WHITEPAPER

Managing and Protecting Web Server Resources: A Critical Piece of The Security Puzzle and it Can Save You Money.

About DOSarrest

DOSarrest, an internet security firm with customers who rank among with world's most recognized brands and agencies, is the result of five years of research, experimentation and mitigation of malicious traffic. With an expert, dedicated team of network security specialists, network engineers and developers focused on mitigating DoS and DDoS attacks, DOSarrest's solution is a comprehensive, multilayered cloud based DDoS protection service that has successfully mitigated thousands of real world attacks for thousands of websites since 2007.

In June 2013, DOSarrest translated years of experience analyzing customer websites into a new product, Website Vulnerability Testing and Optimization (VTO). The DOSarrest team has assembled a suite of tools - some generalized, and some tailored to specific types of attacks - to scan your website, which is then complemented by expert security engineers who review the report for thoroughness and accuracy. The result is a comprehensive report based on the most up-to-date threats and backed by DOSarrest's years of internet security intelligence.

www.DOSarrest.com

The server is not infinite.

A server needs to be carefully managed for its resources, to ensure the highest performance and operational efficiency. Your web server is no exception to this rule.

The experienced team at DOSarrest, DDoS mitigation and internet security experts, have been helping clients defend against DDoS attacks for years. They have had the opportunity to analyze hundreds of websites, and have discovered an alarming number of vulnerabilities and inefficient coding. They have seen insecure web applications exposed by hackers, bringing sites down with just one web transaction. They have also seen web servers unable to deliver while under legitimate load due to something as simple as improper CSS and cache settings. And with the constant changes introduced by developers, managed hosting providers and website administrators, security holes and suboptimal web coding are almost guaranteed to appear over time and can then be exploited by internet criminals and pranksters.

A 2008 SQLi breach compromised 130 million credit and debit cards and cost Heartland Payment Systems at least $140 million[1]. In 2011 a single hack cost Sony over $170 million[2]. More than 8 million usernames, emails and encrypted passwords were stolen from Gamigo during a hacking incident in 2012, and publicly shared[3]. In recent years, hackers have been generating real concern and attracting serious media attention with the colossal amount of damage they have been able to cause. We read almost daily about how another organization's data was compromised by a hacker, with greater financial and political implications each time.

But hacking is not new. One of the earliest documented examples of a hack was in 1903, when disgruntled magician and inventor Nevil Maskelyne interfered with John Ambrose Fleming's wireless telegraph demonstration by hacking and sending insults and rude poems[4]. His hack did then what hacks still do today: it exploited a flaw in the system to inflict damage. In the case of the wireless telegraph incident, the damage was done to reputations and didn't prove significantly expensive. In today's world where billions of dollars worth of business intelligence is available online, the damage can potentially be far worse.

Whatever their particular flavour of havoc, all hackers play the same game: they create tools that exploit weaknesses in systems, and then use the internet to access those systems. What they do from there is limited only by their imagination: extracting customer data to sell to competitors; mounting a DDoS attack to take a site offline; inserting a virus to destroy systems and data; retrieving proprietary information to use for their own benefit. The possibilities for how to inflict damage on an organization are endless.

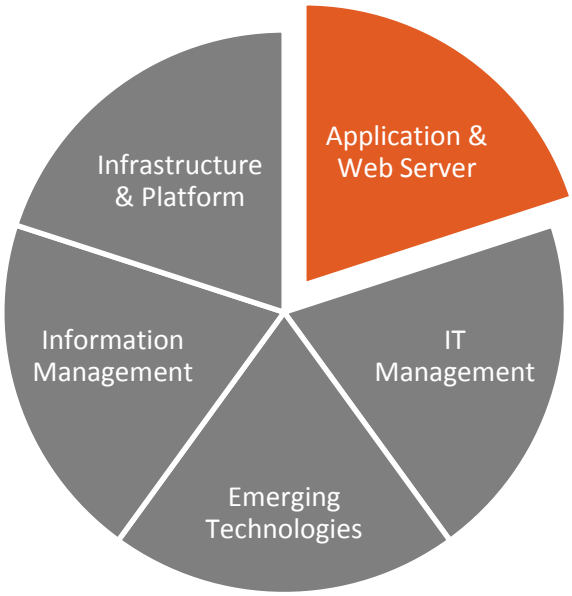
[1] http://www.computerworld.com/s/article/9176507/Heartland_breach_expenses_pegged_at_140M_so_far
[2] www.networkworld.com/news/2011/052311-playstation-network-hack-will-cost.htm
[3] http://www.forbes.com/sites/andygreenberg/2012/07/23/eight-million-passwords-spilled-from-gaming-site-gamigo-months-after-breach/
[4] http://www.newscientist.com/article/mg21228440.700-dotdashdiss-the-gentleman-hackers-1903-lulz.html?page=1

The Security Puzzle.

With the huge variety of attacks and vulnerabilities, and the costs involved in recovering from an attack, the business of securing your business can be a complicated and overwhelming matter. The more evolved the internet and all its entanglements become, the greater the opportunity for risk, or from a hacker's standpoint, opportunity for exploitation.

All organizations with a web presence must consider several factors when it comes to securing their assets. There are laptops to be patched, software to update, staff to train in acceptable online behaviour (ie, don't click on attachments in emails that come from people you don't know). You house your equipment in a 24/7 data centre with built in n+1 redundancy so you'll never experience network downtime. You've done your disaster planning, installed firewalls, SSL certificates, and anti-virus programs. You have encrypted, backed up and masked your data.

Figure 1. The Pieces of the Security Puzzle.



Of all the security precautions mentioned above, perhaps the most overlooked is web server security. This should be of particular concern because the internet, which can of course be accessed from anywhere in the world, connects directly with your web server in a variety of ways, and it is these connections that hackers exploit.

What Makes Web Servers Vulnerable?

Web servers operate by definition under a particular set of circumstances that makes them attractive to hackers. As a result, web application attacks are on the rise. In their annual report on worldwide infrastructure security, Arbor Networks reported that 86% of the respondents to their survey experienced application layer attacks.[5] Interestingly there was a 54% increase over the prior year in attacks focusing on HTTPS applications. This means attackers are increasingly using more surgical tactics in pursuit of the most valuable data, a statistic which echoes the DOSarrest team's experience.

What are the characteristics that enable hackers to take advantage of web servers and web applications?

1. TCP ports 80 and 443 must remain open.

In order to be available to send information when someone on the internet attempts to access it, web servers must "listen" using TCP port 80, or if they are using an SSL certificate, port 443. These ports must remain open, and relatively unprotected, which presents a tempting access point for hackers. Aspects of your website which involve web applications like shopping carts, login pages and contact forms all provide a connection from the internet directly through these ports to critical data like customer information, financial details, or passwords. It is only a matter of time and a hacker's ingenuity and determination before that divide is bridged and the information is in the wrong hands.

2. Access to valuable content.

Web applications often involve business-critical data. Examples include a customer login area (with access to valuable customer contact and account details); a shopping cart (with access to billing details); wikis (which may include proprietary business data); employee login (which may include confidential employee data). The value of this data can vary depending on who the hacker is or who the hacker is selling it to, but in most cases this is information that is in a business's best interest to protect at all costs.

3. Updates, revisions and new modules each risk opening new security holes.

Every time an application is modified it needs to be thoroughly tested to ensure that the changes did not compromise its security. Even small changes in code can introduce security flaws that create an opening for hackers, especially given the fact that new techniques for exploiting web applications are emerging every day.

[5] Arbor Networks: Worldwide Infrastructure Security Report, 2012, Volume VIII

4. Applications built in-house lack testing rigour.

The problem is that lots of people can develop web applications. Not all of them, however are capable of developing secure ones. Organizations regularly have their own coders develop the web applications required for their business. This has the twin benefits of being customized to suit the exact needs of the business, while keeping costs low. However applications developed in-house also carry risks. Whereas applications that are developed commercially are rigorously tested by the parent company before releasing to consumers, in-house coders may not have the experience or the breadth of expertise to ensure that an application is thoroughly tested before use. Developer bias can also create blind-spots. Screening as a result can be incomplete, resulting in security holes.

5. Off-the-shelf applications have low default settings.

When applications are created and sold off-the-shelf, the default security settings are set too low to be very useful. This is done to provide the greatest opportunity for compatibility with other systems and software, however it is important for someone with experience to ensure they can customize the security settings to be more restrictive where possible.

6. Infrastructure, software or policy changes.

New attacks are constantly emerging, so any changes in server infrastructure, updates to security policies or other interfaced software can expose new vulnerabilities, which can quickly be accessed via the internet and exploited.

How Do I Secure My Web Server?

As we have now discussed, web servers are inherently prone to a variety of exploitations. To ensure your web server is secure, begin by assessing potential weaknesses. Start by taking the Web Server Vulnerability Quiz (Figure 2) below.

Figure 2: Web Server Vulnerability Quiz

Web Server Vulnerability Quiz

Ask yourself the following questions:

1. Was the site created more than six months ago?
2. Do you have web applications (like a shopping cart or a login page) running on your site?
3. Does more than one person have access to modify your site, or associated web applications?
4. Have permissions for users been disabled if they are no longer meant to have access?
5. Have any of your web applications been modified within the last 3 months?
6. Have all the latest security patches been applied to your server?

If you answered YES or didn't know the answer to any of the above questions, your website could be one of the **9 out of 10 websites** that is at risk.

While the task of sealing up the vulnerabilities on your web server can be onerous, there are some key precautions that every company can take:

1. DEFENSIVE PROGRAMMING
2. REGULAR SECURITY AUDITS including:
    a. WEBSITE VULNERABILITY TESTS (DOSarrest Website Vulnerability Testing & Optimization)
    b. LOG ANALYZERS

This paper will examine the compelling business benefits that organizations can derive from all three website protection strategies, with an especially close examination of website vulnerability testing and optimizing, presenting the DOSarrest approach.

Defensive Programming

Defensive programming is a mindset used when writing software that should be utilized in the very initial planning stages of an application's development, through launch, upgrades and even into the wrap-up and decommissioning when has served out its useful life. Defensive programming takes security into account when writing the code, to ensure that inasmuch as issues can be foreseen, that they are addressed up front rather than causing major damage later. Some of the important rules of thumb for operating with a defensive programming mindset:

1. Use only known good libraries OR reuse known good code.

Something that was developed in-house, or is a "state-of-the-art" technique, hasn't been used in the wild of the internet, and can not truly be tested until it has been made accessible to the world at large and hackers have had a go at it. So rather than take the risk of reinventing the wheel, use known good libraries that have already stood the tests of time and regular use and abuse. It doesn't have to come from somewhere else, but even good code that has been successful within other parts of your own organization can be useful to replicate. Just be sure that it is truly good code, and you are not reintroducing legacy problems. Reusing sloppy code is much worse than creating something good from scratch.

2. Avoid data structures and patterns that are prone to abuse.

Languages like C and C++ were written in the 60s. Other languages like Java have then been built off of them. In this particular instance there are fundamental flaws in the original language that create known bugs or issues that are easy to exploit. Buffer overflow is an example of an issue with C and C++. To give an example of buffer overflow, let's say a program asks for your name. The coder may have allowed for 100 characters in the field. But a hacker could insert hundreds of thousands of characters and inject a piece of malicious code. Poorly designed code will accept all of that and write it in memory somewhere, overwriting existing memory with malicious code.

3. Use explicit error handling, never "on error resume next".

Software needs to be coded to handle every specific error in an appropriate way so as not to create opportunities for access. Hackers look for opportunities to create error messages that are very general and leave part of the program open awaiting input. The part that is awaiting input can be injected with malicious code if not coded properly to close off when an error occurs, or an appropriate response is not given.

Some less experienced coders may use the command "On error resume next" to ensure a user doesn't get stuck on a simple error and not be allowed to continue. This essentially causes the program to overlook the error and carry on authenticating. The problem is when the errors were legitimate problems and then a malicious user gains access.

4. Rigorously validate, sanitize and normalize all input.

Whenever you are coding for input from the end user or another program, it is critical that you do three things:

4.1 **Validate.** Make sure the input is in the format you are looking for. For example that a phone number is numeric and a name is alphabetical.

4.2 **Sanitize.** An SQL injection, one of the most common kinds of hacks, can be made up of numbers and letters and spaces and punctuation, much like an address. It needs to be sanitized, using sanitizing code, to ensure it contains none of the kinds of strings of information that might be found in an SQL injection, for example.

4.3 **Normalize.** When you leave a field open you may get any number of permutations of the data you are looking for. This can be more difficult to protect against, and so by normalizing, or forcing users to input data in a particular order or style, you can ensure that the possibilities are limited to the few that you know you can for sure run your checks against.

5. Assume a 'zero trust' model for data - even internal data.

Whenever your program is getting data from somewhere, assume it is NOT safe. Even if it is pulling data from within your own system and was never used externally, or if it was only just put there by your program and then immediately retrieved, it can still be altered by malicious code that you didn't know was there. It is important to include a function written into the program that sanitizes and normalizes the data, so every time the program reads or writes that data it has to go through that filter before being used. Just because it comes from a source that had zero contact with the outside world, it doesn't mean it hasn't been affected by malicious code.

6. Use the least permissive approach possible.

When designing systems to work with each other, you have control over what the other program can access directly. In a perfect world, it would be easiest to give every program access to every other program so they would work together seamlessly and in harmony. But with malicious users and tools scanning the internet constantly for opportunities for exploitation, it is safest to only permit the bare minimum access necessary to achieve the program's goal. Limiting permission is important for both other programs and users.

7. Assume that any bugs are potential security flaws.

When an unknown bug is reported, assume it is a security flaw and treat accordingly. By assuming the worst you will take the most precautions and ensure nothing unpredicted happens as a result.

8. Adopt secure coding standards.

Create or adopt a set of secure coding standards. This will ensure all coders are working to the same sets of expectations, and that subsequent versions are not decreasing the security of the application, but are maintaining the integrity of the original program.

Security Audits

There are two forms of security audits that are both critical to your web security strategy. One method is a proactive scan of existing vulnerabilities in order to identify and address weaknesses before an attack happens. The second uses log analyzers, which are essential with regards to incident response,

1. Website vulnerability testing

The other security audit method is website vulnerability testing. A vulnerability scan can crawl through an entire site and identify insecure elements and applications, and report inefficient settings in the website code. These are then assembled into a report, which the team responsible can then use to ensure servers are as secure as possible. The best vulnerability scans are kept up to date with information on the latest threats, assess virtually every aspect of the web server, and are backed by experienced security staff who can complement the technology with the subtlety of human experience. Scanners need to be able to interpret results of various technologies and also be able to address policies and procedures, like password strength and cryptographic libraries. Vulnerability scans can be run internally by a company on its own servers, or can be outsourced to a third party.

WEBSITE VULNERABILITY TEST BENEFITS

- Provides an opportunity for proactive management of server resources to protect your business and save money.
- Puts control in the hands of the business owner, by ensuring there are no security holes that can be breached.
- Proactively evaluates many aspects of a web server and detail a report of what needs to be done in order to secure it.
- Attends to the problem of web server security *before* an attack takes place, mitigating damage.

WEBSITE VULNERABILITY TEST DRAWBACKS

- Only effective if used proactively. Does not provide any benefit during a live attack.
- Useless if the scanner is not up-to-date, so need to ensure the provider is a trusted source.

The DOSarrest Approach: Website Vulnerability Testing & Optimization (VTO)

During the years since 2007 that the experienced team at DOSarrest have been helping clients defend against DDoS attacks, they have had the opportunity to analyze hundreds of websites. They have discovered an alarming number of vulnerabilities and inefficient coding, and were compelled to create a solution that would help customers learn more about protecting their own websites, thereby making their web servers more difficult to breach.

The DOSarrest Website Vulnerability Testing & Optimization (VTO) report is a suite of tools that intelligently crawls your site, identifies insecure elements and applications, and reports inefficient settings in your website code. It can pinpoint practically any vulnerability or design flaw a website may have, and also:

- Assists in securing web applications against vulnerabilities, by analyzing the site with the most advanced SQL injection and Cross Site scripting testing;
- Checks for industry information security compliance, such PCI/DSS, HIPPA, SOX, and many more;
- **ONLY AT DOSarrest:** Provides specific details on how to optimize caching (for a CDN or otherwise) and minimize request overhead and payload size.

THE DOSARREST ADVANTAGE: Optimize caching and save your money

Thanks to the experience and network savvy of the DOSarrest team of experts, unlike other vulnerability scans the DOSarrest VTO not only checks for vulnerabilities in website code, but also provides details on how to optimize caching (for a CDN or otherwise) and minimize request overhead and payload size. This can have an immensely positive effect on the server's computing power, which improves performance and frees up space meaning you don't have to spend money on costly server upgrades.

How it Works

The DOSarrest VTO report consists of a series of scans, and depending on the size of a website, can take up to 6 hours to complete. This test will not interfere with a website's normal operation in most cases, and therefore can be run at almost any time. The DOSarrest team is actively involved in the testing process and reviews results to ensure all potential vulnerabilities are thoroughly explored.

What it Provides

A report is created, comprised of four sections:

1) An **executive summary**, which breaks out key information from the report indicating the number of vulnerabilities in four categories of severity;
2) A detailed description of **where the security lapses are located** on the website;
3) A detailed explanation of **what the particular issues are** and how to remedy them;
4) A summary showing all **transgressions** of what we have determined to be web performance best practices.

What the Report Tests For

Vulnerabilities

- ✓ Cross site scripting
- ✓ OWASP Top 10
- ✓ SQL injection, including blind SQL injector
- ✓ PCI/DSS compliance
- ✓ HIPPA compliance
- ✓ SOX compliance
- ✓ Web 2.0 application compatible including AJAX, XML and SOAP (JSON)
- ✓ Google crawled content analyzer, find search engine hacker holes before they do.
- ✓ Directory traversal
- ✓ Test password protected login areas
- ✓ Test web-forms automatically including CAPTCHA protected forms
- ✓ HTTP Parameter pollution

Optimizations

The only website vulnerability test to offer the following scans is DOSarrest's VTO. The website's code is reviewed for all of the following and recommended fixes are provided.

- ✓ Excessive external css
- ✓ CSS sprites
- ✓ Excessive external js
- ✓ Defer loading / parsing of js
- ✓ Compressing resources
- ✓ Browser caching
- ✓ Proxy caching
- ✓ Minimize redirects
- ✓ Optimize images
- ✓ Remove unused CSS
- ✓ Consistent URLs

Backed by Expert Security Operation Centre

A DOSarrest security engineer can walk you through the report and help your technical team plug any holes found and rerun the test to ensure everything on the website is secure as it can be. Regularly scheduled tests are the key to keeping a website secure.

As illustrated by Table 1 below, DOSarrest's Website Vulnerability Test & Optimization provides a level of detail, relevancy and precaution not found in other vulnerability tests.

Table 1. Comparison of DOSarrest's VTO with Other Vulnerability Tests.

	TYPICAL WEBSITE VULNERABILITY TESTS	DOSARREST'S WEBSITE VULNERABILITY TEST & OPTIMIZATION
Uses most up-to-date information possible, including zero-day information on all latest vulnerabilities	No	**Yes**
Fully managed solution with a team of experts to support the scanning technology with human experience and coaching	No	**Yes**
Identifies logical flaws that can be difficult to find, like weak cryptographic functions, information leakage, etc.	No	**Yes**
Uses a heuristic scanning technology	No	**Yes**
Scans for full scope of variants possible for each vulnerability	No	**Yes, when initial testing requires it.**
Optimizes caching	No	**Yes**
Understands behavior of applications with dynamic content such as JavaScript and Flash, etc.	No	**Yes**

Conclusion

Websites are useful communication tools for organizations around the world, and they will continue to be for the foreseeable future. But with those who wish to share information, there are also those who wish to steal it, and hackers will continue to pester companies *ad nauseum*. Until there is a completely hacker-proof website created, organizations must rely on defensive programming up front, log analyzers during an attack, and proactive website vulnerability testing on an ongoing basis. When organizations use DOSarrest's Website Vulnerability Testing and Optimization they also benefit from an optimized web server, which saves valuable server resources and money.

For more information about DOSarrest's **Website Vulnerability Testing and Optimization** please visit www.DOSarrest.com.