NeoPrime, LLC

Version 1.01

September 12, 2017

Whitepaper on protecting websites on AWS while minimizing costs.

This document covers a real-world customer problem and the solution that was implemented using AWS services and tools. To see more articles, visit our web site: https://www.neoprime.io/articles.

Table of Contents

Summary

A customer contacted me that had a WordPress website hosted on AWS on a Windows instance. Their problem was that periodically their website would become very slow and sometimes crash.

A review indicated that their website was periodically under attack. Normal traffic was a few hundred visitors per day, but when attacked millions of hits per day. This caused the website to slow down with response times for the homepage exceeding 9 seconds. The website crashed or stopped working several times per month. The customer had no tools in place to notify them of website issues.

After reviewing the customers situation, we decided to implement CloudFront and S3 for a backup website and Application Load Balancer with WAF to protect the web site. Additional AWS services were selected to provide more management and alerting capability.

Problem

- Attackers were using brute force methods to login to the server.
- Attackers were site scanning hitting invalid URLs looking for vulnerabilities.
- The website did not have a software firewall.
- No user or password policy was in place.
- Multiple employees would use the same login.
- The increased bad traffic forced the customer to use larger and larger EC2 instances which increased their monthly costs.
- Numerous web server logfiles were being created with mostly attacker traffic. This required adding additional storage to the EC2 instance while a solution was deployed.

Solution Overview

- Backup website.
- Install a software firewall.
- Move the domain to Route 53.
- Create a static website on S3.
- Create a CloudFront distribution for the S3 static website.
- Configure Route 53 for failover.
- Create a Bastion Host.
- Create a new website in a VPC Private Subnet
- Create the Application Load Balancer.
- Add Web Application Firewall (Amazon WAF).
- Create a "What is my IP" page on the website written in PHP.
- Create Access Keys with a WAF policy to modify the WAF Whitelist rule.
- Create a PHP program to update the WAF Whitelist rule from a desktop.
- Create a website page to update the WAF Whitelist rule.
- Miscellaneous.

AWS Services and Tools

- AWS ALB (Application Load Balancer)
- AWS AMI (Amazon Machine Images)
- AWS Certificate Manager

- AWS CLI (Command Line Interface Program)
- AWS CloudFormation
- AWS CloudFront
- AWS CloudWatch
- AWS EC2
- AWS IAM
- AWS Lambda
- AWS Route 53
- AWS S3
- AWS WAF (Web Application Firewall)
- JavaScript for Lambda functions
- PHP for tools

Solution Details

The solution was implemented in stages.

Step 1: Backup website

We created an Amazon AMI of the website. During the re-architecture, we also created additional AMIs for key steps to allow quick fallback.

Step 2: Install a software firewall

We selected the Wordfence firewall to provide basic protection of the WordPress core. This is a very good firewall for WordPress and provided many features including the ability to lock out brute force password attacks.

Cost: $99.00 per year.

Step 3: Move the domain to Route 53

Amazon's Route 53 offers many features for managing domain names. Our goal was to use Amazon S3 as a failover site during site maintenance and to help with site crashes while the next steps were implemented. This step took a week for the domain to transfer. While waiting we continued with the next steps.

Cost: $12.00 per year. The previous registrar was charging $35.00 per year so this step saved the customer money.

Step 4: Create a static website on S3

An S3 website was created. A static copy of the website's homepage was created and uploaded to S3. Under the top banner graphic, a red message is displayed that the website is under maintenance and will be back up soon.

Cost: Estimated to be $1.00 per month for both S3 and CloudFront.

Step 5: Create a CloudFront distribution for the S3 static website

CloudFront was implemented to provide the frontend for the static website and provide SSL. Amazon Certificate Manager was used to create the SSL certificate for the domain.

Cost: Estimated to be $1.00 per month for both S3 and CloudFront.

Step 6: Configure Route 53 for failover

Once we had the S3 static website configured, we changed the Route 53 configuration:

- Created a Health Check for the web server.
- Created a CloudWatch Alarm sending a SNS notification when the Health Check fails.
- Modified the Record Sets for the domain to use Failover with Health Checks.
- Added a new record pointing to the S3 CloudFront distribution for the secondary record. Now if the website becomes unresponsive, Route 53 will switch over to the static website. This is better than a site not found error for the customer.

Cost: $0.00.

Step 7: Create a Bastion Host

A new EC2 instance was created with OpenVPN in the same VPC as the website but in a public subnet. OpenVPN client software was installed on each employee's desktop that required administrator access to the EC2 instance. Note: website admins still access WordPress thru the public URL and do not need OpenVPN access. Batch scrips using the AWS CLI were created to start and stop this instance to minimize cost, but more importantly to shut down the bastion host so that it could not be attacked.

Cost: Estimated to be less than $5.00 per month as this instance is shutdown when not required. This EC2 instance is a T2 micro.

Step 8: Create a new website in a VPC Private Subnet

We decided to use the Amazon Application Load Balancer to offload SSL and provide better protection from denial of service attacks. We also decided that the website should be in a private subnet to reduce the attack surface. While these steps were completing the original website was still up and running.

Cost: $50.00 per month for the EC2 instance.

Step 9: Create the Application Load Balancer

Even though the planned configuration would only have one EC2 instance, a load balancer was selected to provide the following benefits:

- SSL offload. This would reduce the CPU load on the website.
- Block ports. Only ports 80 and 443 are accepted by the ALB and only port 80 is passed thru to the website. This would stop all traffic common attack points (SSH and RDP).
- Allow the web server to be in a private subnet. This would now require a bastion host.

Cost: $50.00 per month for the ALB.

Step 10: Add Web Application Firewall (Amazon WAF)

We selected the Amazon WAF to provide several features. The WAF sits in front of the ALB.

- Auto Block Rules. Using Lambda, three published IP lists of known bad actors are loaded into WAF. These IP addresses are blacklisted.
- Cross Site Scripting. This rule blocks XSS attacks.
- SQL Inject. This rules blocks SQL injection attacks.
- URL Blocking. This rule blocks access to the website login and management pages.

- URL Whitelisting. This rule allows specific IP addresses to access website management and login pages.
- Honeypot. Bad bots will scan the website for a huge number of invalid URLs and ignore robots.txt directives. When the bad bot scans hidden URLs, they are sent to a honeypot (Lambda function) that adds their IP address to the Bad Bot blacklist. The Bad Bot is blocked from further website access.

Cost: $13.00 per month.

Step 11: Create a "What is my IP" page on the website written in PHP

Since we created a WAF blocking rule for all login pages, we needed a method to determine the IP address for employees who work from home, etc. This page returns their current public IP address so that it can be added to the WAF Whitelisting rule.

The company's static IP was manually added to the WAF Whitelist rule.

Step 12: Create Access Keys with a WAF policy to modify the WAF Whitelist rule

Using IAM, a new user was created with a policy only allowing WAF rule modification.

Step 13: Create a PHP program to update the WAF Whitelist rule from a desktop

Next a small PHP program was written that could call the "What is my IP" page and add the user's public IP address to the WAF Whitelist rule. This program, along with PHP was installed on key employee's laptops. A windows batch script was written with a link on the desktop. All an employee had to do was double-click the icon and their public IP address would be added to the WAF Whitelist.

Step 14: Create a website page to update the WAF Whitelist rule

A website page was created in PHP to also allow an IP address to be manually added to the WAF. This way an admin at the office could easily add an IP address to the WAF without needing to use the AWS console and more importantly learn how to configure and modify the WAF.

Step 15: Miscellaneous

Additional configuration changes were made such as exporting the Apache logfiles to CloudWatch, creating alarms based upon CPU usage, etc. We also configure SNS notifications for key CloudWatch events.

Summary

During serious attacks, which happened at least once per month lasting two to five days, the website would be hit over 1.5 million times per day from bad actors.

After this deployment, the combination of WAF rules combined with the ALB blocked the attackers from reaching the web site.

- The number of attacks dropped to zero for four consecutive weeks.
- Average load time went from over 9 seconds to under 1 second.
- The website did not crash during the past four weeks.
- The logfiles are smaller and less numerous.

Benefits

This implementation has the following benefits:

- Resistant to Denial of Service attacks. Amazon WAF and ALB will take the brunt of the attacks shielding the website.
- Resistant to brute force attacks. The WAF rules block the most common attacks and provide protection to website management and login pages. Whitelisting rules only allow known IP address to access management and login pages on the web site.
- Reduced attack surface. The website is in a private subnet that is not reachable from the public Internet. The significantly reduces the attack surface.
- Reduced EC2 instance size. SSL encryption and decryption are offloaded to the ALB. Attackers are blocked by the WAF and ALB reducing CPU usage. The Wordfence firewall has significantly less traffic to monitor.

Weakness

- The WordPress EC2 instances are stateful. This means that the database is running on the same instance as the website. The WordPress setup should be separated from the database. Solution: add an Amazon RDS instance and move the MySQL database to RDS.
- No Fault Tolerance. Since only a single EC2 instance is running, the website cannot scale with traffic growth. Also, if an availability zone fails, there is no automated recovery mechanism in place. Solution: add an ASG (Auto Scaling Group) configured for the ALB. This step requires implementing RDS to separate the database from the EC2 instances.
- Failover is to an S3 website.

Appendix A: Changes required for WordPress

Two changes are required when using WordPress behind a load balancer:

1) Since we have enabled SSL offload at the load balancer, WordPress only receives HTTP traffic. We set a flag that fools WordPress into believing that it receives HTTPS traffic.
2) WordPress sees the load balancer IP address instead of the client IP address. Amazon's load balancer forwards the client IP address in the header "HTTP_X_FORWARDED_FOR".

Near the bottom of the WordPress file "wp-config.php" add the following lines:

```
if (strpos($_SERVER['HTTP_X_FORWARDED_PROTO'], 'https') !== false)
{
        $_SERVER['HTTPS'] = 'on';
}

if ( isset($_SERVER['HTTP_X_FORWARDED_FOR']) &&
!empty($_SERVER['HTTP_X_FORWARDED_FOR']) )
{
        $ips = explode(',', $_SERVER['HTTP_X_FORWARDED_FOR']);

        $_SERVER['REMOTE_ADDR'] = trim($ips[0]);
}
elseif ( isset($_SERVER['HTTP_X_REAL_IP']) && !empty($_SERVER['HTTP_X_REAL_IP']) )
{
        $_SERVER['REMOTE_ADDR'] = $_SERVER['HTTP_X_REAL_IP'];
} elseif ( isset($_SERVER['HTTP_CLIENT_IP']) && !empty($_SERVER['HTTP_CLIENT_IP']) )
{
        $_SERVER['REMOTE_ADDR'] = $_SERVER['HTTP_CLIENT_IP'];
}
```

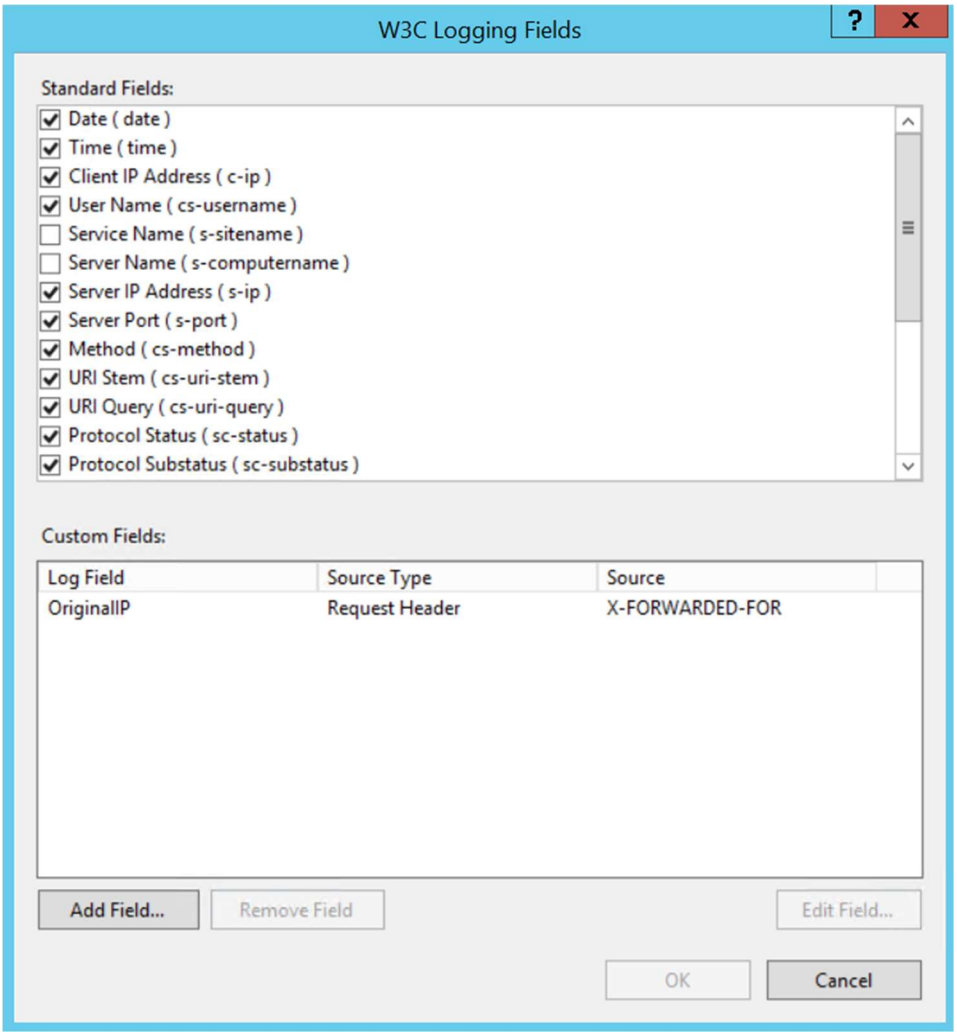
Appendix B: Changes required for IIS logfiles

After placing an EC2 instance behind an elastic load balancer, the IIS logfiles "Client IP Address" (c-ip) will report the IP address of the load balancer instead of the visitor's IP address.

Steps to correct this problem:

- Start Internet Information Services (IIS) Manager.
- In the left pane, select your web site.
- In the middle panel double-click "Logging".
- In the Logging screen, click "Select Fields".
- Click "Add Field".
- Enter "OriginalIP" for the "Field Name".
- Select "Request Header" for the "Source Type".
- Enter "X-FORWARDED-FOR" for the "Source".

Example screenshot showing the new custom field:



Appendix C: Create OpenVPN Start and Stop Batch Files

Using the Amazon Console, look up the instance ID for the OpenVPN EC2 Instance. Modify the batch files with the correct ID. The pause in each script is necessary for Windows desktop links so that the Command Prompt window stays open to see the result.

START_VPN.BAT

```
aws ec2 start-instances --instance-ids <REPLACE_WITH_YOUR_INSTANCE_ID>
pause
```

STOP_VPN.BAT

```
aws ec2 stop-instances --instance-ids <REPLACE_WITH_YOUR_INSTANCE_ID>
pause
```