ALERT LOGIC®

5 Techniques To Protect Against A SQL Injection Attack

"What is SQL Injection, and Why Is It Still Causing Issues?"

SQL Injection is a term dreaded by web developers, database admins, and CISOs alike. Even if your regular job in IT doesn't focus on cyber security or coding, chances are you've still heard of SQL injection and are familiar with the havoc that it has caused throughout the history of security breaches.

The SQL injection flaw – commonly referred to as SQLi – has been the bane of the web application space since first being identified in 1998 (and possibly earlier, depending on your source), when it was revealed in issue 54 of Phrack magazine by the cyber security researcher/hacker known as Rain Forest Puppy (a.k.a. rfp).

That article was *published almost 20 years ago*, yet he we are still dealing with SQLi on a regular basis today. Injection-type cyber attacks (of which SQLi is one) are still ranked as the highest-ranked threat by the Open Web Application Security Project (OWASP), whose Top 10 list of the most critical web application security risks is the de facto web application security standard for companies and regulating bodies around the world. Research findings from organizations like Verizon and Gartner also confirm that SQLi remains a very real and very problematic cyber attack vector. And to add fuel to the fire, this is also backed up by data we have collected here at Alert Logic. In the 18-month period of data-collection that our **Cloud Security Report** is built on, 55 percent of all observed cyber attacks against our customer base used SQLi.

But don't give up on creating your web application just yet. There is good news. SQLi is a well-understood cyber attack precisely because it has been around so long. That means that there are great mitigation strategies already out there. It also means that it's pretty easy to explain, which means more people can get educated about how SQLi works and the dangers it poses. **So, let's dig into SQLi.**

Follow these **5 techniques** to help you get ahead of cyber criminals when they come after your SQL-based apps that you have deployed in the cloud:

- **Secure coding practices**
- **Vulnerability scanning and penetration testing**
- **Layered defense**
- **Cloud security blocking and tackling**
- **Intelligent log analysis**

STEP 1 STEP 2 STEP 3 STEP 4 STEP 5

01 Secure Coding Practices

DON'T TRUST THE INPUT. THE FIRST LINE OF DEFENSE AGAINST WEB APPLICATION SECURITY FLAWS LIKE INJECTION AND CROSS-SITE SCRIPTING SHOULD BE THE CREATION AND DEPLOYMENT OF SECURE CODE.

The first rule of secure coding is that all input going into a web application should be considered untrusted and potentially malicious. The source of the data does not matter, even if you consider that source trusted (i.e. data coming only from an internal source of some kind and not from the public internet). Secure coding techniques can then be applied to make sure that all data that is coming into the web application is cleaned or blocked.

Let's look quickly at an illustration of why secure coding is so important. A cyber attacker is focusing on a customer relationship management (CRM) application. In this case, the CRM app has a simple web form that a sales person can use to input a 5-digit customer ID to look up their latest purchases. The cyber attacker finds the form and starts inputting various SQL commands to see if the form field is vulnerable to the much-dreaded (and still very common) SQL injection (SQLi) attack.

AT THIS POINT, THERE ARE TWO POSSIBILITIES:

- The developer wrote the web application in such a way that data plugged into that form is validated and thus hardened against SQLi
- The form field allows pretty much any input, and the cyber criminal can send SQL commands from the web application directly into the database.

If the second of these two scenarios is true, the cyber attacker is taking advantage of code that essentially trusts all input and does not attempt to validate that input. Commands sent by the cyber criminal can be interpreted by the database to perform actions like downloading a table from the database, or maybe even the entire database. Or, if the cyber attacker is having a bad day and just feels like wrecking your day, commands can be issued to delete or overwrite the data altogether. Either way, if your business is using a vulnerable form like this, someone will soon be explaining to your customers how their data got leaked or destroyed. That will not be good for business.

THE GOOD NEWS IS THAT THE FIRST SCENARIO OF A HARDENED AND SECURE WEB APPLICATION IS VERY ACHIEVABLE.

Web application attacks like SQLi and others are very well documented. Methods of mitigation (i.e. sanitization, parameterization, whitelisting, and others) are easy to learn and can be found all over the web.

One such source for learning about how to code more securely is the Open Web Application Security Project (OWASP). Their Top 10 list of the most critical web application security risks is the de facto web application security standard for companies and regulating bodies around the world. Each of the Top 10 entries contains various methods for blocking attacks.

02 Vulnerability Scanning and Penetration Testing

WHILE SECURE CODING PRACTICES ARE VITAL FOR WEB APPLICATION SECURITY, THERE WILL INVARIABLY BE AREAS OF THE CODE THAT STILL HAVE SECURITY ISSUES.

The best way to find those is through testing the web application both statically (non-running) and dynamically (running or runtime) so that developers can fix the issues during the development lifecycle. Let's discuss the strengths and weaknesses of these two modes, as well as explore the variations of each.

STATIC TESTING YOUR WEB APPLICATIONS

First, let's look at static application security testing, otherwise known as SAST. When testing an application statically, you are doing so from the perspective of the developer of the application. This means you have full access to the source code and documentation of the application, effectively giving you full knowledge of the how the application works, how data flows through the application, and where there might be a security flaw in the code (that is why this method of testing is also called white-box or transparent box testing).

Testing statically is important in the software development lifecycle (SDLC) because it catches flaws before the code has been fully written and deployed into a running environment. That translates to cost savings due to bug fixing becoming more expensive the further you are in development. But how you statically test the code is really the important question, and the answer is dependent on your needs.

The most accurate method of static testing is to have a web application security expert with knowledge of secure coding practices run through the code manually (this is usually called a code review) to look for potential problems. Getting the expertise to do that is often difficult and expensive. Also, while this method of review is often done in development shops as code is written, performing a full code review to look for security issues is hardly feasible with modern applications that often run into the millions of lines of code. Essentially, this method of static review is just not very efficient for the DevOps-y world we live in.

In order to speed that process up, many companies use automated static code analyzers. These tools analyze the static code to determine if a security flaw exists and (typically) create a report that shows where issues exist down to the line of code. The higher-end tools also give advice on how to remediate the flaws and integrate into the development lifecycle (automated scanning and bug tracking take a huge load off of developers and web application security teams).

Automated analyzers do have a down side: their propensity for false positives. Because of this, the results require review by a person highly trained in secure coding and web application security in general. So, while the tool does a lot of the heavy lifting, the same expertise is needed to review results as is needed to do a manual review. It's not the same level of effort as in code review, but it is highly recommended to perform the review of the results so that developers don't have to chase down false positives themselves and take away cycles from actual development.

DYNAMIC TESTING YOUR WEB APPLICATIONS

Dynamic application security testing (DAST) is an entirely different animal than SAST. Where SAST is called white box testing because of the tester has the same view as the developer, dynamic testing is called black box testing because the tester has the view of the cyber attacker.

This means **two things**: 1) the web application is in its running state; and 2) there is typically little to no detail of the inner workings of the web application available to the tester (it is possible to glean details of the web app via various nefarious practices, but that is beyond the scope of this discussion).

These distinctions between static and dynamic testing are important. If a tester finds a flaw with a web application in its running state, it is typically much easier to verify than with static testing. The tester simply has to run through the same steps again to make sure the flaw appears again. Essentially, the false positive rate is lower (especially when using automated tools) in dynamic testing than with static testing. So, let's look at some of the methods of dynamically testing web applications.

Similar to the manual effort involved in the static code review mentioned above, dynamic testing can be performed using a manual methodology by an individual. This involves someone with expertise in web application security running through the application page by page looking for potential flaws in web forms, authentication mechanisms, session management, and others. This type of testing, while more accurate, can be time-consuming and expensive. Modern development timelines may cause this level of testing to become untenable to an organization.

Again, in a similar fashion as static testing, automated dynamic web application analyzers were created to cut down the time needed for a full dynamic test of an app. Typically, the analyzer is setup and ran by an individual (who does not need to be as proficient in dynamic testing) to perform and automated analysis of the website structure (called "spidering") to find all the potential vulnerable elements of the web app. These are recorded and tested according to the configuration of the analyzer, and the results are provided when testing is completed (these results also need to be reviewed by an expert).

OUTSOURCING SAST AND DAST

SAST and DAST testing obviously require expertise. Organizations often don't have the resources and/or inclination to bring in that level of expertise, which means that outsourcing comes into play. There are a lot of companies that perform application testing. Some focus on the more manual efforts like code review or manual dynamic testing. These are typically firms or business groups in a larger company that are highly specialized. There are other companies that use a mix of manual and automated testing, and others that almost strictly perform automated assessments.

Another more recent development in the dynamic testing world is the bug bounty program. A bug bounty program is essentially a crowdsourcing effort for finding software bugs (in this case, the bugs are focused on security). Individuals participate in the program by signing up as a tester and performing dynamic testing against the web application. If that individual finds and reports a unique and qualified flaw, the company running the program gives that person an award (typically monetary). There are also companies that plan and run the bug bounty for you. While somewhat controversial, bug bounties have the benefit of a high number of individuals with potentially different techniques testing a web application.

The approach your company takes to testing its web applications depend highly on its business goals and any resource restrictions that may be in place. But no matter what path is taken, it is highly recommended that both static and dynamic testing be performed on your web applications. Both have their strengths as weaknesses, but they can bolster each other when both are used in an effective web application security testing program.

03 Layered Defense

A GOOD SECOND LINE OF DEFENSE FOR PROTECTING YOUR WEB APPS IS THE DEPLOYMENT AND USE OF TECHNOLOGIES DESIGNED TO DETECT MALICIOUS TRAFFIC GOING TO THE APPLICATION SO AN ACTION CAN BE TAKEN TO MITIGATE.

Intrusion detection systems (IDS) and a web application firewalls (WAF) are two traditional technologies that are used to fulfill this role. Using both together can provide a strong layered defense against web applications attacks.

Because we're specifically talking about web application attacks, let's start this section with the web application firewall. WAFs are designed to work at layer 7 (application layer) of the OSI model, which means that they can inspect the actual HTTP request coming into the web application. This is an ideal place to detect anomalous or malformed HTTP requests, and a good WAF will have a default policy set that contains a signature set for detecting that malicious traffic.

Another advantage that web application firewalls have at layer 7 is their blocking ability. If the web application firewall is deployed inline, bad traffic can be stopped before ever reaching the web application, thus preventing the attack from having any effect. Another advantage of blocking is the ability to apply a "virtual patch" for flaws found in an application. Essentially, a temporary signature can be created on the WAF that is specific to the flaw, which allows time for developers to create a more permanent fix. This is often also used for legacy applications that are no longer being maintained.

One warning about blocking with a web application firewall is that it can also block legitimate traffic (false positive) if not tuned correctly. Tuning is the process of putting the web application firewall into a non-blocking state, which gives it time to learn the traffic patterns coming into the web app. A good WAF will allow the user to tweak and/or disable the signature that are causing false positive. Some web application firewalls, like Alert Logic Web Security Manager, will even offer the ability to drill down on every parameter, write your own signatures, and control every input, thus reducing the possibility of false positives.

In contrast to a web application firewall, an intrusion detection system does not operate at the application layer and cannot perform blocking of traffic. However, this doesn't mean that an IDS isn't an effective tool against web application attacks. It just means that it is used in a different manner.

Instead of working at layer 7 like the web application firewall, the intrusion detection system works at layers 3 and 4 (network and transport) of the OSI model. While the intrusion detection system can't see the request in the same way as the web application firewall, it can still detect layer 7 attacks at those lower levels. The IDS signatures (using regular expressions and other methods) just have to be written specifically to find those attacks as they come across at those lower levels.

Because an intrusion detection system does not block attacks, the main purpose for an IDS is to detect the attack and alert (typically) an internal security team, which allows that team to take some remediation action. Detection allows decisions to be made based on policy, which can give the security team flexibility of action during their incident response phase. This kind of measured response can cut down on panic and the potential for blocking of legitimate traffic.

TO SUMMARIZE, HERE ARE THREE QUICK POINTS TO THINK ABOUT WITH A DUAL WAF/IDS DEPLOYMENT:

1

Deployment:

How you deploy your WAF/IDS combo depends entirely on your infrastructure and business goals. However, we often see the pair deployed with the web application firewall closest to the ingress point of traffic into the web application. This allows it to weed out most cyber attacks because of its inherent ability to see layer 7 traffic. Anything that gets past the web application firewall can be then detected by the intrusion detection system, which can send an alert to the security team. This also serves the dual purpose of lowering the number of alerts that the IDS creates while also detecting other compromising actions by the cyber attacker at layers 3 and 4 (remote exploits, SSH login attempts, etc.).

2

Layers:

While a quality web application firewall and intrusion detection system combination that is well-tuned and maintained is very effective against web application attacks, a determined attacker who is very familiar with how to craft application layer attacks can often bypass these protections. Don't fall prey to the notion that all you need to do is deploy a WAF and IDS. Make sure you emphasize a secure software development lifecycle.

3

Traffic:

A huge benefit to cloud security technologies like a web application firewall and intrusion detection system is their ability to capture the network traffic in your environment, even if that traffic was not seen as malicious when it traversed your network. That traffic can be centrally stored as data and analyzed to find information on any successful attacks that were not detected, which can help you avoid future cyber attacks.

04 Cloud Security Blocking and Tackling

SECURE CODING, VULNERABILITY SCANNING AND A LAYERED DEFENSE PROVIDE PROTECTION AGAINST MOST EXTERNAL CYBER THREATS.

There are a couple housekeeping things you need to handle as well to guard against unauthorized or inappropriate access to web applications and data, and to ensure known vulnerabilities in your servers and web applications are mitigated.

Now let's talk about some basic blocking and tackling. There are plenty of basic tasks that need to be performed for a cloud security program. Two of the most important are identity and access management (IAM) and patching, both of which often don't get enough attention - and sometimes get completely overlooked – particularly in cloud environments.

Let's start with identity and access management. IAM is the system used to securely control user access to your cloud resources (i.e. apps, databases, etc.). The proper implementation of IAM greatly reduces your web application security risk by controlling the authentication and authorization of your users. Essentially, an effective IAM strategy will limit resource access to the right individuals at the right time for the right reasons.

Two very important considerations for building that IAM strategy: least privilege and managing permissions within groups. These will minimize who can access and make changes to your web applications and databases. **Both are described below**:

Least privilege is the granting of only those permissions to users that are required to perform their tasks. When thinking through how to implement IAM for your environment, always establish what tasks your users need to perform, and then build policies for them that limit the user performance of only those tasks. This will greatly reduce risks of breach through issues such as credential theft.

Managing permissions with groups is very important in battling "permission creep" (the steady buildup of permissions as users move around an organization). Assigning the proper permissions (i.e. administrator, database admin, developer, etc.) to a group and then adding users to those groups stops the need to assign permissions directly to the user. Adding and removing users from groups is considerably less difficult and inherently more secure than trying to add and remove permissions individually.

PATCHING IS ANOTHER AREA THAT SHOULD BE A BASIC BUILDING BLOCK IN A SECURITY FRAMEWORK.

Yet the recent WannaCry and PetrWrap malware outbreaks provide evidence that patching is very often neglected. Both took advantage of a flaw in the SMB protocol in Windows that had been patched for months.

Fortunately, patch management is a mature market. You can purchase high-quality patch management systems that will work in the cloud (see the quick notes below for some caveats). These will help you create a regular patch cycle so you can keep up to date with patches as they come out, as well as quickly and easily apply emergency patches if a zero-day exploit is discovered.

Another consideration for patching in the cloud is the use of infrastructure automation solutions like Chef or Puppet. These kinds of tools help automate patching for running instances of Linux or Windows servers, or they can also be used to keep your machine images up to date with the latest packages so that server instances in your environment can simply be refreshed regularly. Infrastructure automation tools are very powerful and offer flexibility so you approach patching in a way that best fits your environment.

Finally, Amazon Web Services and Azure both have fairly new solutions for helping you patch your systems. Both Amazon EC2 Systems Manager and Azure's Operations Management Suite (OMS) are agent-based, meaning an agent is installed on the machines for patches to be installed. If you are using one or both of these IaaS providers, their solutions might be something to investigate further.

TWO QUICK NOTES ON PATCHING:

- The cloud delivery model you are using has high relevance to patching. We assume you are deploying your web apps in an IaaS (Infrastructure as a Service) model, where you have full control over your environment. SaaS (Software-as-a-Service) and PaaS (Platform-as-a-Service) models are different animals and take different approaches.
- It is important to ensure that you test patches before you apply them, especially when patching critical systems. Though a healthy amount of testing goes into the development of patches that come from the developer, you can never be too careful. Be sure to build that testing into your patching cycle and emergency patching procedures.

05 Intelligent Log Analysis

ALL DATA HAS VALUE. SOMETIMES YOU JUST DON'T KNOW WHAT THAT VALUE IS YET.

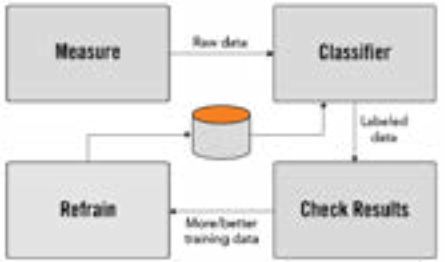
The first step is capturing and storing the data, but the challenging part is sifting through it all effectively and efficiently in order to extract relevant or critical insights.

The various elements of your infrastructure produce logs - access, system, etc. – that contain all sorts of data about what is happening in your environment. Your security technologies (like intrusion detection systems (IDS) and web application firewalls (WAF) that we mentioned earlier) also produce data about your environment by capturing traffic and producing logs based on that traffic. Analyzing all that high-value data in an intelligent and consistent fashion can lead to information about successful cyber attacks against your environment.

Pulling all that data into the same central spot (log management tools are a good solution here) so that you have a single pane of glass for the next phase: analysis of all that data. For instance, your intrusion detection system may have had a signature fire on malicious traffic, but the attacked server was not vulnerable to that cyber attack. You can probably conclude that the attack failed in that scenario. However, if the server logs show a successful login and a large amount of traffic going out to the Internet in a relatively close period after the attack, you likely have an issue that needs to be raised to the incident response team.

The cyber attack in the example scenario above seems like is a simple issue to find. However, finding all the pieces to correlate that cyber attack when you have terabytes or more of data is not simple at all. The mounds of data that can be captured can make it impossible to sort through when only being done by humans. Even small organizations can produce large amounts of traffic data and logs, especially in today's cloud-enabled world where many web applications can be deployed quickly.

One way of sorting through that amount of data is by creating a big data group within your organization and bring in an analytics engine to evaluate and identify attacks. Many organizations today are also leaning on machine learning to help refine that data even more. The key requirements to making any kind of analytics/machine learning system work well are consistent measurement techniques, having good training data for the engine, and human verification of the results. Functions like making sure the data is consistent, preprocessing the data, building models to train the engine, determining whether the results are valid, etc. will ensure that the results coming from the analysis can be trusted.



But doing all that log analysis is costly and difficult. Analytics and machine learning don't help much if there aren't people behind the scenes like data scientists and other experts who can make sure everything is working properly. While a large enterprise may be able to staff that kind of effort, most companies don't have those kinds of resources.

Outsourcing that effort is an option that many organizations take to sort through their big data. But what if your company's primary business is retail? You might have a very large data set on pricing, brands, store location, marketing, etc. But it is unlikely that you have much data on cyber attacks from outside your organization. With a limited data set, it's hard to meet that key point of having good data to train the engine as mentioned above.

EMPLOYING A MANAGED SECURITY SERVICE PROVIDER (MSSP) TO HELP MIGHT BE VALUABLE BECAUSE THEY SHOULD HAVE A PRESENCE IN A LARGE NUMBER OF ORGANIZATIONS, SO THEIR DATA SET CAN BE VERY LARGE.

However, typical MSSPs manage many different brands of cloud security appliances (i.e. intrusion detection system and web application firewall), which means that data coming from all those devices – even from very similar cyber attack types – will be different. It is very difficult – if not impossible – to satisfy the key requirement of consistency in measuring techniques if the data is not also consistent.

The best way of making sense of all that data is by outsourcing the analytics/machine learning to an organization that:

- Aggregates a large amount of useful, consistent training data from multiple environments that all use the same infrastructure; and
- Uses experts to build high quality log analysis models and verify results. This meshes really well with what we're doing with analytics and machine learning here at Alert Logic.

At Alert Logic®, we collect in the neighborhood of 20 petabytes of data a year from our customers (and as we grow, so does that dataset). The alert data we pull from that huge dataset comes entirely from the infrastructure we provide. This means that we have a collection of consistent, high quality, and high volume cyber security data. We have data scientists to curate and label subsets of data needed for training the machine learning algorithm, as well as cloud security and web applications domain experts and SOC analysts to guide the data scientists. This creates a feedback loop that evaluates the algorithms performance and continually improves the results, while our production team converts the data scientist results into a scalable, high quality detection implementation. In short, Alert Logic is the only solution available that integrates all required elements that can make machine learning add tangible security value to customers.

STRONGER WEB APPLICATION SECURITY

The web is not going away. Business via the web will continue to grow, especially with the cloud creating faster and more agile ways of building web applications and storing data. Which means that cloud security is critical to keep those web applications and databases protected. The first line of defense against web application attacks like SQL injection is making sure your application is coded securely. Training developers to think securely – specifically that all input coming into the web application is untrusted input will go a long way in stopping cyber attacks.

But the burden of web application security should not rest entirely on the developers. The business should also invest in cyber attack detection technologies like web application firewalls and intrusion detection systems. These types of devices allow the security team to see what kinds of cyber attacks are happening so they can decide on the response based on their IT security policy.

The basic blocking and tackling of cloud security should also be given enough attention. Creating, implementing, and maintaining a strong identity and access management strategy is very important when controlling access to your resources in the cloud. Concepts like least privilege and managing permissions with groups can go a long way in limiting resource access to the right individuals at the right time for the right reasons. Patching your systems is another one of those basic tasks that can get missed or setup inadequately. Making sure you chose the right system for patching your systems depends a lot on your environment. Luckily, there are a lot of choices.

And finally, the intelligent analysis of the data coming from your environment can help analyze successful cyber attacks, find attacks that might have been missed, and refine detection of those cyber attacks for the future.

But to do that, you have to figure out how to go through all that data, which is substantial even for small organizations. Analytics and machine learning are very promising in their ability to help sort through the quagmire, but that entails building a team of all kinds of different experts, as well as actually bringing in more data from outside your organization to enrich the mounds of data you already have. It's a daunting task, but it is one with which Alert Logic is uniquely qualified and positioned to help.

The overall lesson here is that while there are a lot of things that need to be done to protect your SQL-based apps in the cloud, there are a few things that can help tremendously right out of the gate. Don't be intimidated by it all. Keep doing business, and keep growing in the cloud.

ABOUT ALERT LOGIC®

Alert Logic® Security-as-a-Service solutions integrate cloud-based software, analytics and 24x7 expert services to assess, detect, and block workload threats and help you comply with mandates like PCI, HIPAA and SOX COBIT. With more than 4,000 customers worldwide, we focus on threats most relevant to AWS hosted applications by defending your full web application and infrastructure stack, including hard-to-detect web application attacks such as SQL injection, path traversal and cross-site scripting. Integrated expert services for detection, blocking, and compliance augment in-house security and empower cloud and application professionals. With native API integration and templates for AWS and DevOps tools, Alert Logic solutions provide agile security that scales.

TO LEARN MORE, VISIT HERE:

https://www.alertlogic.com/cd