



Myth-Busting Web Application Buffer Overflows

By Jeremiah Grossman
Founder and CTO, WhiteHat Security, Inc.
www.whitehatsec.com

Contrary to popular belief, buffer overflow¹ exploits do not occur in custom web applications. While technically possible, the truth is that they are just not seen in the real world. Our experience at [WhiteHat Security](http://WhiteHatSecurity.com), having assessed hundreds of websites and identified thousands of vulnerabilities, shows that statistically, buffer overflows appear near the bottom of the list of total discovered issues. Conversely, the Open Web Application Security Project² (OWASP) has stated that buffer overflows are one of the Top-10³ most critical web application security flaws. And, it is true that buffer overflows are a primary vehicle used to propagate the most notorious viruses and worms in operating systems from Microsoft Windows to Linux. So if both observations are true, then where's the disconnect?

The disconnect comes from a common misunderstanding of where, when, and how buffer overflows are exploited. A buffer overflow is a software condition in which an application attempts to store more data in a memory buffer than has been allocated. The overflow will overwrite adjacent memory, which could cause the application to crash. Cleverly designed buffer overflow exploits are able to take advantage of this situation by overwriting specific pieces² of memory to execute arbitrary system commands and take over computers. The nature and origin of buffer overflows has been covered exhaustively in numerous articles and white papers⁴⁵. And, we've all seen the result of well-executed buffer overflow exploits on corporate networks.

Normally, buffer overflow vulnerabilities are found through source code analysis, or by reverse engineering application binaries. Depending on the instance, a person could spend hours, weeks, or even months stringing together memory addresses in order to create functional exploit code. For hackers to uncover these issues, they really need to get their hands on the running application. The hacker can then repeatedly exploit the buffer overflow, capture the crashed application state (via SoftIce or Gnu Debugger) and trace the buffer through



memory. This is why buffer overflows usually appear within compiled commercial and open-source software, rather than in custom web application code.

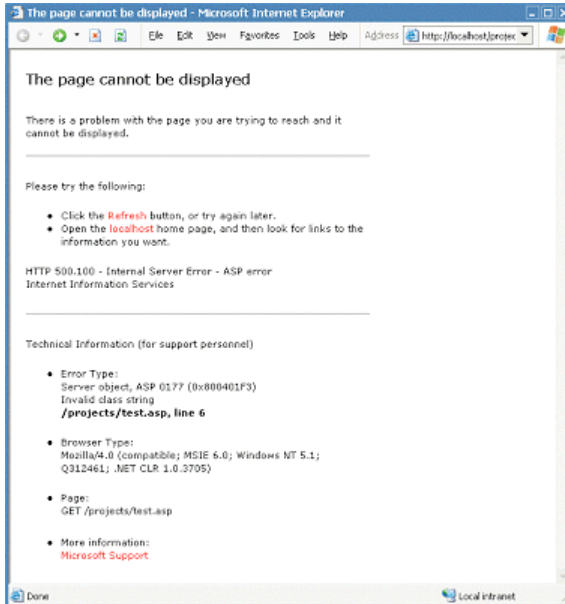
Custom web applications are exactly that — custom software that an organization develops for itself and hosts on its web servers. We're not talking about commercial Web software components from IBM, Microsoft, or SAP. This means that the public, including hackers, is unlikely to have access to the application source code or essential elements for testing. As a result, the way buffer overflow vulnerabilities are identified and possibly exploited in custom web applications is a completely different process. With the exception of a source code review, testing a web application for buffer overflow⁶ vulnerabilities must be performed remotely over the network.

Penetration-testers, as well as hackers, do not have access to source code, application binaries, or memory dumps. Testing is blind. A typical test will input several thousand characters into a URL query string parameter ([Example 1.](#)) and wait for the web server to respond with a "500" error response code ([Example 2.](#)). A 500 code may be an indication that an application has crashed server-side due to a buffer overflow, but this is extremely unlikely. It is more often the case that a device in the connection chain, including web application firewalls, http proxies, load balancers, web servers, application servers, or others, caught the abnormal test and returned an identical 500 error message. This is why blind buffer overflow tests are prone to heavy false-positive rates. The fact is that anything could have caused a 500 error, but only rarely is it a web application buffer overflow.

Example 1.

```
http://webserver/foo.cgi?param1=AAAAAAAAAAAAAAAA...5000 A's
```

Example 2.



However, if a buffer overflow did exist, it would likely be in common off-the-shelf components used to build the website, such as Microsoft IIS, Apache, Oracle, MySQL, SiteMinder, and thousands of others.

In order to further understand why buffer overflow exploits do not normally occur in custom web applications, consider this hypothetical, and rare, situation: a hacker wants to perform a buffer overflow exploit and nothing is restricting user-supplied input to a custom web application. What needs to take place for the attack to succeed?

In the simplest of circumstances, the hacker would have to find a vulnerable web application and parameter, discover the size of the buffer, properly overwrite the stack pointer, and load in well-written shell code⁷. And, all of this must be done while completely blind to any application or CPU architecture specifics. Call me jealous, but you'd have to be one of those Hollywood movie hackers⁸ to pull this off. It would be like swishing a basketball blindfolded while standing outside in the stadium parking lot. Perhaps this is why the Web Application Security Consortium (www.webappsec.org) has yet to identify a single media reported incident⁹ where a website was hacked due to a buffer overflow within a custom web application.

There is another critical aspect to consider when evaluating the risks of buffer overflows in custom web applications. Popular web application programming languages¹⁰, such as Java, are immune to buffer overflows by their very nature. These languages perform their own memory handling and unless there is some problem with the interpreter (rare), you're safe. So if someone alerts you to a buffer overflow in your web application, you have every reason to be skeptical, especially if you are using one of these languages. Let me be careful here to say that I'm not suggesting we have carte blanche to leave our code riddled with memory handling issues. In fact, if you're using a C/C++/C# language or variant, and you do happen to be vulnerable, then maybe there is some risk worth investigating.

So the real source of the disconnect between real-world findings and the OWASP top ten is risk evaluation. We agree that IF someone managed to exploit a buffer overflow in a web application, then it would result in a critical situation. However, businesses need to prioritize their risk based on the likelihood of occurrence and focus on those vulnerabilities/issues most likely to be found and exploited by hackers. The question is, what exactly are they?

My advice is that to get the most bang-for-your-security-buck take care of your Cross-Site Scripting (XSS) and SQL Injection vulnerabilities, Authentication/Authorization loopholes, and business logic flaws. These are the vulnerabilities worth going after first because they significantly reduce risk. To defend against these threats, assess early and assess often to protect sensitive data and stay out of the headlines.

About the Author

Jeremiah Grossman is the founder and Chief Technology Officer of WhiteHat Security (<http://www.whitehatsec.com>), where he is responsible for web application security R&D and industry evangelism. As an industry veteran and well-known security expert, Mr. Grossman is a frequent international conference speaker at the BlackHat Briefings, ISSA, ISACA, NASA, and many other industry events. Mr. Grossman's research, writings, and discoveries have been featured in USA Today, VAR Business, NBC, ABC News (AU), ZDNet, eWeek, and BetaNews. Mr. Grossman is also a founder of the Web Application Security Consortium (WASC), as well as a contributing member of the Center for Internet Security Apache Benchmark Group. Prior to WhiteHat, Mr. Grossman was an information security officer at Yahoo!, responsible for performing security reviews on the company's hundreds of websites.

Originally published on [SearchAppSecurity](#), March 2006.



Footnotes

¹ Wikipedia - Buffer overflow

http://en.wikipedia.org/wiki/Buffer_overflow

² The Open Web Application Security Project (OWASP) is dedicated to finding and fighting the causes of insecure software.

<http://www.owasp.org/>

³ The OWASP Top Ten represents a broad consensus about what the most critical web application security flaws are.

<http://www.owasp.org/documentation/topten.html>

⁴ The Tao of Windows Buffer Overflow

http://www.cultdeadcow.com/cDc_files/cDc-351/

⁵ Smashing The Stack For Fun And Profit

<http://www.insecure.org/stf/smashstack.txt>

⁶ Web Security Threat Classification - Buffer Overflow

<http://www.webappsec.org/projects/threat/>

⁷ Wikipedia - Shellcode

<http://en.wikipedia.org/wiki/Shellcode>

⁸ Hackers - The movie

<http://www.imdb.com/title/tt0113243/>

⁹ Real World Web Hacking URL's

A compiled list of news sources, specific to real-world websites, describing an actual web application security hack.

http://www.webappsec.org/documents/real_world_web_hacking.shtml

¹⁰ Cold Fusion, Java, Perl, PHP, Python, Ruby on Rails, and Visual Basic are popular web programming languages that perform their own memory handling.