You Want Better Software

Let's Fix the Broken Pieces

Kevin E. Greene

OpenText Cybersecurity, Public Sector CTO

Agenda

- Introductions
- Reflections
- Major Software Security Initiatives
- The Broken Pieces
- If I was still a Fed…
- OpenText Software Security

Introductions

Science and Technology

MITRE

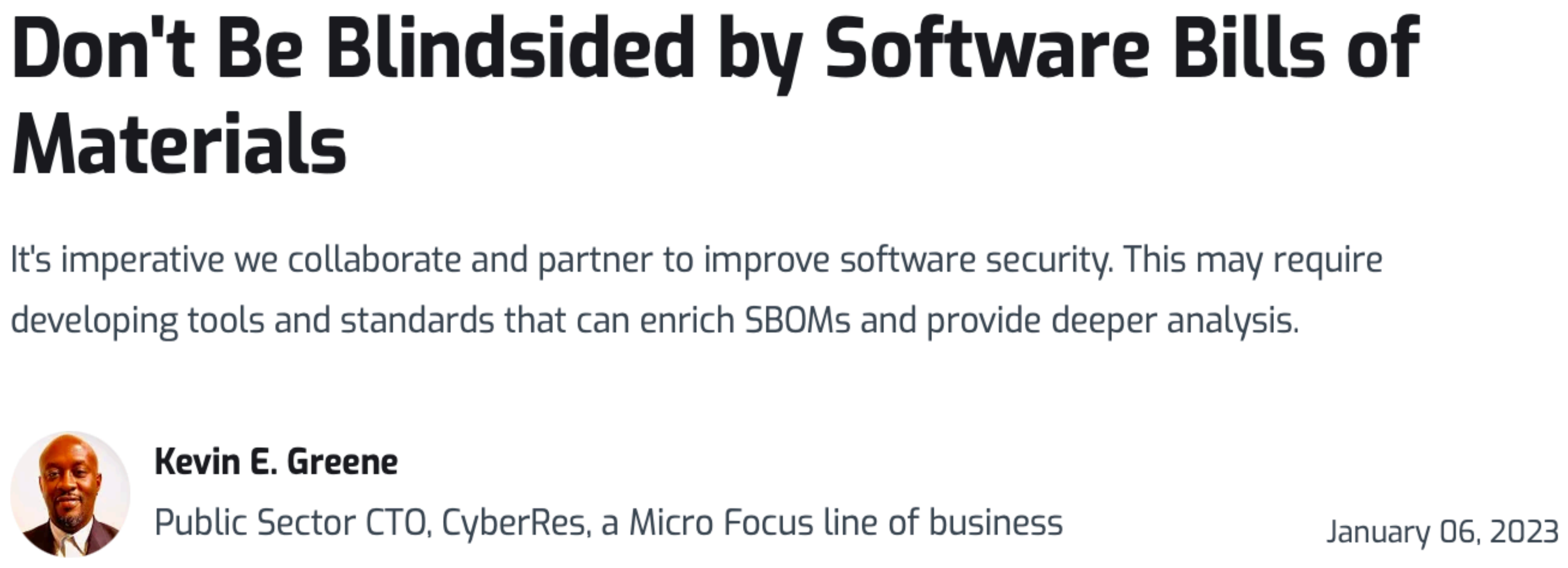
opentext™

Kevin E. Greene

Key Highlights

- Software Assurance Marketplace (SWAMP)
- Common Architectural Weakness Enumeration (CAWE)
- Hybrid Analysis Mapping (HAM)
- Static Tool Analysis Modernization Project (STAMP)
- RevealDroid – Mobile Application Security
- General Analysis Toolkit Using Record and Replay (GATOR) - REnigma
- DevSecOps Tiger Team
- CTID Research - ATT&CK Research
- CWE Program Support
- Threat Informed Defense – Zero Trust
- Zero Trust Lab

Reflections



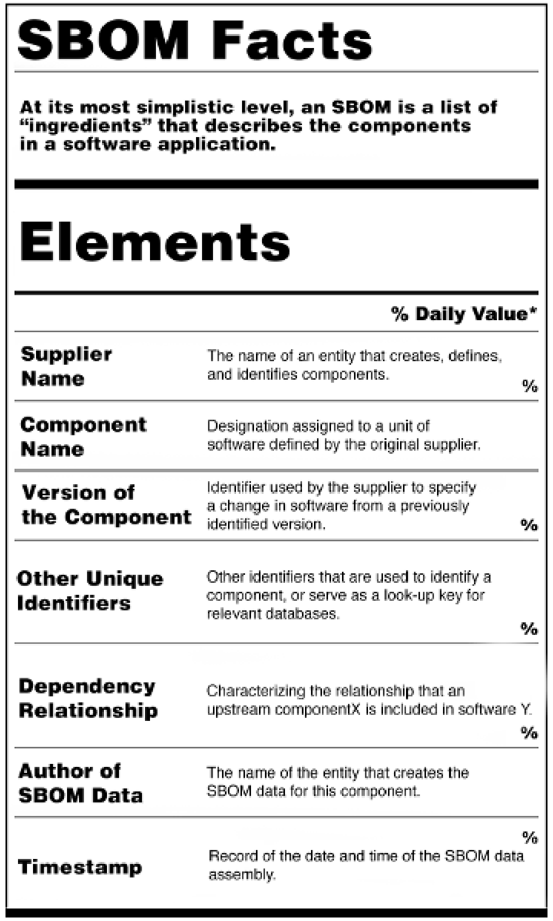
Don't Be Blindsided by Software Bills of Materials

It's imperative we collaborate and partner to improve software security. This may require developing tools and standards that can enrich SBOMs and provide deeper analysis.

Kevin E. Greene
Public Sector CTO, CyberRes, a Micro Focus line of business

January 06, 2023

Major Software Security Initiatives



SBOM Facts

At its most simplistic level, an SBOM is a list of "ingredients" that describes the components in a software application.

Elements

		% Daily Value*
Supplier Name	The name of an entity that creates, defines, and identifies components.	%
Component Name	Designation assigned to a unit of software defined by the original supplier.	
Version of the Component	Identifier used by the supplier to specify a change in software from a previously identified version.	%
Other Unique Identifiers	Other identifiers that are used to identify a component, or serve as a look-up key for relevant databases.	%
Dependency Relationship	Characterizing the relationship that an upstream componentX is included in software Y.	%
Author of SBOM Data	The name of the entity that creates the SBOM data for this component.	
Timestamp	Record of the date and time of the SBOM data assembly.	%

Software Bill of Materials (SBOMs)

Software Attestation

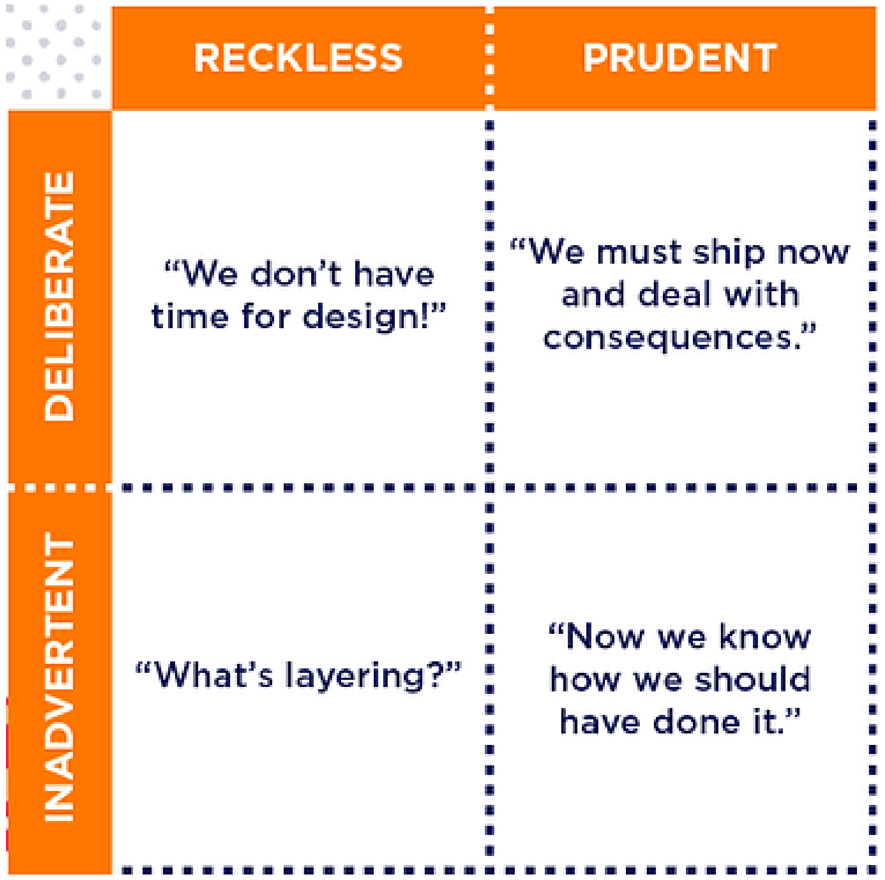
Software Liability

The Broken Pieces

Too Much Technical Debt

There is insurmountable technical debt that has not been paid down that accelerates vulnerabilities in software. Poor designs that do not enforce good design patterns are prone to be high maintenance.

$$\begin{tikzcd}\boxed{\text{CAWE}} \arrow[r, dash] & \boxed{\text{CWE}} \arrow[r] & \boxed{\text{CVE}}\end{tikzcd}$$



	RECKLESS	PRUDENT
DELIBERATE	"We don't have time for design!"	"We must ship now and deal with consequences."
INADVERTENT	"What's layering?"	"Now we know how we should have done it."

Source: https://www.iteratorshq.com/blog/what-is-technical-debt-in-software-development-and-how-to-manage-it/

"If tech debt was real debt, Dave Ramsey would yell at us all day long."

Brian Knapp

The Broken Pieces

Secure Software is Not Realistic

Secure software is not attainable nor realistic in modern software development - all the industry best practices imply (whether **implicitly or explicitly**) the notion or idea that these set of best practices will produce “**secure software**”. That is a flawed premise and sets unrealistic expectations. NIST SSDF points to various sources and list activities, but the context implies secure software — <u>assured software may be more realistic in framing the outcomes for software development activities.</u>

> ***software developed or engineered in such a way that its operations and functionalities continue as normal even when subjected to malicious attacks***

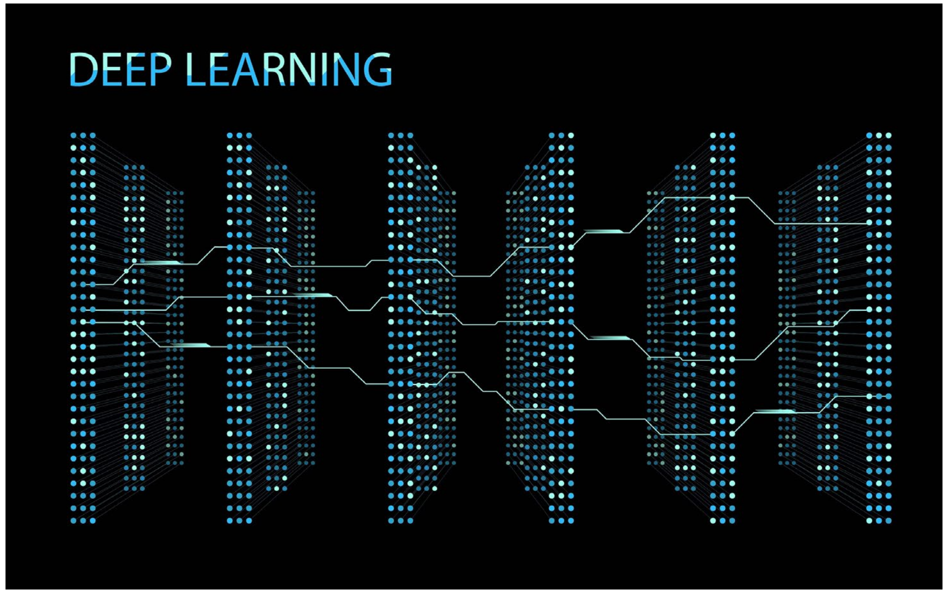
Source: Study.com

All software have vulnerabilities. Building something securely, doesn’t guarantee ”secure software”.

The Broken Pieces

Residual Risk in AST Tools

We cannot consistently measure the performance of AST tools, scientifically we do not know what tools are good at (sweet spot), and conversely where these tools struggle. There needs to be some ground truth, and confidence in what tools can and cannot find. Otherwise, we are missing key issues that result in poor tool coverage and give a false sense of assurance and security that will make software attestation and software liability difficult to formalize.



Understanding the false-negative, false-positive, and true-positive rates in tools is crucial to managing risk in software.

The Broken Pieces

SDL and SDLC Do Not Work

Prove to me otherwise with data and scientific proof that is reliable to draw from. This is my assertion given the growing rate of bad software prone to attacks. Identifying the activities and tasks that are known to produce better quality and security in software will streamline and improve how we develop and build software. Prioritizing and integrating these activities and tasks in DevOps and DevSecOps practices will give software vendors the confidence in making the appropriate investment decisions to formalize software security as a standard practice.

SOMEBODY JUST MADE UP 'SDLC' AND EVERYONE WENT ALONG WITH IT

Source: scrum.org

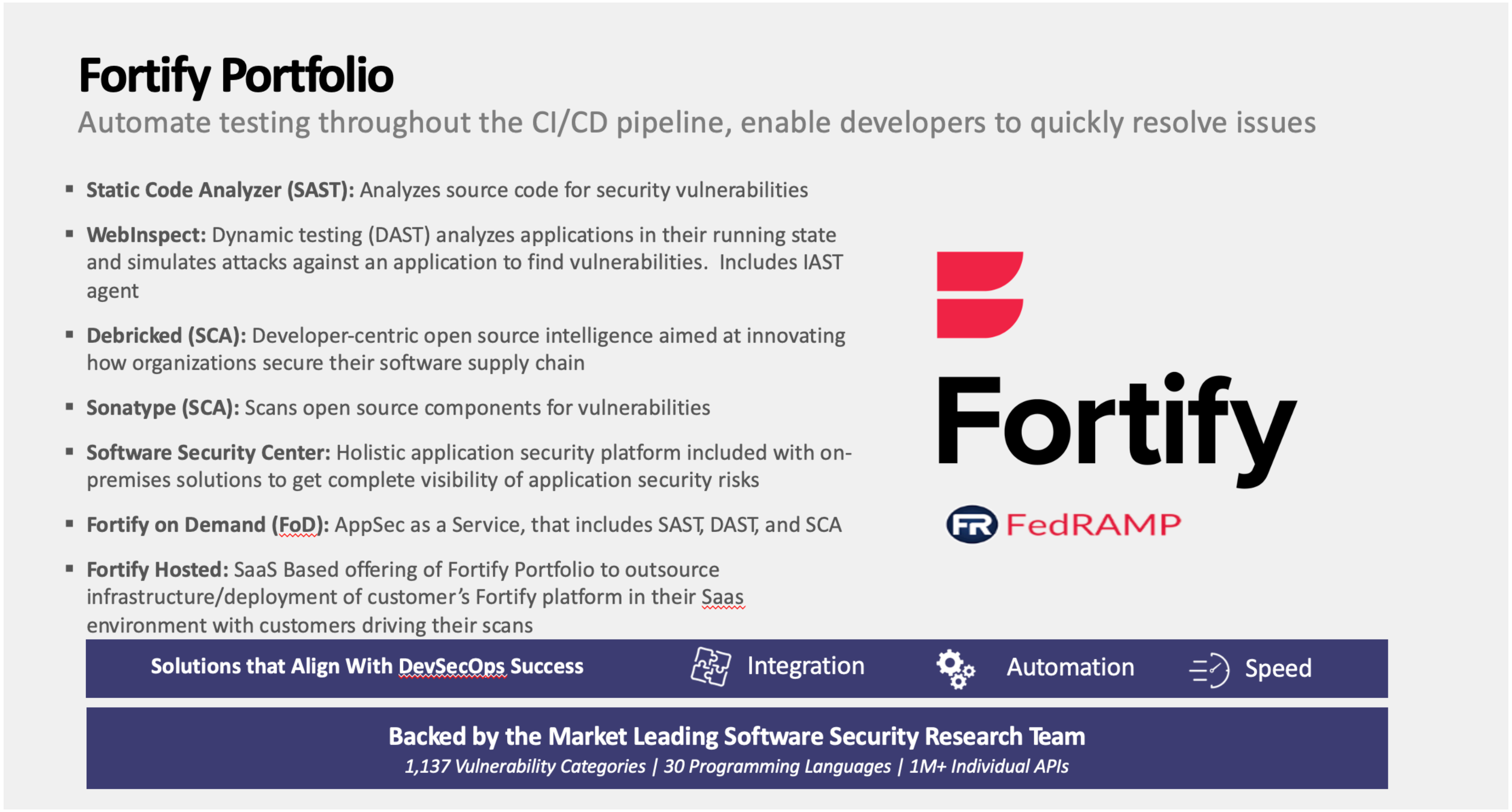
Find those tasks and activities and do them religiously!!

If I Was Still a Fed

Research questions that are fascinating and interesting

- How close can we get in predicting attack and defect proneness rates in software?’
- Can LLM be used for automated and iterative threat modeling to reason about design decisions and the ability to enforce robust security protection?
- How do you incentivize OSS developers to design/develop better quality and security in software?
- What properties in software are directly related to quality and security?
- Is self healing or automated patching possible in modern software development?
- What percentage of weaknesses (CWEs) in OSS will result in CVEs?

OpenText Software Security



Fortify Portfolio

Automate testing throughout the CI/CD pipeline, enable developers to quickly resolve issues

- **Static Code Analyzer (SAST):** Analyzes source code for security vulnerabilities
- **WebInspect:** Dynamic testing (DAST) analyzes applications in their running state and simulates attacks against an application to find vulnerabilities. Includes IAST agent
- **Debricked (SCA):** Developer-centric open source intelligence aimed at innovating how organizations secure their software supply chain
- **Sonatype (SCA):** Scans open source components for vulnerabilities
- **Software Security Center:** Holistic application security platform included with on-premises solutions to get complete visibility of application security risks
- **Fortify on Demand (FoD):** AppSec as a Service, that includes SAST, DAST, and SCA
- **Fortify Hosted:** SaaS Based offering of Fortify Portfolio to outsource infrastructure/deployment of customer's Fortify platform in their Saas environment with customers driving their scans

Fortify

FedRAMP

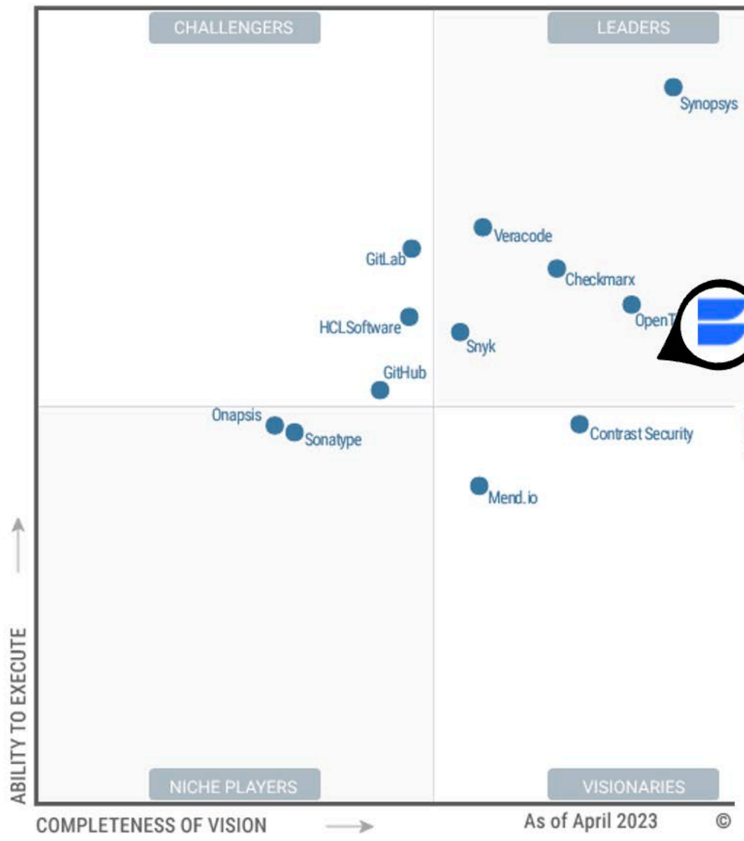
Solutions that Align With DevSecOps Success Integration Automation Speed

Backed by the Market Leading Software Security Research Team

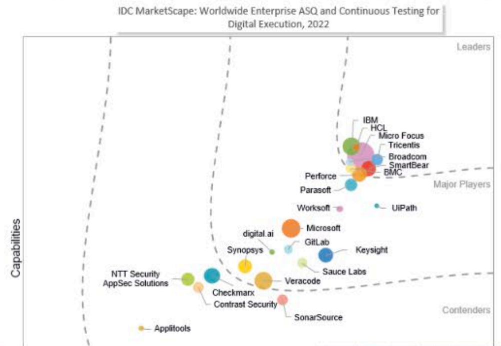
1,137 Vulnerability Categories | 30 Programming Languages | 1M+ Individual APIs

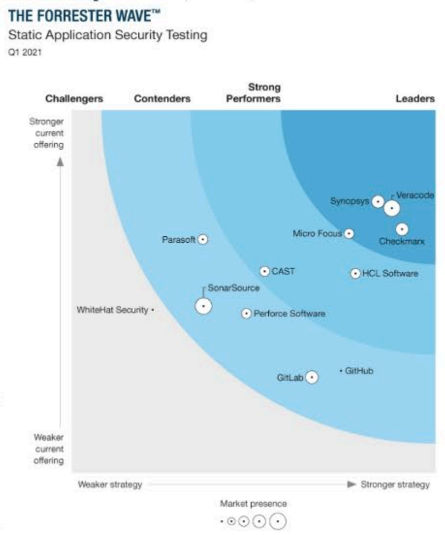
OpenText Software Security

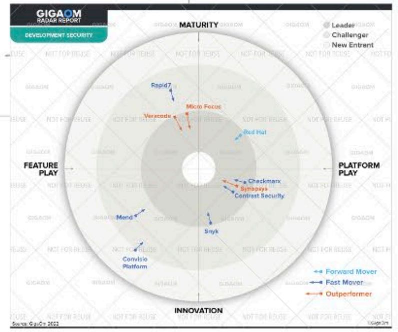
Fortify leadership in the AppSec market



IDC MarketScape Worldwide Enterprise Automated Software Quality and Continuous Testing for Digital Execution Vendor Assessment







Gartner peerinsights customers' choice

PeerSpot No.1 Ranked ★★★

Gartner®

FORRESTER®

G2 Leader SUMMER 2022

GIGAOM

Thank you for your time

Email – kgreene@opentext.com

@iamkevtorious

https://www.linkedin.com/in/kevgreene/

#dontbugmetodeath

opentext™