WHITEPAPER

INDUSFACE™

Haiku - A revolutionary approach to application scanner

SPA's are here to stay
Scanners struggle, need revolution
Indusface introduces Haiku

Need for Revolution?

Single Page Applications (SPAs) have taken over the web world . Even if you have not heard of them before, I am pretty sure you have encountered websites adopting this movement in some form or the other. With the advent of Angular JS and React JS, most of the sites have started adopting SPA frameworks and for a good reason. Angular & React style frameworks lends to better experience for both users & developers. Traditionally, individual pages of the website were loaded separately as user navigates through the website. This includes calls to backed server/ cache, loading resources and rendering the page. SPAs cut out much of the back-end activity by loading the entire site when a user first lands on a page. SPAs dynamically updates the single HTML page based on user interaction instead of loading of new page each time user interacts with the site. SPA frameworks use advanced JS templates to render the site, which means the HTML/CSS page overhead is next to nothing. Through SPAs, users experience super-fast sites while developers have templates that allows them to customize, test and optimize pages effectively.

All this is very good, but with this comes challenges for tools like automated scanners that are trying to crawl these sites. Traditional crawlers work on HTML pages by sending HTTP requests to a website and parsing the HTML response to get elements like links and input elements. Input and forms are used to send attack strings to probe the website for vulnerabilities and elements like links are used to get more pages to crawl. But this approach won't work in case of SPAs. Parsing HTML responses no longer gives the full picture of the website.

New age SPA sites presents the following challenges:

- Elements on pages are created dynamically via Ajax, JavaScript and DOM manipulation
- CSS styling breaks down typical button, input, link type of parsing – anything can be clickable
- Events can be attached to any element dynamically; not only as part of HTML page from server
- Frameworks like Angular use events and APIs extensively to sync model & view in their Model-View-Controller architecture making it onerous to impossible to fill in values using JavaScript. It even lets site designers make up their own pseudo tags that the framework will generate into real HTML in the browser.

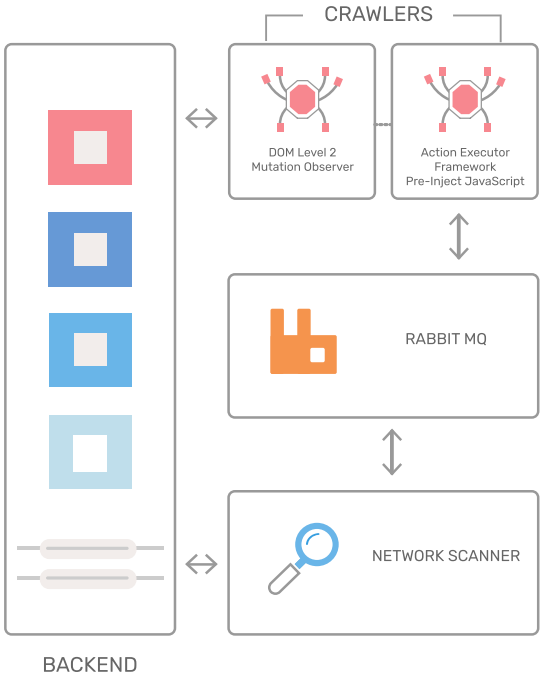
Crawler/Scanner for New Age Sites

To overcome these challenges it is paramount that crawlers are made JS aware and understand the browser context. This requires a complete relook on how scanners works and it is for this reason that Indusface Scanner was rebuilt from scratch (Project name Haiku). Built from the ground up on a completely different technology than its competitors, Haiku goes beyond traditional signature-based scanners to find more "real-world" vulnerabilities based on context aware crawling and scanning of websites.

Haiku uses Chromium based Electron browser to get a true rendering context so that websites can be truly rendered and crawled. In addition, the technology used gives 'debugger like' hooks into the browser that allows things like injecting JS, intercepting network requests, generating real mouse/keyboard events etc. All this ensures that Haiku Crawler can crawl sites built with any modern technology as well as traditional websites.

Haiku crawler has the following advantage

- Built using Electron (Chromium based), Node.JS enables the crawler to get true browser context & create deep hooks into browser
- Has ability to navigate .js heavy, single page applications
- Seamlessly Handles DOM Level 2 Event handlers
- Capture all requests including AJAX requests
- Action Executor Framework built into the crawler generates real user behavior like click/type and sequence of actions enabling crawler to mimic user behavior and navigate to places where traditional crawlers cannot
- Pre-injects custom JavaScript in every page enabling crawler to hook into actionable run time events and keep track of dynamically created elements after page load.



Deduplication

Another big challenge of crawling new age sites, is deduplication. Think about it, an user while navigating knows and understands if a similar page is revisited but how would crawler be aware of the same? With SPAs there is no concept of unique pages/URIs, different elements are loaded on the same HTML page depending on different user interaction. If the crawler is not aware of context and understands similarities/differences between elements then it risks getting stuck in a loop. Haiku crawler solves it in a unique way using page fingerprinting through Similarity Hash (SimHash). The crawler keep tracks of different elements as it navigates down a path interacting with these elements, and it remembers the state + action that it has encountered in the path so far and creates a SimHash. This SimHash is compared in every step and if it matches with any path the crawler is about to take then it knows that it is path it has already taken and skips it.

Creating Context

Even if all the above mentioned challenges are overcome, there remains the problem of creating context for the crawler. It needs to identify the correct heuristics that will enable it to completely crawl a site. E.g. sites with multi step wizards, unless crawler selects the right values and provides the proper inputs there is no way crawler will go to the next page. Indusface takes a multi-fold approach to address this problem.

Guided Scans:

This works with some human intervention that is done as part of managed service. As part of onboarding, our managed service team through inputs provided by customer or from monitoring the scan done, finds if crawler is unable to reach certain pages due to lack of proper inputs. If such instances are identified, our team updates the profile of the website to enable guided scan where they tell crawler to take certain action or provide particular inputs when it encounters a elements which it was not able to navigate before. When crawler reaches this place in the next scan, it will now know that guided scan is provided for this element and takes the action as provided in the configuration. This is one time configuration that needs to be set by our managed service team when the site is onboarded.

WAF Integration:

Indusface customers who have signed up for AppTrana Advance or Premium has additional advantage of automatic WAF integration where based on the traffic pattern observed in WAF, the places which are not reached by crawler are identified automatically and fed into site profile so that crawler navigates to these places during next scan.

Apart from it through WAF, scanner also learns the context of various pages/elements, what inputs are valid for certain field etc.. Based on these intelligence gathered, crawler during the next scan provides right inputs to these fields which enables it to navigate to places which it would have not been able to do in normal cases.

Premium and Advanced customers get immediate benefit from this integration. Also, learnings gained from the integration and guided scans are fed back to the scanner team to trigger crawler improvements for all customers that work out of the box.

What about finding vulnerabilities?

Great, but what about finding the vulnerabilities? Is it as simple as finding the pages then everything works seamlessly?

Well, nothing is as simple as that, to find vulnerabilities, the network requests identified through crawling of site need to be probed for vulnerabilities by simulating attacks. But in case of SPAs, there is a need to remember the context, more over as frameworks evolve rapidly, there is a need to keep adding and updating signatures. Having these signatures enabled for all sites is inefficient and would slow down the scan.

Another major challenge that was unique to Indusface that had to be considered was how to tap into the expertise of its pen-testing team that scans over 1000s of sites in a year. Indusface ethical hackers had huge untapped knowledge that could make Haiku stand out and find vulnerabilities that no competitors could. But for that we needed to build scanner that had a flexible architecture that enabled our pen-testing team to not only use the scanner in their day to day work but also, through that process, transfer their knowledge to scanner seamlessly that will benefit for all our customers.

It is to meet this need, as part of designing Haiku the scanning module (module that attacks to find vulnerabilities) was split from crawler and made completely plugin based. The scanner module was built to be platform agnostic which means it can support plugins written in any platform. This automatically opened up lot of new doors. Now, when our pen-testers find an attack vector or new way a vulnerability can be exploited, they can write a plugin in whatever platform they are comfortable with and feed it into scanner module.

For example, say a parameter is custom encoded, then our pen-tester could provide code for custom encoding for this site/parameter and automated scanner would do the rest in next scan. Another case can be when a pen-tester finds a way to exploit a particular vulnerability in a unique way then they would write a Haiku plugin to automatically attack this.

When Haiku Scanner scans the site next time, it would take this plugin enabled for the particular site and probe the site for this new vulnerability. This means now every automated scan for this site takes advantage of learning of pen-tester. Also, since the plugin is automated, our scanner team monitors these plugins and makes necessary changes where ever possible to make it generic which can be enabled for similar sites. So, next time, for similar sites, these vulnerabilities would be found through automated scan without the need for pen-testing.

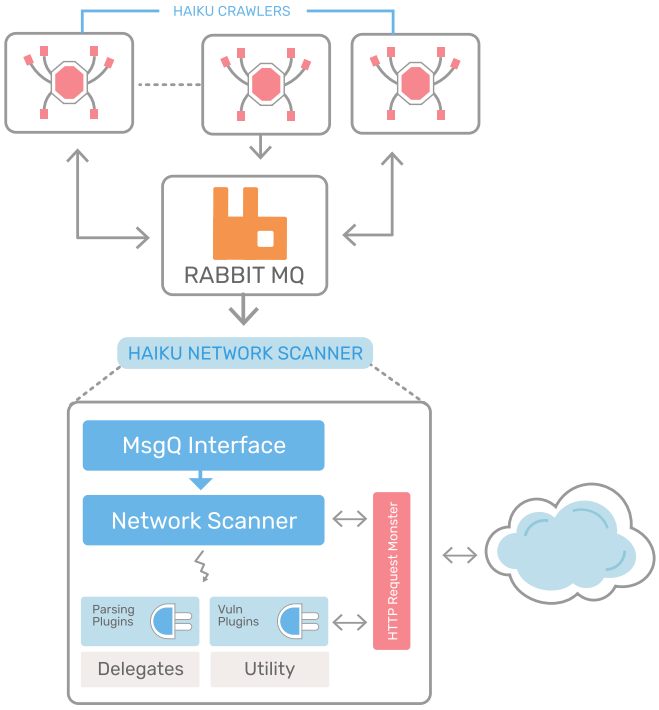
This speeds up their test process enabling us to provide pen-testing scans to customers at an affordable price that competitors cannot. This also now seamlessly transfers the knowledge, which was before only known to the pen-tester, to the scanner which could benefit all our customers.

How does scanner module work?

With scanning module completely decoupled from crawler, scanning/attacking works concurrently. As crawler finds new network requests on a webpage, it passes it on to scanner module by adding the requests on to Queue. Scanner module picks these request and based on plugin enabled for the site performs the necessary attacks making the scanner extremely performant.

Major advantage of Indusface Scanner module is:

- Concurrent execution with crawler makes it highly efficient.
- Decoupled architecture enables ability to take feed from not only crawlers but other sources for attacking.
- It is asynchronous, event based, extremely performant
- Protocol agnostic attack plugins enables pen-testers to easily add plugin for new vulnerabilities found.
- Enables Complex 'state-machine' detection rules instead of simple regex based detection



Authenticated Scans

Haiku takes a fresh approach to authenticated scans. It has built in heuristics to identify login page, so there is no need for record & replay. Customer has to just provide login details and it will work seamlessly. At any point if Haiku is logged off, it automatically identifies that it has encountered a login page and logs in again making it highly context aware.

But how would it work with decoupled crawler & scanner that supports pause/ resume ?

Well, scanner module asks crawler to refresh the request before it picks it to attack. This is how it works, before attacking a request, the scanner asks the crawler to refresh the request sent earlier. If for some reason the network request is not generated again, the request is updated with the latest session info. The advantage of all this is that session will be updated even across pause resume spanning any length of time. More than that, since requests are being refreshed, not just sessions but also params/tokens are updated. While other scanners track sessions, Haiku takes it to next level and refreshes requests ensuring all the context is maintained and up-to date for scanning.

To conclude, no matter what scanner you choose, please be aware of the following facts

- SPAs is taking over the internet, even if your application has not yet adopted SPA framework, it is just matter of time and in order to provide user experience equivalent to your competitor you will be forced to adopt it soon.
- Even if you don't completely move, there will be modules that use cutting edge browser features that need to be supported.
- Traditional scanners will not work well with SPA sites.
- Any crawler that needs to scan new age sites, needs to be context aware and needs special integrations/ability to generate context either through human inputs or some 3rd party integrations like WAF which could provide it context. Without this, the crawler will be ineffective and not go anywhere from the home page.
- Rapid changes to frameworks means attack surface keeps changing constantly and requires expertise and constant updates to scanner/plugin to make the scanner performant.

Hope, this short blurb on Indusface scanner, provides a reasonable insight on why Indusface scanner is different from the competitors and could find real world vulnerabilities that competitors cannot.

Evaluate
Your Site
Security

START FREE

Biweekly security scans | OWASP top 10 threat detection | SANS top 25 vulnerability detection | Scan behind authenticated pages

ABOUT INDUSFACE

Indusface is a leading application security company protecting 1100+ customers in more than 30 countries. Our products provide continuous threat visibility, instant protection, whole site acceleration & monitoring. Available On-premise, As-a-Service (AppTrana) and through the AWS Marketplace, our security products have also been mentioned in the Gartner Magic Quadrants for Application Security Testing and Web Application Firewall.

Bengaluru | Vadodara | Mumbai | New Delhi | San Francisco

Contact: marketing@indusface.com