

Background Article

Dynamic Whitelisting

Web application developers cannot foresee future dangers – but a web application firewall can

Poorly programmed web applications are often cited as being the cause when companies suffer serious problems with their web security. If the security issue had been considered in advance, then no problems would have arisen in the first place. Unfortunately, this argument fails to consider two important points. Firstly: Web applications are extremely dynamic. They are often adapted and provide different user groups with quite different operating options. Secondly: New security leaks are constantly being uncovered in the technologies upon which the web applications are based. This means that a security filter must also be installed that can deal with these challenges –especially outside of the web application.

Web applications provide users with a flexibility that places low demands on their clients and even give companies the facilities to create completely new business models. One example is how home banking has changed the way banks and their customers interact with each other. However, the elements that make these applications so efficient – i.e. that the user relieves the company of certain tasks by, for example, entering data into the database or the CRM system himself – also increase their vulnerability. If every computer everywhere in the world can communicate with the corporate database then at some point, sooner or later, someone somewhere will attempt to manipulate it.

Web application security is already a highly complex issue today and this complexity will increase dramatically as online services become more advanced. Even the best programmer cannot anticipate a potential danger that only becomes apparent six months down the line. This means that the process of designing a secure web application is both expensive and involves many dependencies - and once the design is complete, you have to start again from scratch.

So it seems obvious that the aim should be to realise a Separation of Concerns: The business-oriented developer can dedicate himself to his software functionalities, the security specialist ensures that the application is and remains secure. However, the latter will not always be able to achieve this just by updating the code. An inserted web application firewall (WAF) should also be

installed in front of the web applications that can curb as many dangers as possible at any one time. After all, one of a web application firewall's most important tasks is to reduce the web applications' exposure. However, URL filtering it is not enough in today's usage scenarios for web applications.

The fundamental weakness of almost all web application firewalls available today is that they are set up based on static policies despite the dynamic nature of the subject matter. As such they require that all policies are managed in detail and constantly adapted to cope with new dangers or functionalities. WAF suppliers may well extol the learning modes with which the software adapts itself to new dangers. However the reality is: Even the learning modes output nothing other than static rules. There are certainly many known dangers which can be controlled with static Blacklisting, such as an SQL injection for example. However static rules do not provide a solution with Whitelisting because the static Whitelist lacks the foresight to identify future dangers and changes to web applications. At present this weakness has to be intercepted by the administrator with maintenance actions.

The organisational effort can be clearly reduced by establishing a dynamic procedure within the WAF. It requires software here that is closely contacted to the web application and that notices any unusual activities within the application automatically. It forces users to operate the web application in adherence with the regulations.

An approach which is otherwise fundamental for security measures has so far been forgotten for web applications: Namely the separation of declaration and control. Web applications provide a great deal of declarative information in the data that they output via the HTTP protocol and in the HTML language. Dynamic Whitelisting in the WAF means interpreting this information and making it examinable without having to resort back to static rules.

The first component required for dynamic Whitelisting is the URL encryption. The WAF encrypts and signs URLs which have been output by the application in HTML. As such, users are forced to use the application by conforming to the rules so that only URLs that have been defined by the application can be returned to the servers. A user - or perhaps even an attacker - does not receive too much information about the technologies used as is otherwise the case with web applications. URLs cannot be manipulated by the attackers anymore either. URL encryption offers extensive protection against all attacks which are based on URL manipulations, such as forceful browsing or URL parameter injections. Moreover, they also make it more difficult for an attacker to analyse the web application (reverse engineering) and force the user to abide by the application's

predefined sequences. Many WAF suppliers also still use the approach of noting valid URLs in cookies or on the WAF itself – a practice which is doomed to failure after a short time. However if every URL is protected in itself by encryption, then the WAF does not need to note the application context. The very same protection also applies if the application outputs certain URLs with a different content, for example subsequent to an application update or depending upon the user profile. Bookmarks and clear text URLs can be defined easily in the configuration as entry points. The WAF already covers a large target area with the encrypted URLs. Today's web applications sometimes use hundreds of URLs on each HTML page for links, pictures, CSS, Java Script and references to other web contents. The intelligent URL encryption adapts itself automatically to the business logic and gives strong protection without having to service thousands of signatures and rules.

However something that cannot be achieved with URL encryption alone is guarding against attacks based on manipulated user input. Web applications do not just output URLs, they also accept various input data in the forms, too. For example, the user name and password can be entered on a login page or the user can select the currency, date, account number or other user parameters on a deposit form. A WAF must ensure that an external user can only make the entries that the application actually wants to receive. Numerous Meta information already exist here in the web application's HTML which can be interpreted for dynamic Whitelisting instead of applying static rules. For example, the required input parameters are visible in the HTML, as well as the size of the input fields, which fields have unchanged values (hidden fields), which input fields only allow certain selection values (selection lists) and other criteria. The WAF reads and interprets the HTML when it is sent to the browser by the application. As soon as a form is identified the WAF creates a cryptographic signature which includes the form's criteria and enters this signature into the HTML form. As a result the HTML form – similar to the encrypted URLs – is protected in itself. As soon as a user sends the form to the application the WAF checks the input data against the signature and ensures that none of the input infringes the application's criteria. A major factor here too is that an administrator does not need to configure any details but that the WAF really identifies, interprets and examines the form's input conditions dynamically.

New W3C web standards such as the X-forms provide more declarative information which can be used by the WAS with dynamic Whitelisting. The WAF can thus examine more extensive input validations dynamically.



The administration effort can be significantly reduced by using dynamic Whitelisting. The initial configuration is sustained even if the application contents change or are very dynamic. A static rule can still be defined for particularly exposed pages, such as a login page or a registration form. However, experience shows that only five to ten static Whitelist rules are required in combination with dynamic Whitelisting per web application, the rest is covered dynamically.

About phion

phion is one of the leading European providers of solutions to protect company communications. With the netfence product portfolio phion offers solutions for the most demanding standards in terms of availability, security and management. netfence appliances consistently address all security-related aspects: From perimeter defence to the safe and highly available connections to branch offices, right up to blocking dangerous contents and protecting internal networks. airlock protects web applications like e-banking platforms and web services from any attacks or misuse.

All phion products have core management and are distinguished by their remarkable TCO.

phion is listed at the Vienna stock exchange (symbol: PHIO) and is based in Innsbruck, Austria and Zurich, Switzerland. phion's customers include well-known, internationally operating companies from all sectors of industry.