fireglass

WEB APPLICATION ISOLATION

White Paper

Organizations and their web applications

For many organizations web applications are at the core of the business. Unfortunately, application security is not only challenging in its own right, it suffers from additional hardships due to possible liability of compromised customers or partners, and the high cost of vulnerability patching and management. Public facing applications have to weigh security risks against the loss of business that restrictions or controls would cause, and internal applications provide a wide range of access so employees can execute their role, but creating the potential for breaches through malware compromise or malicious intent.

In order to protect from these threats, organizations deploy a dazzling array of different controls. However most of these solutions rely on detection to operate: distinguishing legitimate content from malicious content. Because 100% accuracy in detection is impossible, organizations deploy additional layers of protection that increase operations resources, false-positives, cost and complexity. Nevertheless, the "arms race" continues and compromises increase.

The Fireglass Threat Isolation Platform

The Fireglass Threat Isolation Platform eliminates common attack vectors against web applications. Fireglass executes and renders all potentially malicious content within secure containers, and sends a safe visual feed to the end-user without altering user experience. While requiring no client installation, it is compatible with all browsers, devices, and operating systems. Normally an attacker can bypass the rendering of the page and submit HTTP requests directly to the web-server. With Fireglass, the user can only communicate with the application through Fireglass' Transparent Clientless Rendering technology, and only by understanding the rendered image and interacting through key strokes and gestures.

This means that Fireglass can protect many forms of attack, including:

- **Bots, brute force attacks, worms and all other forms of automated attacks**
- **HTML manipulation attacks such as XSS, Unicode, or URL interpretation attacks**
- **Malware on the end-point executing attacks such as Man-in-the-Browser, HTML Injection, or form-grabbing**
- **Any infrastructure level exploit such as heartbleed, Shellshock, or any vulnerability in the web server, operating system or application server**
- **Input validation attacks such as SQL Injection or format string attacks**
- **Intelligence gathering by viewing HTML source**
- **Direct access to website APIs, bypassing client-side logic**

In addition, this protection comes without the onerous cost of alerts, false positives, and undue restriction

Current approaches to protecting against attacks

What do organizations do today?

Web applications today face many different threats from external attackers, ranging from DOS (Denial-of-service) attacks, platform or application specific exploits and vulnerabilities, and web-inherent attacks such as XSS (cross-site-scripting) or SQL Injection. There are multiple Techniques today to counter such threats, often deployed as IPS (Intrusion Prevention) or WAFs (Web Application Firewalls). These techniques suffer from significant drawbacks which will be outlined below.

Attack signatures

Many solutions such as IDS, or WAF employ a database of signatures to detect and recognize known attacks, that effectively function as a black-list.

Drawbacks

The reputation approach to detection is by definition behind the curve, it can only address known attacks. This creates an arms race between attackers and defenders and the former have a huge advantage as they adapt faster, and can use attacks that are hitherto unknown. Signatures are no match for sophisticated or persistent attackers. Beyond that, because signatures are static and lack context, an attack need only slightly modify its implementation in order for the signature to be subverted.

Positive model

Many WAFs allow the implementation of a positive model: only allowing traffic that has been established as legitimate and blocking all other traffic. In essence, a white-list only approach.

Drawbacks

The positive model relies on a comprehensive model of all forms of legitimate traffic. Even a simple application may have billions of permutations, so the self-learning of the application may take months – and even then may fault legitimate traffic. This constant self-learning never ends. The result is a solution that is very expensive to deploy and maintain, suffers from false positives, and limits the ability of the organization to react quickly to threats. The high cost makes deployment for most applications prohibitive, despite sensitive information within.

Heuristics, machine-learning and other "advanced" solutions

Many solutions augment or replace signatures with heuristics in order to detect previously unknown attacks. They rely on advanced algorithms to recognize malicious content based on similarities to past threats.

Drawbacks

Heuristics by their very nature are attempts to "guess" whether a pattern falls into a certain category (in this case "legitimate" or "malicious"). Because computer software lacks innate understanding and context about the content it inspects, it is prone to very high levels of false-positives. In addition, the algorithms are developed and trained based on existing attacks and future attacks do not necessarily resemble them.

How is Fireglass different?

Fireglass provides future proof security against external attackers with no degradation in user experience, no training periods, no costly on-going maintenance, no false positives, no operational burden and no costly training. Fireglass deters attacks not by detecting malicious traffic, but by eliminating the surface area where attacks can happen. With Fireglass the only possible surface area for attacks is a human attacker manually typing in a visible form field, where the client side Javascript validation would still allow the attack. This prevents the majority of possible attacks: all automated attacks break, invisible form fields cannot be manipulated, no intelligence is available to the attacker due to not being exposed to the HTML source, and no non-printable characters can be sent. The surface area remaining is so small that it can easily be hardened to prevent the remaining attacks.

Protecting web applications from infected or compromised users

Web applications today face many different threats from external attackers, ranging from DOS (Denial-of-service) attacks, platform or application specific exploits and vulnerabilities, and web-inherent attacks such as XSS (cross-site-scripting) or SQL Injection. There are multiple Techniques today to counter such threats, often deployed as IPS (Intrusion Prevention) or WAFs (Web Application Firewalls). These techniques suffer from significant drawbacks which will be outlined below.

Applications housing sensitive information, PII, or financial or financial-equivalent transactions are often affected not only by direct security threats, but also by threats targeting their users. Many breaches are the result of a "beach-head" of a compromised user that is infected with malware allowing the attacker access to resources. Public facing applications from malware used to harvest credentials or perform malicious transactions. This is very common in applications such as online-banking, e-commerce and other applications with financial transactions.

What do organizations do today?

Many companies have investment in a plethora of solutions:

- costly authentication mechanisms reduce login volumes, increase abandonment, and increase help-desk and operational costs,
- transaction monitoring solutions are extremely costly to implement and upgrade, and have additional operational expense
- malware forensic solutions come with an "expiration date" as attackers adapt, and can rarely be a stand-alone solution

Failure to keep malicious transactions or attacks in check, causes damage to the bottom line of the company and possible sanctions by regulatory bodies and credit card associations. Internal applications and networks are often not scrutinized in the same fashion, and an attacker can attempt many different attacks with low chance of detection.

How is Fireglass different?

When using the Fireglass Threat Isolation Platform, web-sites are delivered to the user as a visual feed rather All DOM elements, Javascript code and APIs are hidden and can't be modified or circumvented. Malware that attempts to form-grab will not be able to capture data. Malware that attempts to inject new HTML into the web-site will not be able to modify the visual feed. MITB (Man-in-the-browser) or MITM (Man-in-the-middle) attacks will fail direct HTTP or API access is not possible. Attempted "credential stuffing" is difficult as automated logins are not possible.

Protecting web applications from malicious or negligent insiders

Malicious users are often the source of the most destructive data breaches. Security controls are often geared towards identifying third parties or access violations which is not applicable. Other controls focus on auditing and creating a paper trail, but this means that often by the time the behavior is detected, the damage is done. Cloud applications are especially vulnerable as there is less visibility and control into the communication between the end-user and application beyond rudimentary logging.

What do organizations do today?

The core issue is that there are few if any effective tools to control the flow of data once the user has logged in. Malicious activity often remains unknown, or detected using anomaly detection or log analysis solutions after the fact, when damage is already done.

How is Fireglass different?

Fireglass allows organizations to protect the information in their web-applications by isolating it and enforcing a policy on when it can leave the user's browser. The deployment is simple and inexpensive, requires no training, and has no operational overhead or productivity loss. The users can interact with the data, but not remove it from the browser, depending on configurable policy. Admins can control operations such as clipboard (copy/paste), file downloads, "save image as", printing, and more.

Summary

Fireglass isolation fundamentally changes how organizations deal with web application security. Until Isolation, security teams had to integrate disparate solutions, while suffering from high implementation, management, and operational costs. All the while the arms race with attackers continued, and security incidents increase. Fireglass not only offers future-proof security that is resistant to adaptation - it does so without harm to user experience, and at a trivial implementation and management cost.

Contact us

www.fire.glass +1-(650)-50-FGLAS (34527) contact@fire.glass

4 World Trade Center, New York, NY 10007

fireglass

Copyright 2016 Fireglass, Inc. All rights reserved.