NETWORK BOX USA, INC.
MANAGED SECURITY SERVICES

Why You Need a WAF

a white paper by Pierluigi Stella, CTO of Network Box USA, Inc.

At the time of writing of this whitepaper, attacks against web servers are (by far) the most prevalent issue in cyber security. The Verizon Data Breach Investigations report for 2014 showed that 35% of breaches were caused by web application attacks. And by mid 2016, this number is only rising.

Hackers are getting control of web servers for different reasons, the main ones being the ability to control a server and spread malware. Often, these two objectives go hand in hand.

Why do hackers want to control a web server? Because a server is often hundreds of times more powerful than a workstation, and that allows them to have a platform to launch attacks from a single point, rather than having to deal with multiple workstations. Servers are often also connected to larger bandwidths, enabling these attacks to be increasingly efficient. They are also online 24/7, users don't turn them off at night as they tend to do with a personal computer. And they are connected to public IP addresses on a public network, not in someone's home or office. A server can, and most times is, used as a command and control center to manage a network of zombies – a botnet. Finally, a server can be used to 'serve' malware. In this case, the web server may not even look compromised, and yet malware lurks in the background, ready to attack unaware browsers.

This should make one thing abundantly clear – protecting web servers is critical; not only does a company stand to lose data, the server can also be used as a bridge into the company's network. Or to launch attacks against other companies; and although we have yet to see lawsuits in such cases, it is only a matter of time before someone gets sued over allowing their server to become a tool for conducting attacks against someone else.

But how exactly does one protect a web server? Especially one that is running an application, which is, in turn, connected to a database?

Up until a few years ago, the practice was to put the web server itself into a DMZ, the database server on the LAN, and 'hope' that the firewall and IPS could stop malicious traffic between the two. The idea being that if the web server were to become compromised, it is easy to rebuild and data is not lost. Unfortunately, this method of protection is insufficient; it never really was to begin with, but today, it is clear how and why.

First of all, too many people misinterpret the meaning of DMZ and allow all traffic, rendering the very idea of DMZ pointless. Second, the ports needed for the web server to run queries against the database are the very ports hackers are trying to attack. This does not in any way prevent a SQL injection or a database DDoS attack, as both methods use database ports that are open from DMZ to LAN to allow the web server to run queries.

The next idea was to set up IPS in line with firewall, to intercept malicious traffic. Though not a bad idea, it is, unfortunately, very limiting. An IPS is a layer 3 protection. Think of it as an AV at the packet layer. It reads the content of a single packet, inspects it against known vulnerabilities and exploits, using a mix of signatures and heuristics (behavior analysis). And that is all it does. You might be wondering what the issue is. IPS is used for many different reasons. The problem is that IPS is a layer 3 protection, whereas HTTP is a layer 7 – it is an application layer protocol. And web servers run applications. This means that attacks can be carried on using transactions that span across several packets; and each packet can appear to be perfectly clean. An IPS can protect against some low level SQL injections and XSS attacks; it can protect against things like the Slammer worm, on which more information can be found **here**.

SQL Slammer is a computer worm that caused a denial of service on some Internet hosts and dramatically slowed down general Internet traffic, starting at 05:30 UTC on January 25, 2003. It spread rapidly, infecting most of its 75,000 victims within ten minutes. The reason why that is possible is because the Slammer was less than 400 bytes in size, so the entire malware fit into one single packet.

But a web server attack of recent times is almost never just a single packet of malware. More often than not, such attacks try to work around the HTTP protocol, injecting commands that are not part of a normal sequence, to gain access to resources on the server or cause the application to react in a way it wasn't designed for. In other words, an IPS cannot understand web application protocol logic, and cannot fully distinguish if a request is normal or malformed at the application layer.

Therefore, an IPS cannot be effective against all potential application vulnerabilities. Typically, an IPS ends up producing too many false positives, which in turn yield two possible reactions – either (1) it delays application response time and incorrectly interferes with the application; or (2) it allows attacks as normal behavior in an attempt to reduce false positives.

An IPS looks at signatures and anomalies; a WAF looks at behavior and logic, analyzes traffic in both directions, looks at what is being requested, and what is being returned.

A Web Application Firewall's main task is to protect web applications by inspecting the semantics of the flowing traffic and also inspecting HTTP/HTTPS for typical attacks at layer 7 such as SQL Injections, Buffer Overflow, Cross Site Scripting (XSS), File Inclusion, Cookie Poisoning, Schema Poisoning, Defacements, etc. To obtain a better understanding of the topic, a good source of information is found at **this link**.

OWASP is an open software security community collecting, among other things, the list of top attacks against web servers. Any WAF on the market today will have to at least protect web applications and servers from the OWASP top 10.

Another aspect of using a WAF is called SSL offloading. The internet is headed towards complete encryption. Increasingly, websites are using HTTPS, even when there is no confidential data to be protected. If the website runs an application that requires confidential data to be entered, such as your online banking portal, the choice of HTTPS is obvious. But there are plenty of examples of sites that are using HTTPS even though in principal they wouldn't need to, because there is no private data to be protected.

Soon after the NSA scandal of 2014, Google.com chose to use HTTPS for all their pages. Yahoo! has done the same. And the examples are countless. Encryption is adopted not just to hide searches, but to also guarantee the authenticity of a website. Too many attacks are conducted by spoofing the looks of other websites, and the mechanism of private/public key/cert inherent to the HTTPS protocol is likely the last bastion against a widespread diffusion of such attacks.

For instance, when you go to **https://www.networkboxusa.com/**, the certificate on that website (published and guaranteed by a certificate authority) tells your browser that you have indeed reached the intended site, and not some hacked, spoofed site that might try to collect information to, in turn, conduct an attack against you.

What does this really mean for one's webserver?

If the server capacity was designed for HTTP and suddenly every single transaction is encrypted, the CPU load can become such that the server itself may not be able to handle it. One feature where a WAF can help is HTTPS offloading – the certificates are placed on the WAF, the encrypted connection is terminated on the WAF. Between this and your servers, the traffic does not necessarily need to be encrypted, thus "offloading" the function to the WAF. This allows control over which version of TLS to use. That said, given that every version of SSL has now been compromised, so in reality, no one should really be using that for their HTTPS protocol. Yet we still see many websites that have not been updated and corrected, likely because the software version cannot be updated (for the reasons explained earlier). The WAF can easily enforce use of the appropriate TLS version, without the need to touch anything at all on the web server.

Because the WAF decouples the traffic between web server and internet, and the browsers are no longer connecting directly to the webserver, a WAF is an inbound proxy. Once we consider this, we see how it is possible to deploy other technologies commonly applied to outbound proxies to a WAF, such as AV and access control. Applying AV rules to an inbound proxy may sound like a stretch because of possible performance limitations, and there are certainly issues with this; but it is vital to remember that in this case, we are not protecting a workstation.

We are protecting a very specific technology; so the applied AV can be limited to only signatures and heuristics that are relevant to a web server; there is really no need to run 11 million signatures to protect a web server. And this can make running an AV more plausible.

At this juncture, it is also worth mentioning what WAFs do not do.

A WAF will not enforce access control in the traditional meaning of the term. Do not be confused by the term "firewall" present in the name of this technology. A WAF only protects the server farm behind it, adopting signature based or anomaly detection algorithms but, unlike a network IPS, it will focus only on the HTTP and HTTPS protocols. A WAF is a layer 7 technology, not layer 3.

Some Web Application Firewalls will also provide layer 7 protection against DDoS attacks, although most vendors separate the 2 types of protection, offering specific DDoS protection as a different service or appliance. However, since this whitepaper pertains to WAF, we will not dwell on the details of DDoS attacks against web servers and why protection against such attacks is equally necessary.

Gartner's magic quadrant for WAFs for 2014 says:

- Protect web applications against hackers' attacks, to monitor access to the web applications, and to collect access logs for compliance/auditing and analytics

- Primary WAF benefit: providing protection for custom web applications that would otherwise go unprotected by other technologies that guard only against known exploits and prevent vulnerabilities in off-the-shelf web application software

- WAF technology

    - Maximizes the detection and catch rate for known and unknown threats
    - Minimizes false alerts (false positives) and adapts to continually evolving web applications
    - Ensures broader adoption through ease of use and minimal performance impact

There is another aspect to WAF technology that must not be underestimated. A web server runs 2 'applications' – the web server itself (be it Apache or IIS or something else) and the user application (what you see when you go to that website). And in between, there are the tools used to develop the user application itself. These could be PHP, WordPress templates, java, ASP, you name it.

Each of these layers can have vulnerabilities. Apache has its own and plenty of them; IIS does as well. But these are often known, announced, patched and therefore, made irrelevant after a while. Aside from that, once they are known, it is often possible to create IPS signatures that can protect the web server itself.

The real issues, the ones that can cost an organization its entire database, are those caused by the user level application and tools used to develop it. Whether it is an off-the-shelf product or one developed in-house (which is most often the case), software invariably has errors, which leads to vulnerabilities. It is not a matter of if, but how many, and how exploitable.

For example, PHP and Java have vulnerabilities. But these aren't tools you can always update as this (act) could break the actual application. As such, these tools are often not updated, vulnerabilities are not patched, and they cause the application to be, well, laid bare and open to attacks.

Finally, the application you're currently using? More than likely, it was not developed with security in mind. No matter how much we discuss the topic and we talk about security driven application development, how many people and companies really even know how to do that? How many developers test their applications from a security stand point?

And what if the application in question is old, a legacy development that was written 10 years ago? Developers have moved on, documentation is scarce, if present at all, and yet the application plays a vital in your company's business.

Updating the tools it relies upon isn't even a question – the application will break. Fixing the applications issues may well be even harder and often unfeasible. The entire construct is a vulnerability disaster waiting to happen.

This is likely the most important example of where a WAF can be very useful.

A WAF will have a configurable layer where a business owner, or vendor can create specific signatures. Therefore, instead of breaking the application, or living with one that is vulnerable and can expose confidential data, a WAF allows for the creation of customized protection, if you like, dedicated signatures tailored to very specific applications. This allows organizations to achieve strong protection for their web applications without the need to alter functionalities, and without having to fuss over updating them in a rush.

Note that we are not advocating running that old COBOL application for the next 40 years. And yes, sooner rather than later, we'd be better off scraping everything and rewriting applications with more advanced tools. But the adoption of a WAF ensures that process to be just that – a process – instead of a frenzied decision dictated by the need to cover up security holes.

About Network Box USA, Inc.

Leading IT security service provider, Network Box USA, was formed in response to increasingly sophisticated threats stemming from widespread use of the Internet. We aim to provide organizations of all sizes with enterprise class, true real-time protection that is effective and affordable.

For more information, visit our website: **www.networkboxusa.com**