Powerful protection for web applications

Dr. Götz Güttich

With Airlock Web Application Firewall, Ergon Informatik offers a powerful solution for securing web applications. As a general rule, these applications don't just need protection from hacking and security breaches, we also need to ensure that only authorised users have access to the data within them. This is why the manufacturer combines its Web Application Firewall with the 'Airlock Identity and Access Management' authentication solution. Working together, these tools secure the data traffic between users and applications and also ensure that there is absolute clarity as to which users have access to which applications and functions. This also means that a central single sign-on infrastructure can be set up. We took a close look at both products in the test lab.

Due to the wide functional scope of both solutions tested, we had to divide the test into two sections. This section deals with the Web Application Firewall (WAF); its functions and administration. Section two looks at the Identity and Access Management (IAM) solution.

Airlock WAF

The Airlock WAF works as a reverse proxy and terminates HTTP(S) connections. It is controlled via a central management interface, providing a central point for implementing application access policies. It also provides a large number of powerful filter functions, including ICAP content filtering and SOAP/XM, AMF and JSON filters. One of the top features of the Airlock WAF 7 is API security, with protection of SOAP and REST web services. Other functions include policy learning, dynamic white listing, smart form protection, cookie protection and URL encryption.

Many functions can also be incorporated in existing security infrastructures: Malware scanners from third-party providers can be integrated via ICAP, HSM devices can be added and WAF notifications can be sent to SIEM installations. Load balancing and failover features can also be provided. Ergon Informatik also has an appliance for companies wanting to implement their WAF in hardware form.

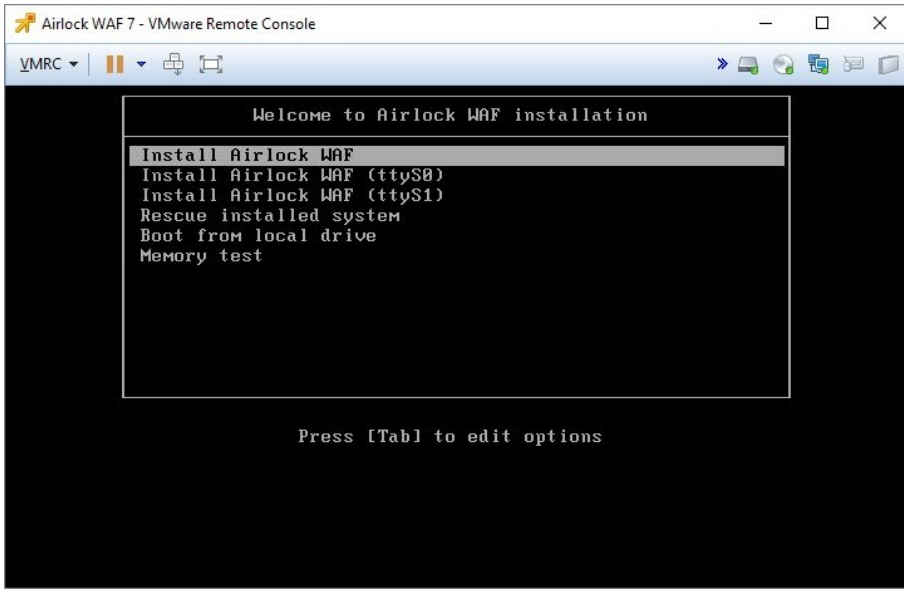
Functionality

In operating mode the WAF works between the regular company firewall and the applications. Production environments, therefore, need interfaces in order to maintain clear-cut separation between incoming and outgoing data traffic. This is not essential in test environments. Data transfer protection configuration happens via mapping. This is set up using a clear page within the management tool, where the virtual hosts, that represent the protected applications to the outside, are located on the one side and the applications to be protected are located on the other, in the DMZ. Only ports 80 and 443 are open to the outside.

To secure the application, with the mouse, users drag lines between the virtual host, the mapping and the application requiring protection, thus configuring the data transfer routes. These connections can be cut or diverted at any time as the need arises. For security reasons, when new mapping is installed, all associated rules are active, initially, to prevent any unwanted data transfers. To carry mapping over into regular operating mode, administrators need to adapt the configuration in the next step to ensure

that the desired information can pass through.

Threat handling establishes how the system deals with policy infringements. For example, if the WAF establishes that a user is attempting to tamper with the data in a web form, the solution can either simply generate a log entry



Installing the WAF should not present IT administrators with any insurmountable obstacles

to block the request or end the session completely.

'Deny rules' are a core component of the Airlock WAF security concept. They create a negative security model, operating as blacklists. In the configuration interface, every deny-rule entry, i.e. every line, represents a group of rules. However, each group addresses a specific hacking attempt. The 'exceptions' entry encompasses acquired and manually created rules.

When the administrators open a group in the deny rule configuration they will see the various filters. They have different security levels. As strict filters generate more false positives than filters that are not, it makes sense for operational purposes only to set the security level very high where absolutely necessary. Security levels are therefore dependent on the application requiring protection and individual data protection requirements. It is also important to know whether hacking can come anonymously from the Internet, on a login page, for example, or if there has already been a login and the user is known to the system. The manufacturer has already pre-defined a large number of rule groups for all types of hacking. If the need arises, users can add their own custom rules at any time.

The test

To run the test we installed Airlock WAF in our test lab, linked it up to the IAM solution mentioned before and set both products running. For the purposes of the test we used the WAF to secure access to the web interface of the Paessler PRTG network monitoring solution we use here in the test lab. We also used a test environment provided by the manufacturer with a bookshop as the application needing to be secured in order to take a close look at the functionality of the WAF.

Installation

The Airlock WAF is based on CentOS 7. After downloading the installation ISO file from the manufacturer's website we integrated it as a boot medium in a virtual machine running with Vmware ESXi 6.5 Update 1. The virtual machine had a CPU with a 2.6 GHz cycle frequency and four cores, eight gigabytes of RAM and 60 gigabytes of hard disk space. After the system started from the ISO image we selected the 'Install Airlock WAF' boot menu item. As is usual with Centos, the 'Anaconda' installation manager came up right away. Before the actual installation began we had the option to adapt the configuration to suit the management interface. The system did take a functioning configuration from our DHCP server but it would be sensible in most cases to give the WAF to a dedicated IP address, which we did here. We then specified the destination disk for the installation, which was easy, as our VM only had a virtual hard disk, and accepted the pre-defined partitioning – this can also be adapted as required. Finally, we set the time zone and the root password and defined a user for the configuration tool. The installation then started, and the whole process took around ten minutes in all. When the setup was complete, we imported the latest updates to bring our installation completely up to date.

Initial configuration steps

After running through the setup, we accessed the WAF management tool via the URL: https://{IP adress of the management interface}. We then logged in with

our pre-generated credentials and went to 'System Setup' to upload our licence.

We were then ready to secure access to our first web application. All we needed to do was to go to the management tool in the 'Application Firewall' menu on the 'Reverse Proxy' page and enter the application server with its IP address and a port number as the 'backend server'. If DNS name resolution already exists on the network, you can enter the system name instead of the IP address.

As soon as the 'Reverse Proxy' has been defined, it's time to set up a new virtual host to receive incoming requests. Definition is via a fully qualified domain name and an IP address with network mask. To encrypt data traffic via HTTPS, relevant employees can also add a certificate at this point.

Having done that, we configured the mapping between the virtual host and the backend server. As already mentioned, this takes over the link between the virtual host and the backend application, implementing the security logic. It must include an entry and a backend path. The entry path is the path that users access via their browser and the backend path is the path via which the WAF sends requests. This leaves a lot of configuration options open. So you could have symmetrical mapping, where an entry path takes data to an application or part of one, in order that you can secure whole applications or just certain sub-pages. Alternatively, you can also have asymmetrical mapping, where a virtual host takes different entry paths such as '/Application1', '/Application2' and similar data, which are then sent on to various backend servers with different applications. In the test we first configured symmetrical mapping with the manufacturer's default details. To ensure that data transfers wouldn't be blocked we set the previously mentioned threat handling at 'Log only'. This is a good idea for new mapping, as the relevant employees can then analyse what would happen if the WAF blocking rules were, in fact, active. This prevents unwanted activity and the necessary adaptations can be made.

Finally, we used the mouse to create the said links between the mapping, server and host. Once we had activated our changes on the WAF, the system was ready to go and we were able to connect to the virtual host.

Important functions

Before we look at how the WAF works in practice, let's take another quick look at the product's major features. It's opportune to talk about protection against injection hacking like SQL injection and XSS protection against hacking in the form of forceful browsing. Forceful browsing is when the hacker tries to gain access to protected pages or obtain sensitive information by entering your URL directly into their browser's address list. Unprotected systems provide an opportunity to use this method to gain access to content that would normally only be available after authentication, or acquire detailed information about the configuration of the web server in question. The Airlock WAF offers a URL encryption function for preventing forceful browsing. This means that, in operating mode, encrypted URLs are only valid in the current session when in operation mode. We will look at this feature in more detail later in a use case.

On the other hand, smart form protection secures web application forms against tampering during runtime. The WAF also ensures that user data matches the details on the application's original form. This prevents hackers from adding a fifth parameter when hoping to tamper with an input form that only requires four. This feature also protects preset values from tampering. We will also look at this again in a use case.

DyVE

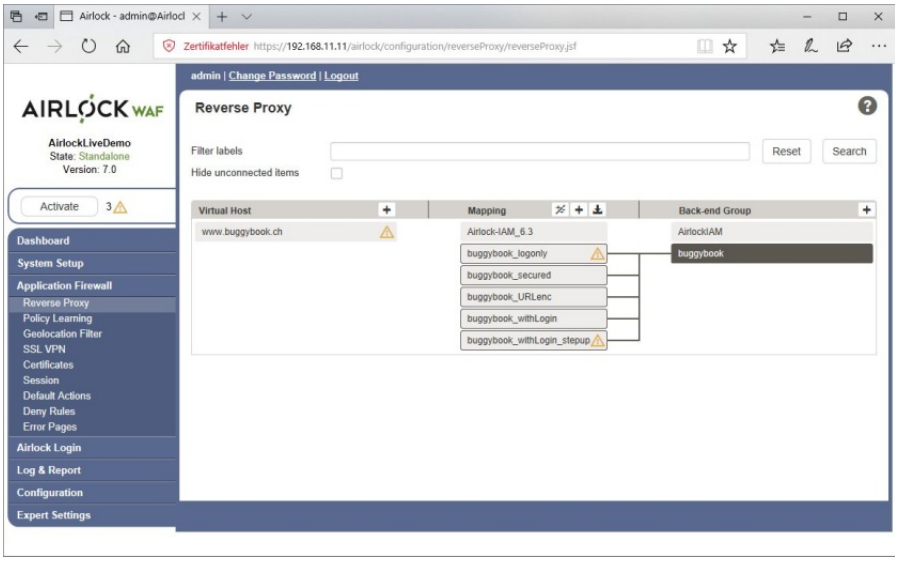
Dynamic Value Endorsement (DyVE) is a function that searches dynamically through JSON objects delivered by backends for values permitted for the current user session. The parameters or JSON attributes of any subsequent requests, such as REST API calls, can then be checked for the use of reliable values. DyVE can also stop hackers changing an account number in a transfer and tampering with online banking. Session fingerprinting notices changes in browsers and IP addresses used. Combined with login information and the like that the system obtains from the IAM solution, it is flagged in operating mode if a login comes from say Frankfurt at 9.00 am and then Bangkok at 9:12 am. In this scenario, the system can be configured so that a second authentication factor, such as a security question, is added automatically for the second login, ensuring that unauthorised users can't gain application access.

Policy Learning
If the policy learning function is activated the WAF checks and analyses data traffic and indicates what has been sent, from where and to which location. It also suggests which rules should be implemented to increase the security level. Administrators can then accept, modify or reject them. This function is useful for adapting the configuration to detailed requirements.

Allow Rules
Allow rules in operating mode determine which requests are permitted. They therefore constitute white lists. Allow rules are used for analysing HTTP requests. Deny rules are used when an allow rule has permitted a request. The Airlock WAF also offers an allow rule learning function which can be used to record all details during a login procedure and converts them into rules that IT managers can then accept.

Working with labels
In practice most companies use a large number of the said mappings – this can exceed 500 in many environments – in their configuration for controlling data transfer between the virtual hosts and the applications that need securing. IT staff have the option of labelling them to obtain a clear view of the mapping and make input configuration easier.

For example, there is the option to label an Exchange or SharePoint input with 'Microsoft' or an Exchange 2016 input with 'test environment' and an Exchange 2013 input with 'product environment'. When an administrator enters the label in the overview search line the system shows them the relevant mapping. The labels can be combined in all sorts of ways, making this a very powerful feature for IT managers as it makes configuration highly transparent. For example, if an IT person wants to change the security level for all Microsoft applications within a company, all they need to do is enter the Microsoft label in the search field and then change the security level once for all entries. We had no problems with this function during the test.



Mapping configuration defines data flow between the virtual host and the application requiring protection

Monitoring
When it comes to reporting and logging, as already mentioned, the Airlock WAF not only manages the transmission of information to SIEM solutions, with log formats JSON and CEF, but also offers a log viewer with filters, and dynamic display of relevant and pre-defined searches. This comes with various dashboards and the option to run your own evaluations. Once in the log viewer, you can also run drill down procedures right down to actual log entries. A map is also included in the monitoring tools. When administrators zoom in on it they will see further details of the regions accessed, telling them, for example, if most of the hacking attempts in the last hour came from a particular country or city.

The Airlock WAF in practice
For the next part of the test we used the demo environment with the bookstore, as previously mentioned. It was called 'BuggyBook' and, as the name suggests, has a large number of different security breaches. We started with a close look at the form protection function. We logged into the unprotected bookstore as users and bought a book. There was a minus in our account. To rectify the minus, we switched to account management and entered a voucher code in 'Voucher' that was worth 50 euros in the bookstore. The online application then showed us a confirmation page with the amount of credit that the voucher would give us and enabled us to add it to our account by clicking 'Confirm'. This worked perfectly.

At the next stage we repeated this procedure but did not click 'Confirm' on the confirmation page

but went to the development tools in our browser and looked for the value in the source code that determined the voucher amount. We used the development tool to change this from 50 to 5,000 euros and then clicked 'Confirm'.

The browser then sent this value to the application. Where it had no protection mechanism for comparing the confirmed value with the original voucher value, it credited our account with 5,000 euros without any problem at all. The 'BuggyBook' test application was therefore in urgent need of the security functions provided by the WAF.

To plug up the security breach revealed in this way without modifying the store application we activated form protection in the WAF that prevents precisely this kind of activity. When we repeated our attempt at fraud with active WAF protection in place, the WAF blocked the transfer of the incorrect parameter and recorded the whole procedure in its log files.

In the next use case we applied the Cross Site Scripting protection function. We also sent a message to another user from our bookstore user account via the internal message function.

When we sent the message, we entered a script in the subject field for selecting the user's session cookie information. When we then logged in as the other user and accessed the message a popup appeared with the user's session information. Scripts are often used for hacking unsecure systems like this. The WAF repels scripting hackers via its deny rules. So, in the next stage, we activated a deny rule configuration that would recognise suspicious patterns in data transfers. The configuration then analysed the transmitted fields and the parameters contained in them. If a deny rule recognises a pattern, for example JavaScript, it blocks the request immediately. This also works for JSON structures.

Once our new configuration was active, we attempted the same hack again. The WAF blocked the request and recorded the activity in the log files.

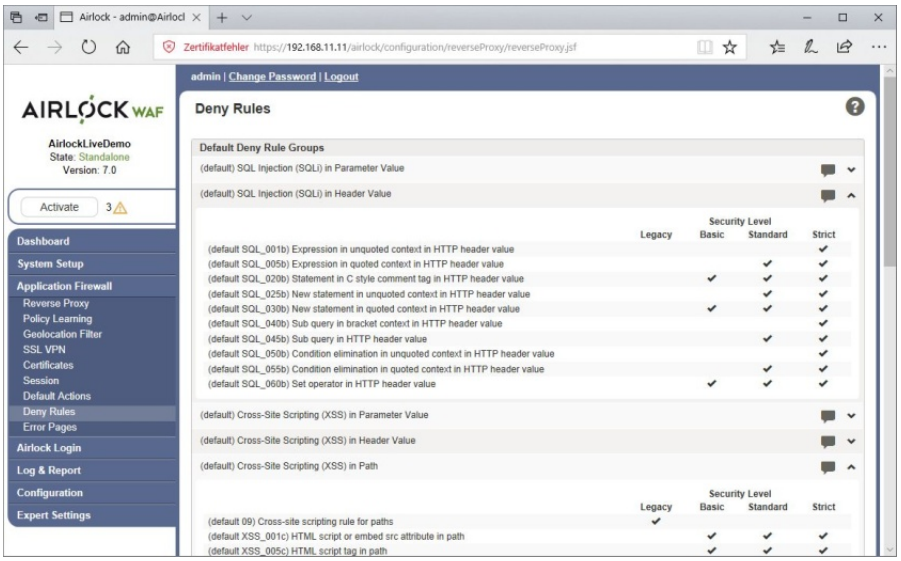
We then attempted a forceful browsing hack via the browser. We entered the URL: "https:// {IP adress of the server} / {Bookstore} / help / index.html? page=../ ../ WEBINF /web.xml;. The browser provided us with various details about the system configuration that we might use for further hacking attempts.

The WAF URL encryption function prevents this type of hacking, encrypting target URLs at session level and ensuring that nobody can access specific targets on the server. After we had activated this feature, the system diverted our attempt at access to the web shop home page, so no damage was done.

Conclusion

We were very impressed with the wide range of functions offered by the Airlock WAF during the test. When configured correctly, it ensures that everything arriving at the application requiring protection has been filtered and all potential threats have been removed.

This means that hackers don't get anything at all from protected applications. In spite of the wide range of functions the management interface was relatively straightforward, enabling security administrators to get to grips with the solution quickly. While running the test we were particularly impressed by the comprehensive reporting and logging, API security, smart form protection and URL encryption. Administrators who have an urgent need to secure their applications from the outside really ought to take a look at the Airlock product.



Individual deny rule filters were assembled in groups. A specific security level can be set for each group.