunities. I've gotten to give presentations on these topics in many venues and locations, including the Pacific Northwest Software Quality Conference, a seminar in London, and training in Dubai, to name a few. I've appreciated all of the e-mails and conversations that I've had with you, and I hope our contacts continue as I make myself available in other media.

Last, but certainly not least, I want to thank my wife, Shannen, and my boys, Nicholas and Taylor. They've been very tolerant of me spending a lot of time in front of the computer. They've been supportive when I was working too much, and now it's time for me to keep my promise to them of "no more major writing projects for a while." Thanks, guys. Now go get your stuff so we can go to the beach!

And as with the previous parts of the Beyond Performance Testing series, the original version of this article was written on commission for IBM Rational and can be found on the IBM DeveloperWorks web site

About the Author

Scott Barber is the CTO of PerfTestPlus (www.PerfTestPlus.com) and Co-Founder of the Workshop on Performance and Reliability (WOPR – www.performance-workshop.org). Scott's particular specialties are testing and analyzing performance for complex systems, developing customized testing methodologies, testing embedded systems, testing biometric identification and security systems, group facilitation and authoring instructional or educational materials. In recognition of his standing as a thought leading performance tester, Scott was invited to be a monthly columnist for Software Test and Performance Magazine in addition to his regular contributions to this and other top software testing print and on-line publications, is regularly invited to participate in industry advancing professional workshops and to present at a wide variety of software development and testing venues. His presentations are well received by industry and academic conferences, college classes, local user groups and individual corporations. Scott is active in his personal mission of improving the state of performance testing across the industry by collaborating with other industry authors, thought leaders and expert practitioners as well as volunteering his time to establish and grow industry organizations. His tireless dedication to the advancement of software testing in general and specifically performance testing is often referred to as a hobby in addition to a job due to the enjoyment he gains from his efforts.



About PerfTestPlus

PerfTestPlus was founded on the concept of making software testing industry expertise and thought-leadership available to organizations, large and small, who want to push their testing beyond "state-of-the-practice" to "state-of-the-art." Our founders are dedicated to delivering expert level software-testing-related services in a manner that is both ethical and cost-effective. PerfTestPlus enables individual experts to deliver expert-level services to clients who value true expertise. Rather than trying to find individuals to fit some pre-determined expertise or service offering, PerfTestPlus builds its services around the expertise of its employees. What this means to you is that when you hire an analyst, trainer, mentor or consultant through PerfTestPlus, what you get is someone who is passionate about what you have hired them to do, someone who considers that task to be their specialty, someone who is willing to stake their personal reputation on the quality of their work - not just the reputation of a distant and "faceless" company.