Web Application Security Requires In-Depth Context Intelligence

Effective mitigation against web-based attacks must reflect the scope, scale, and nature of threats — in other words, be multi-layered, massively scalable, and context aware.

Long gone are the days when organizations can simply erect a firewall at the edge of their internal network to protect themselves against external cyber threats. Organizations today have significant IT assets outside their internal network and therefore also outside the protection of a traditional firewall. Those assets include web applications, which in many respects stand as a proxy for the organization itself out in the world. In addition to the familiar website and ecommerce use cases, many organizations rely on the web to deliver entertainment, app functionality, software as a service, cloud compu¬ting, and more. For purposes of performance and availability the resources comprising web applications are typically highly distributed outside the firewall. At the same time, given the high value content and functionality these applications provide, they are also intimately tied into the organiza¬tion's core IT infrastructure inside the firewall. That often means that a threat to the web application is a threat to the organization as a whole and that protecting the organization means protecting the web application too.

But protecting a web application is different from protecting an internal network. The threats are different. The risks are different. And the environment is different. So, security measures must fit the new threats, the new risks, and the new environment.

Why Web Application Security Is Different

Let's start with what needs to be protected. The reason organizations deploy web applications is to deliver reliable, fast service to end-users wherever they are with seamless scalability. In other words, security is not just about protecting the organization's IT systems anymore (which is still vital); it's also about protecting the end-user / customer. In that context, security is not just about "threats." Yes, organization must protect against attacks on web-based systems, just like they protect internal systems. But now they must protect against attacks on external users' systems as well. And in addition, they must protect the user's experience. And what users don't want to experi¬ence is either slow response times or actual attacks, like having their data stolen, or being exposed to inappropriate content. From a business perspective, a lost customer is a lost customer, whether it's because the customer lost patience or data.

In other words, **from the user's perspective security is part of performance optimization.** That makes good business sense, but also good technical sense in that many of the technolo¬gies underlying better web performance also enable better threat mitigation. In fact, web administrators also have a better "user" experience if threat mitigation results naturally from performance optimization rather than opposes it. But the technical synergy between optimization and mitigation is best explained by looking at the threat itself.

What Makes Threats to Web Applications Different?

Much of what's different about the new threats is obvious: Threats are against IT assets outside the enterprise firewall, not necessarily against assets inside. Threats may be widely dispersed geographically — both in terms of where attacks come from and their targets. What is also different is that an attack on a website is now more likely to be against the actual business itself rather than just against the online equivalent of its corporate brochure. The fact that the website is the business in many cases changes the nature of the threat — making it potentially more disruptive and the attack itself probably more sophisticated.

Much greater threat sophistication and scale are in fact the two biggest new factors driving organizations to seek new ways to protect themselves in the web application era — ways which, not surprisingly, must also be much more sophisticated and scalable.

What makes threats more sophisticated? Most web applications live in the wild, outside the enterprise's protective walls. Anyone can attempt a request to the web application; the request can take virtually any form and contain any content, and the application may offer any number of interfaces ready to handle those requests with virtually any kind of logic. Almost all of those requests may be from legitimate users. What's hard is filtering out only those requests that are not legitimate and dealing with those appropriately — such as by challenging the request, serving a 404 error page, or dropping the connection entirely. On the one hand, the attack logic must be sophisticated enough to do actual harm and to resemble a legitimate user request. On the other, the countermeasures needed to defeat the attack must be advanced enough to identify legitimate-looking attacks, to actually thwart the attack, and to also let through requests that really are legitimate so the users' experience is protected.

Now take the issue of scale. A foundation technology of website performance optimization is the CDN (content distribution network). A CDN widely mirrors websites as proxies at multiple points on the Internet so users don't have to wait for content from the origin server, which may be thousands of miles farther away than the nearest proxy. CDNs also distribute server load so that the same server doesn't have to respond to each user, which also speeds performance. The result is that CDNs make websites highly scalable, and so does the cloud technology that now supports most CDNs. When CDNs are in the cloud, more servers can come online automatically to answer more user requests — making both the website's global footprint, as well as its size as a target, potentially infinitely larger. That scale can work in the attacker's favor. Not only might there be a bigger target to attack, but web technology also enables the attack itself to be scaled, thereby creating multiple launch points from which to mount an attack. The attack could, for example, "spin up" more servers of its own in the cloud or possibly trick the browsers of thousands of users to accept malicious code that then sends bogus requests to the target website. Matching this kind of threat requires countermeasures that are also massively scalable in addition to being highly sophisticated.

Why Different Kinds of Attacks Call for Intelligent Countermeasures

The fact that threats to web applications can be highly sophisticated and scalable also means that attackers have lots of modes of attack to pick from and that modes styles can be very different from each other. Effective countermeasures must therefore have the intelligence to recognize different modes and respond with a defense tailored to the particular mode employed. To see why, let's look at four of the most frequently used categories of web application attacks:

- Volumetric (denial of service, slow-&-low)
- Injection
- Cross-site scripting
- Session hijacking

Volumetric. Here the objective is to overwhelm the target server with so many requests that it crashes. Two key strategies typify this style: distributed denial of service (DDoS) and slow & low (slowloris). DDoS is an attack where potentially thousands of automated agents distributed around the Internet act in the role of human users hitting the target server with potentially millions of simultaneous requests for service. Slow & low attacks are similar except that each agent's requests are spread out (i.e., are slower) so that a connection with the target server is maintained as long as possible without the connec¬tion timing out. Ultimately the target can't service all the requests so it crashes. What makes this attack hard to detect is that requests are slower so they don't resemble a DDoS attack. One way to detect this attack, however, is to analyze traffic to see how long transactions typically take and how many requests are typically involved. For example, if a particular user takes two hours to conduct a transaction instead of the average five minutes and launches 1,000 HTTP sessions instead of the typical five, it may be an attack. However, the only way a firewall would know this is if it had the intelli¬gence to: a) monitor activity to know what a reasonable number of sessions might be over a reasoable period of time, and b) block the requests it deemed "unreasonable."

Injection. The objective here is to exploit an application's syntax, as in SQL, PHP, Java Script, OS shell commands, or other code to carry out malicious instructions contained ("injected") as data in a command or query. These instructions trick the web application into doing things the application's owners might not willingly permit — for example, to access private customer data. Protecting against this type of attack usually involves scanning incoming requests for keywords ("select" and "delete," for example) that might be part of an instruction's programming syntax. Such keywords can be weighted and each request ranked on its number of more highly weighted keywords. Given sufficient intelligence, counter¬measures can then decide whether to allow the traffic.

Cross-site scripting. Countermeasures against this style would also involve intelligent screening of incoming requests — such as to determine whether the traffic was coming from human users, benign bots (as in search engine web crawlers) or malicious automated agents. Cross-site scripting happens when a web server sends malicious code to users' web browsers which causes the web browsers to do things the user didn't intend — like defacing a target website. One intelligent defense of potential target sites is to place cookies and also JavaScript agents inside user browsers. Intelligence in the CDN would also qualify each request based on factors like volume, frequency, and type as to whether the request is likely to be unsafe. The CDN could then instruct the embedded browser agent to "fingerprint" repeat visitors — where the agent takes a snapshot of the browser type, the operating system, and other identifying information and stores it in the cookie — so that their repeat visits can be counted and compared.

Session hijacking. These attacks are similar to cross-site scripting, the major difference being that the attacker doesn't send malicious code to a browser to redirect traffic to a site or to a (possibly private) page within a site. The attacker instead hijacks a website's existing redirect, one that normally accepts a URL as a parameter anyway. Sending a malicious URL may redirect unsuspecting users to the attacker's malicious site (like a phishing site) or alternatively may grant the attacker access to a private page on the same site as the redirect. The target of this attack is the hijacked site, the one with the vulnerable redirect, not another (probably malicious) site users are being redirected to as a result of the attack. One way to protect the target is to scan requests for the presence of suspect URLs and to match those against a blacklist of known offenders (including those "fingerprinted" visiting this and other sites). Obviously, the more traffic that can be analyzed across the Internet, the more complete the blacklist can be and the better the protection — which speaks to the need for intelligence that is also highly scalable.

Scalability Calls for In-Depth Context Intelligence

But even if you have the intelligence to recognize and mitigate these different threats, how do you make that intelligence scalable? And how do you make threat mitigation an ally of performance optimization rather than an enemy? What's clear from looking at just these four kinds of threats is that there could potentially be many, and many different, steps involved in countering them. Denial of service, for example, is a very different kind of threat than session hijacking and so requires very different methods of detection and mitigation. So how do you know which steps to take when?

If not choreographed correctly, those steps could impede optimization by holding up legitimate traffic while it's being analyzed and filtered. The reason traffic flow (and the user experience) is optimized instead of diminished is because the same two key concepts underlie both threat mitigation and performance optimization: security in-depth and context intelligence. So it makes technical as well as business sense that the same CDN handling web application optimization would also handle web application security — *if it aligns with security in-depth and offers context intelligence.*

There are four reasons this is so:

- Some mitigation steps may work for some threats, not others
- Doing all mitigation steps at once at the Internet edge slows performance
- Mitigation may need to scale with the attack (more VMs, etc.)
- Context based mitigation can accelerate most content (e.g., intelligent caching)

Let's take security in-depth first. This was a time-honored defense strategy even before there were walls, moats, large vats of flammable liquids, and men with crossbows defending castles. One reason security in-depth works is that not every defense works equally well on every type of threat or takes equal advantage of the available "terrain." So you use various defenses in combination. Defenses on the Internet edge, for example, will be more effective at blocking volumetric attacks. That's because traffic at the edge can be routed away from the origin server very quickly — with mechanisms like border gateway protocol (BGP), which uses less of the IP protocol stack to make routing decisions, therefore requiring fewer compute cycles, and so takes less time — thus serving the dual purpose of more efficient threat detection/mitigation and more optimized application performance.

Tasks requiring more processing time, like running algorithms against packets, looking for possible malware inside, are better suited for the servers inside the Internet that are already making decisions (to optimize performance) about which content to serve users first. So, again, security and performance naturally align.

But the other reason security in-depth works is that it is not always necessary to do all these steps, or at least do them all at once. By deflecting volumetric attacks at the network edge, for example, you won't need to look at this deflected traffic again. That saves more capacity for serving content to users, thereby also optimizing performance.

So what about context intelligence? A CDN with context intelligence can distinguish different types of content and orchestrate the delivery of different types of content optimally based on rules. So, for example, static content (which makes up the bulk of most websites) can be delivered first and usually from cache, creating the impression that websites load instantly. Dynamic and curated content, a trailer announcing deals of the day, let's say, or social media banners, can be loaded later and probably before the user even notices they weren't there to begin with. Another example of context intelligence is to spin up new virtual machines whether to meet SLAs in the face of higher user demand or to scale up defenses against an attack.

There are two parts to context intelligence, whether employed for security or optimization. One part is the intelligence, i.e., the configurable business rules that tell the CDN what to do under various circumstances. A good example of a business rule is to spin up another virtual machine when the application's processing load reaches a specified threshold. Another good example is the rule that says to deny a request if its content scores too high in terms of possible injection keywords.

The other part of context intelligence is context awareness. In security, that means threat awareness, or awareness of the types of situations that indicate a likely threat — such as the high injection keyword score just mentioned or whether the website is being targeted by high volumes of requests from the same IP address over an extended period. That's the same context intelligence that, when applied to optimization, looks at factors like whether content will appear "below the fold" on a mobile device, so its delivery can be delayed in favor of content the user actually sees when the page loads. So, yes, context intelligence also explains the threat mitigation/performance optimization synergy.

So, let's see how this synergy works in practice.

Yottaa's Seven Layers of Strategic Defense

Yottaa is a CDN that applies scalable in-depth context intelligence in seven layers as a natural byproduct of web application performance optimization. It operates on the principals just outlined — that if one layer can't stop the threat another layer can and it's better to apply some defenses first and hold back on others. The seven layers are:

1. Only allow traffic on web ports
2. Geo-based blocking
3. Validate user against list of known attackers
4. Determine what's cached and what isn't
5. Global distribution, in-line, always on at the edge of the Internet
6. Rate controls
7. Web Application Firewall Rules

Layer 1. Only allow traffic on web ports

Legitimate web traffic should only arrive over ports 80 (http) and 443 (https). So all other ports are blocked. This is done first since there is no reason to examine traffic that arrives over an unauthorized port.

Layer 2. Geo-based blocking

Another immediate way to filter traffic so that other mitigation measures don't have to deal with it is to scan the traffic on ports 80 and 443 by geographic location. That could be done any of several ways, including by IP range or more granularly by blocks of IP addresses known as CIDR (Classless Inter-Domain Routing) blocks. This technique would apply, for example, if a website owner knows in advance that threats are likely to come from a certain region from where the owner does not expect legitimate traffic.

Layer 3: Validate user against a list of known attackers

One of the benefits of working with a CDN that operates at scale (i.e., has a global infrastructure serving many clients) is that it can quickly and efficiently build a comprehensive list of known bad actors. So, traffic that hasn't already been blocked automatically by port or geographic region can then be blocked if it originates from a source on the CDN's blacklist. The traffic that then passes through this layer successfully can be analyzed more thoroughly to see if it might pose a threat.

Layer 4: Determine what's cached and what isn't

A basic tenant of performance optimization is to serve static content from cache rather than waste cycles to reconstitute content that has not changed over the last many times the same content was requested. The same logic applies to threat mitigation, so requests for static (cached) content are always allowed. That way, further mitigation layers need only focus resources on requests for dynamic content and determine how best to respond to those.

Layer 5: Global distribution, in-line, always on at the edge of the Internet

If the context intelligence determines that this might be a volumetric attack, then the best place to respond is at the edge of the Internet. If edge-based intelligence scores the threat probability as very high, it can decide not to handle the requests at all and simply drop the connection. Alternatively, if it thinks this may be a case of the site suddenly becoming very popular with legitimate users (perhaps because a product was mentioned on Oprah) the intelligence could decide to distribute the requests to multiple widely dispersed servers around the Internet, such as by using techniques like BGP, discussed earlier.

Layer 6: Rate controls

If the volumetric attacks look to context intelligence as if they might be part of a slow & low attack, the CDN can apply rate controls to reduce the number of requests allowed to get through over a set period of time.

Layer 7: Web Application Firewall Rules

Up to this point, the traffic has not been analyzed for threats at the application level — i.e., for attacks attempting to use the logic of the application itself against the interests of the application's owner, such as by SQL injection or site hijacking. These checks are more compute intensive, therefore take longer, and so are reserved for traffic that has successfully passed the other layers. At layer 7, the context intelligence answers questions such as: Do requests contain keywords that could indicate an injection style attack? Do they look like they're coming from a malicious bot rather than a human user? If the answers are "yes" the context intelligence can then decide the appropriate level of response, such as to interrogate the user, serve a 404 error page, or simply drop the connection altogether.

Are You Highly Secure or Highly Optimized?

Here's the takeaway: Given the right CDN architecture, web application owners don't need to choose between high security and optimized performance. By definition, poor security is poor performance — just ask an owner whose website has been hacked. The real choice is to rely on technical strategies — like security in depth and scalable context intelligence — that naturally align web security with performance optimization. Today's security threats are far more scalable, more distributed, and more sophisticated than back in the day. And so must be today's web application firewalls.

Sizing Up the Cost of Business Threats

Obviously, the reason organizations deploy a web application is to serve the needs of the business — which means that ultimately threat mitigation is about protecting against business loses. And these losses can be substantial. Consider just these four:

Loss of eCommerce. Industry experts estimate that the average loss of an ecommerce outage is $540,000 per hour. Of course, how much you lose depends on how much business your site could have been expected to do over the course of an outage. However, it's unlikely that sales volume is distributed evenly over time, and even if yours is a small business, there's always the risk during an outage of losing that one big order.

Loss of revenue. Even if your website is not taken offline completely, an attack can slow user response to where users lose patience and abandon your site.

Legal jeopardy. Theft of personally identifiable information (e.g., credit card numbers, social security numbers) in particular can lead to substantial fines and civil lawsuits by injured parties.

Reputation loss. Probably the biggest loss that any business can suffer is damage to its reputation, the fallout of which can be lost business, bad publicity, customer abandonment, and more — perhaps for years to come.