



WHITE PAPER

# Common web application security problems and how to identify them

## Part 9 Application denial of service

By

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

# Introduction

This is **part 9** 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 **Insecure storage**. This paper concentrates on **Application denial of service**, 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 overflows**
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.

# Application denial of service

*Attackers can consume web application resources to a point where other legitimate users can no longer access or use the application. Attackers can also lock users out of their accounts or even cause the entire application to fail.*

## The Problem

Denial of Service attacks are defined as any attack that attempts to deny legitimate users access to a resource. There is also another level of attack called a Distributed Denial of Service (DDoS) where an attacker would utilise the processor power and network bandwidth of hundreds if not thousands of systems across the globe. These systems are victims of the attack also as they will have been exploited and turned into zombies or robots, making a botnet.

application layer Denial of Service attacks are a different beast to typical DoS attacks

Application layer Denial of Service attacks are a different beast to typical DoS attacks. Typical attacks such as SYN floods, Land and teardrop attacks affect the network layers of a system using up the available bandwidth or disrupting the hosting server. The type of data traffic used in these typical attacks is relatively easy to detect and thwart.

Application layer attacks however, closely resemble normal traffic. The attacks enter the application on the normal ports (80 – HTTP or 443 – HTTPS etc) and mostly comprise regular HTTP requests such as GET or POST. Therefore it is very difficult to determine between a normal user performing some authorised function on the application and an attacker attempting to subvert the code.

An attack that targets the application layers will also be much more efficient in comparison to typical DoS attacks. The attacker will not need thousands of bots to mount their attack. If the application code is susceptible to DoS weaknesses, all the attacker needs is a web browser!

Due to the fact that these attacks will generally use HTTP or HTTPS as their transport protocol, they can be proxied through a chain of anonymous proxies. This makes it extremely difficult to trace the attack and successfully prosecute attackers. The levels of anonymity that can be gained means that attackers have greater confidence that they will go unnoticed.

attackers have greater confidence that they will go unnoticed

Application layer DoS attackers have many attack vectors open to them, such as connection exhaustion, back end systems and even simple attacks like locking out user accounts.

Most web servers can handle hundreds of concurrent connections, connection exhaustion attacks attempt to use all available incoming connection slots. If that should happen, the server will not be able to process new connection attempts thereby denying access to legitimate users.

If the web application requests information from back end database sources, then multiple requests to the database could exhaust a number of elements such as connection quota's, the database server's hardware resources (CPU/RAM etc) and if the application is vulnerable to SQL injection attacks, there may be the possibility of dropping tables, corrupting data and shutting down the server.

Locking out user accounts is a very simple process that can be achieved with many different tools freely available on the Internet. If the application has a lockout policy and if the attacker can identify valid usernames, by attempting to log on with possible passwords one of two scenarios will occur. The first and most likely is that the application will lock out that user account, thereby forcing the user to contact the administrative team to unlock the account. The second and most dangerous scenario is if the attacker gets lucky and identifies the correct password for a given username. They would then have gained access to the application and be able to carry out actions under the security context of the victim account.

These are just some example scenarios, yet there are numerous attack vectors available for application layer DoS attacks.

## Identification

It is very difficult to measure an applications resistance to attack without actively attacking it, but it can be done. It will involve much effort in analysing the source code of the application and attempting to identify possible flaws.

mount a series of DoS attacks and measure the response of the servers and application throughout

The most effective manner in which to assess an application's resilience is to mount a series of DoS attacks and measure the response of the servers and application throughout.

It is not recommended that this type of assessment be carried out by inexperienced staff. The potential dangers involved are far greater than regular penetration testing which already has a greater than normal level of risk.

Tests should also not be pitted against live environments as the likelihood of a system crash or failure is high. Many testing organisations will not perform DoS testing as a default service, but will consider it after fully explaining the risks to the client and managing expectations of the likely result.

## Countermeasures

Defensive measures to protect against network based DoS attacks should already be in place, firewalls, restrictive router configurations etc. These play no part in defending against application layer attacks but they are a mandatory step in producing a defence in depth strategy.

one of the best defensive measures is to ensure that the application code is resilient to attack

One of the best defensive measures is to ensure that the application code is resilient to attack, so it does not matter if the DoS traffic is detected or not. That is easier said than done however and may require much effort in auditing the source code.

Care should be taken that:

- ⇒ Resource intensive database queries cannot be initiated by unauthorised users.
- ⇒ Failed logon attempts do not identify the incorrect item – username or password.

- ⇒ Multi-server web farms should not be dependant on one data source.
- ⇒ Hardware resource management should be considered to ensure that any process does not exceed safe levels of CPU/RAM utilisation etc.
- ⇒ Web servers should quickly disregard unused sessions thereby freeing up resources for other users.

The next part of this series concentrates on **Insecure configuration management** and how they are used in exploitation.