



WHITE PAPER

Common web application security problems and how to identify them

Part 5 Buffer overflow

By

Lee J Lawson
Lead Penetration Tester, **dns**.

Introduction

This is **part 5** of a series of papers designed to raise the level of security knowledge regarding common security flaws in web applications. All papers revolve around the top 10 list of web application security issues as defined by the Open Web Application Security Project (OWASP). The last issue described the vulnerability known as **Cross site scripting**. This paper concentrates on **Buffer overflows**, the identification and resolution of this flaw.

Here is a list of the most common security flaws at the time of writing taken directly from the OWASP site.

1. **Unvalidated input**
2. **Broken access control**
3. **Broken authentication and session management**
4. **Cross site scripting**
5. **Buffer overflow**
6. **Injection flaws**
7. **Improper error handling**
8. **Insecure Storage**
9. **Application denial of service**
10. **Insecure configuration management**

It should be noted that this paper is not intended to be used as a replacement to professional penetration testing. Rather it is aimed toward raising the level of security knowledge in the community. Professional penetration testing requires years of experience to be able to apply worthwhile interpretation to results from testing tools and gain the skills required to perform manual assurance testing.

Buffer overflow

Web application components in some languages that do not properly validate input can be crashed and, in some cases, used to take control of a process. These components can include CGI, libraries, drivers, and web application server components.

a buffer overflow occurs when too much data is pushed into any given input buffer

The Problem

A buffer overflow occurs when too much data is pushed into any given input buffer. Any input to a program, whether it is a username and password entry form, or a much more complex system of transferring data between processes, should have an allocated area in memory dedicated to hold that input. The area or 'memory address range' should be protected by the code that supplies the data. By this, I mean that if the input buffer and subsequent memory address range is 10 bytes in size, the application that passes the data into it, should check that the input does not exceed 10 bytes.

If this 'boundary checking' is not performed and too much data is pushed into the buffer, the extra data starts to overwrite adjacent memory locations mostly causing a crash, but in some instances it allows an attacker to execute their own code and gain control over the system.

an in-house built system will only be the target to a hacker that has a motive for attacking

Buffer overflow problems occur in all manner of applications, web enabled or not. This might be a third party e-commerce shopping cart application or an in-house built system for retrieving and displaying data. The level of risk associated with each is that a popular e-commerce application will be the target for many hackers; they may even have the source code, this makes it more likely for a flaw to be discovered. An in-house built system will only be the target to a hacker that has a motive for attacking it; this means that this type of flaw is less likely to be discovered in this type of application due to the number of attackers. However, you should be aware that in-house built applications tend to have more security flaws because they go through a lesser QA process.

Identification

To identify if an application that an organisation uses has a buffer overflow problem requires focussed penetration testing.

If the application is a third party system that is used by many other organisations then there is a good chance that when a vulnerability is discovered it will be made publicly available and the vendor will produce a patch to fix the issue.

If the application has been built in-house, then bespoke penetration testing is the only effective and safe manner in which to discover these flaws and rate the risk posed by them.

penetration testing is the only effective and safe manner in which to discover these flaws and rate the risk posed by them

A technique used by hackers and penetration testers alike is to use an automated 'fuzzer'. This is a tool that can automatically supply different types and sizes of input to any application with the intent of causing a buffer overflow leading to a crash. Once a crash has occurred due to the fuzzer's input, then further analysis can be performed to identify what type of data can be inputted into the application and assess the risk presented by that flaw.

Countermeasures

Secure coding practices are probably the best method of preventing buffer overflow vulnerability's. Prevention is always better than cure and programmers / developers should follow secure coding practises during the D3C of an application:

- ⇒ Secure in Design
- ⇒ Secure in Deployment
- ⇒ Secure in Default settings
- ⇒ Secure Communications.

different languages can also help

Different languages can also help. JAVA, .Net and C# are all type safe languages and it is actually very difficult if not impossible to write an application that is vulnerable to buffer overflows in those languages.

This is all a moot point if you are using a third party application. When this is the case, you are limited to keeping up to date with the latest vulnerability's and patching the system. A penetration test will assist you in rating the level of risk that any application poses to the network and may also offer mitigation advice on preventing successful attacks.

The next part of this series concentrates on **Injection flaws** and how they are used to exploit systems.