Solving Application Security with Artificial Intelligence

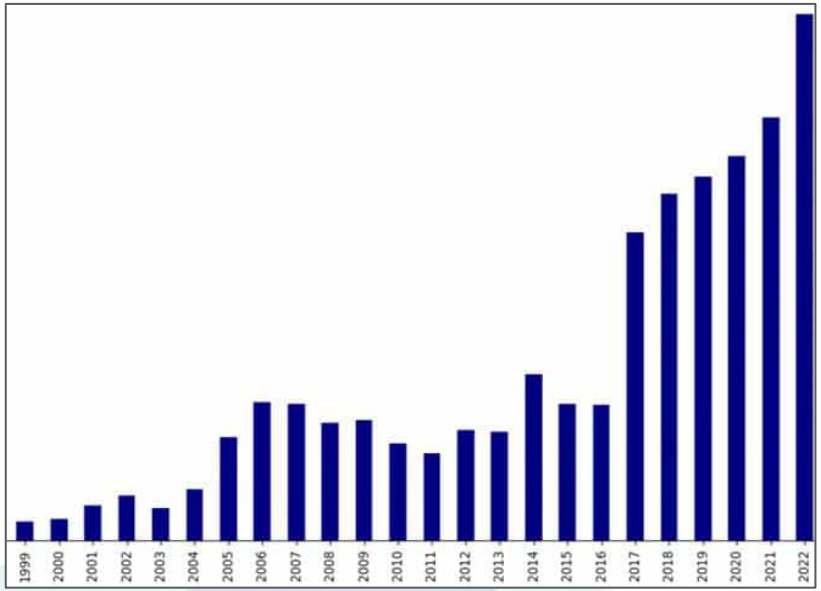
Stefan Streichsbier - CTO
stefan@guardrails.io

What we'll cover today

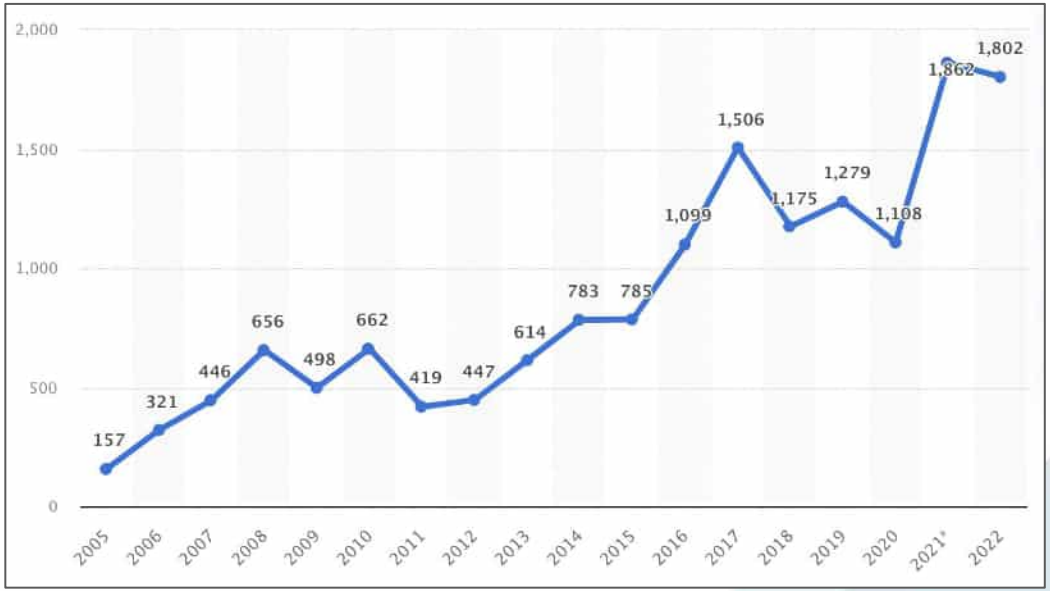
- **Why this is an important problem**
- **Challenges with current state of AppSec**
- **How AI is Revolutionizing AppSec**
- **GuardRails.ai: Timeline and Expectations**

Why this is important

The problem is getting worse …



Percentage of CVEs Published

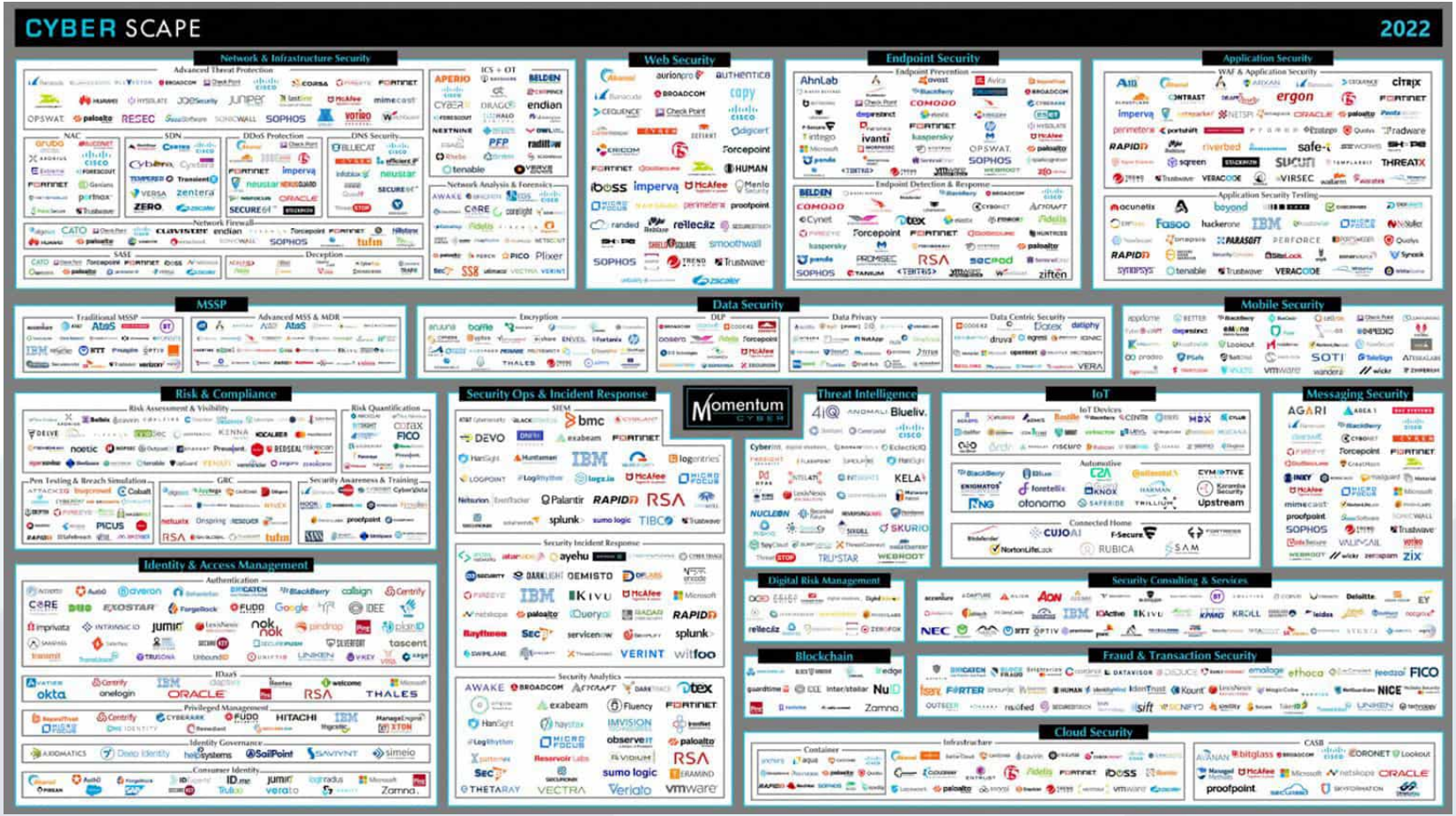


Annual number of data breaches (US)

https://jerrygamblin.com/2023/01/01/2022-cve-data-review/

https://www.statista.com/statistics/273550/data-breaches-recorded-in-the-united-states-by-number-of-breaches-and-records-exposed/

… despite an unlimited supply of solutions



Source: MomentumCyber

Challenges with current state of AppSec

Challenges with the current State of AppSec

1. **Diverse Vulnerability Types**
   a. Affecting every part of the stack code, dependencies, secrets, cloud, etc
   b. Each type requires specialized attention and different mitigation strategies
   c. Rapid tech evolution means the emergence of more complex risk

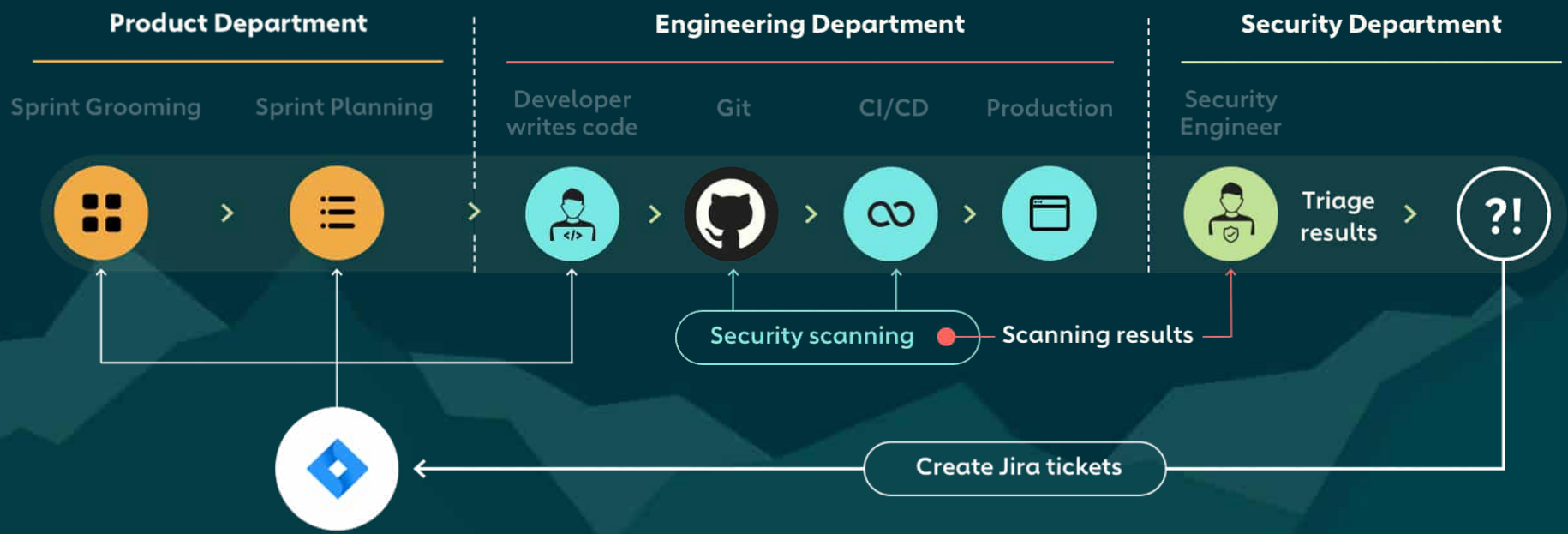
1. **Fragmented Tools**
   a. No single tool excels in detecting all vulnerability types
   b. Companies resort to using multiple tools, straining budgets
   c. Integrating and managing these tools becomes complex

1. **Alert Fatigue**
   a. Tools often generate an overwhelming number of alerts.
   b. Sifting through these to identify genuine threats is time-consuming.
   c. Constant alerts cause fatigue, leading to potential oversight of critical vulnerabilities.

1. **False Positives**
   a. Many flagged vulnerabilities might be false positives.
   b. Discerning real threats from noise is a daunting task.
   c. The effort to validate issues takes away from actually addressing genuine threats.

1. **Shortage of Expertise**
   a. Limited availability of skilled security experts and trained developers
   b. Current team members face a steep learning curve in application security
   c. Time is spent on analyzing, prioritizing, and fixing vulnerabilities rather than on core development tasks, which is prohibitive

Security workflows are broken



Brief Overview of Large Language Models

1. **Defining Large Language Models (LLMs)**
   a. LLMs are state-of-the-art AI models trained on vast amounts of text data
   b. Capable of understanding, generating, and interpreting human language
   c. Evolved from simple keyword recognition to deep semantic understanding

1. **LLMs vs Traditional AI (ML)**
   a. Traditional ML models often rely on specific use-cases
   b. LLMs autonomously identify patterns and relationships in data
   c. LLMs can perform tasks they weren't explicitly trained for

1. **LLMs in Application Security**
   a. Contextual Analysis: Primed with context about the code and business risk
   b. Reasoning Capabilities: Going beyond simple pattern/logic
   c. Generating Fixes: Provide code fixes for vulnerabilities

1. **Adaptability of LLMs**
   a. Tool use allows LLMs to achieve complex tasks requiring real time knowledge
   b. Few shot prompting can fine-tune LLMs for specific use-cases
   c. With feedback loops, LLMs get better over time, adapting to specific company codebases and workflows.

1. **Synergy with Human Expertise (Co-pilot paradigm)**
   a. LLMs amplifying human expertise rather than replacing it
   b. Automating repetitive free up security experts for higher-level strategic work
   c. Collaboration bridges the skill gap in security and development teams

How AI is Revolutionizing AppSec

Revolutionizing AppSec with AI

1. **Understanding Code and Business Context**
   a. Business context and risk about an application are key inputs to the analysis
   b. LLMs can understand code, dependencies, and purpose of an application
   c. Allows people to interact with applications in entirely new ways

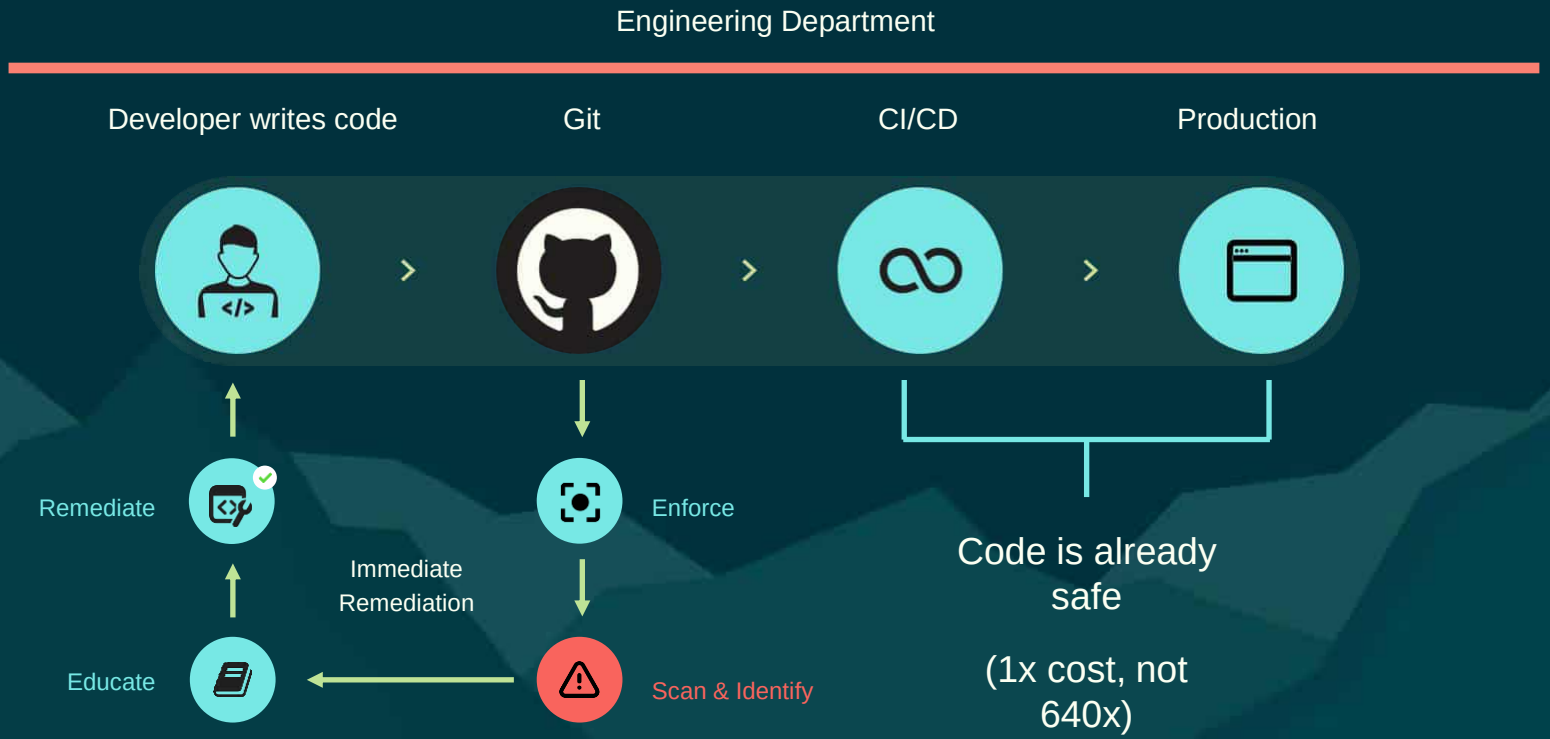
1. **Accurate Vulnerability Detection**
   a. Hybrid approach can leverage traditional detection and super charge it with AI
   b. LLMs are primed with specific code, business and vulnerability context
   c. Detection can be improved through human feedback loops

1. **Reduction in False Positives**
   a. Interpret the broader context of the code to ensure precise alerts
   b. Context-awareness ensures only relevant vulnerabilities are flagged
   c. Reduced noise allows focus on genuine threats.

1. **Automated Code Fixes**
   a. Developers iterate on solutions to create optimal code fixes
   b. Streamline the mitigation process, reducing time from detection to resolution
   c. Preferred fixes can be added to company-wide skill arsenal

1. **100x your existing teams**
   a. Faster time to triage and fix a vulnerability
   b. Security and Software engineers are skilled up through the interaction
   c. Higher engagement and satisfaction with “boring” tasks

GuardRails.ai boosts developer security & productivity



GuardRails.ai
Timeline and Expectations

GuardRails.ai - The Future of AppSec

1. **Get Started with GuardRails.io**
    a. GuardRails.io already covers SAST, SCA, IaC, and Secrets for 22+ languages
    b. An easy upgrade path to GuardRails.ai will be provided
    c. Reach out to our team to learn more and get early access to GuardRails.ai

1. **Launch Timeline**
    a. Anticipated launch by the end of Q4 or early Q1
    b. Early users will be able to shape the direction and order of the roadmap
    c. Continual development ensures consistent advancements post-launch

1. **First Version Highlights**
    a. Reimagined UI/UX including "perfect workflow"
    b. Initial support for GitHub, both cloud and on-premise
    c. Precise static analysis and secret scanning with unparalleled accuracy

1. **Upcoming Features**
    a. Dependencies and additional scanning methods to be integrated soon
    b. Integrations with GitLab, BitBucket and Azure DevOps
    c. Deep workflow integrations with chat tools, issue trackers, and IDEs

Q & A