

API Protection Tools - AI Solutions Overview

Vitaly Davidoff

JFrog Appsec Architect and Leader



Product Security Lead at JFrog, CISSP, CSSLP
15 years of experience as a developer
7+ years experience in application security



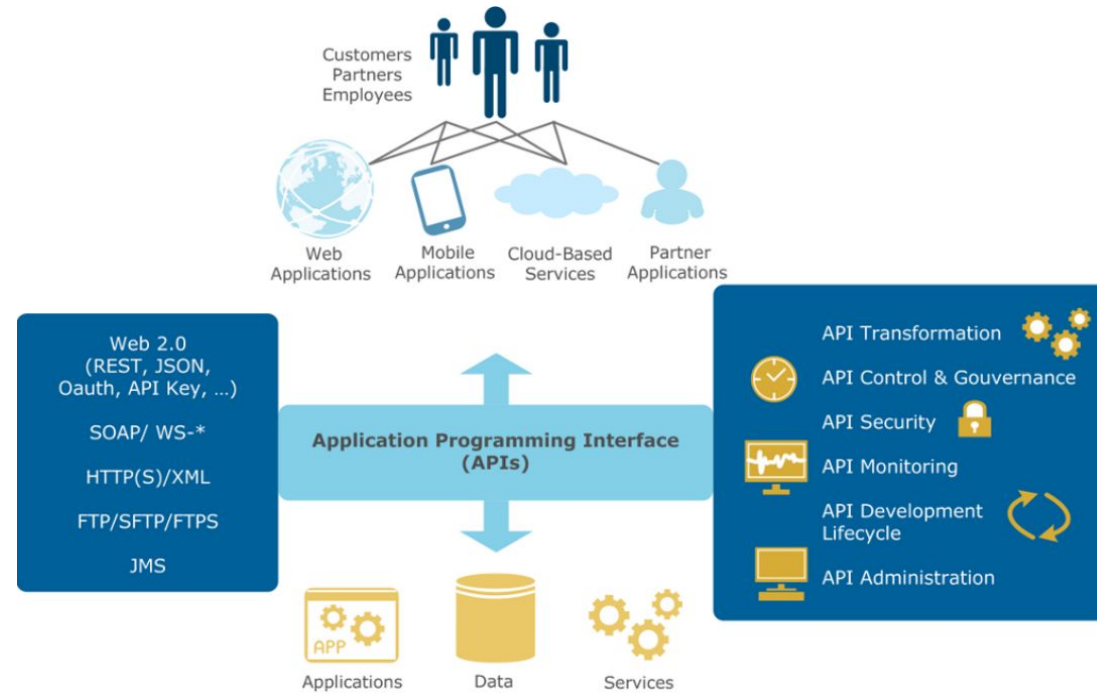
Agenda

- What is API Security
- Protocols Evolution
- Common Attack Vectors
- WAF or not to WAF?
- Technical Part
- Capabilities (AI involving)
- Q&A

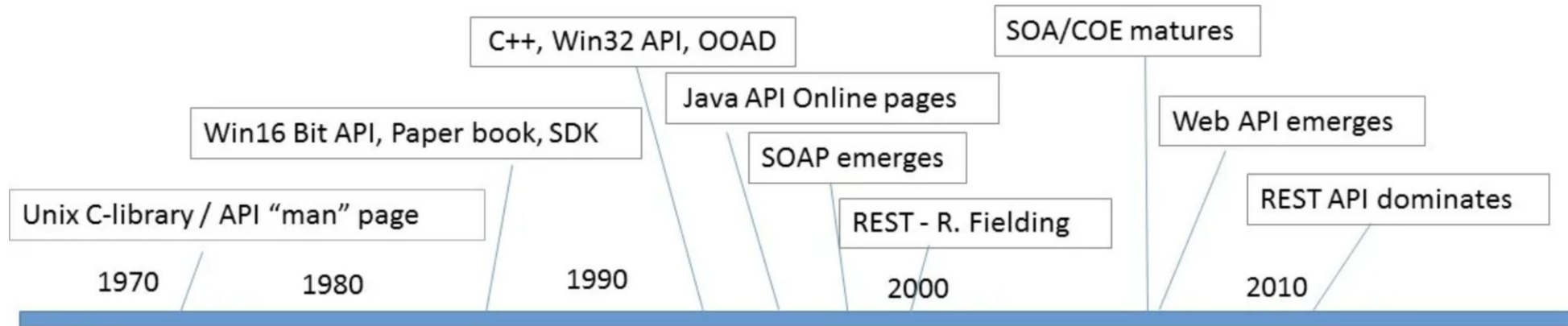


What Is API?

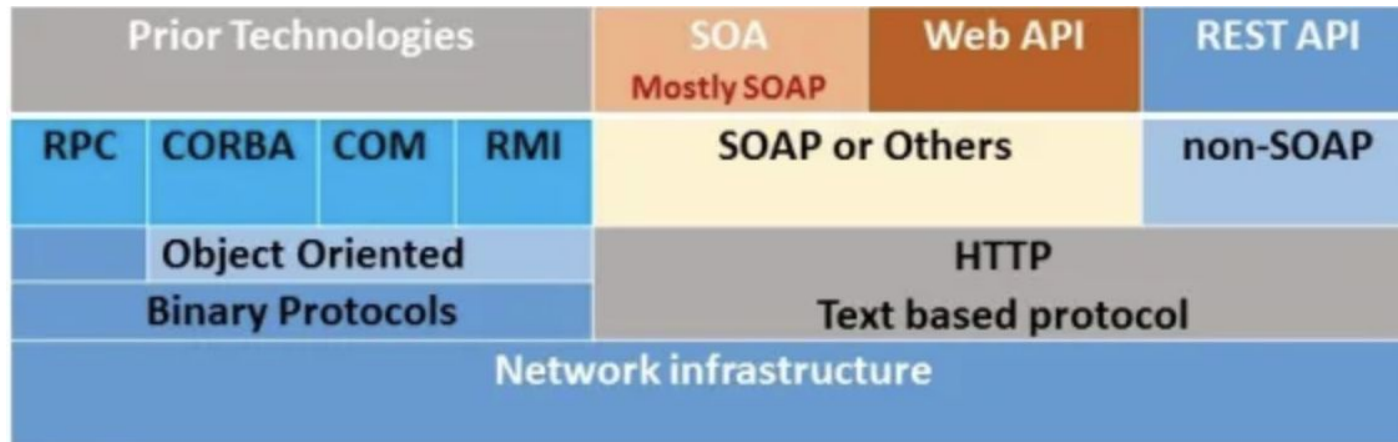
An application programming interface (API) is a set of tools, definitions, and protocols for integrating application software and services. It's the contract that lets your products and services communicate with other products and services without having to constantly build new connectivity infrastructure.



API Protocols Evolution



Rough timeline of API and SOA





Public

Public / Open - APIs used by partners and developers who build innovative apps and products

Partner

Partner / B2B - APIs used by business partners including suppliers, providers, resellers and others for tighter partner integration

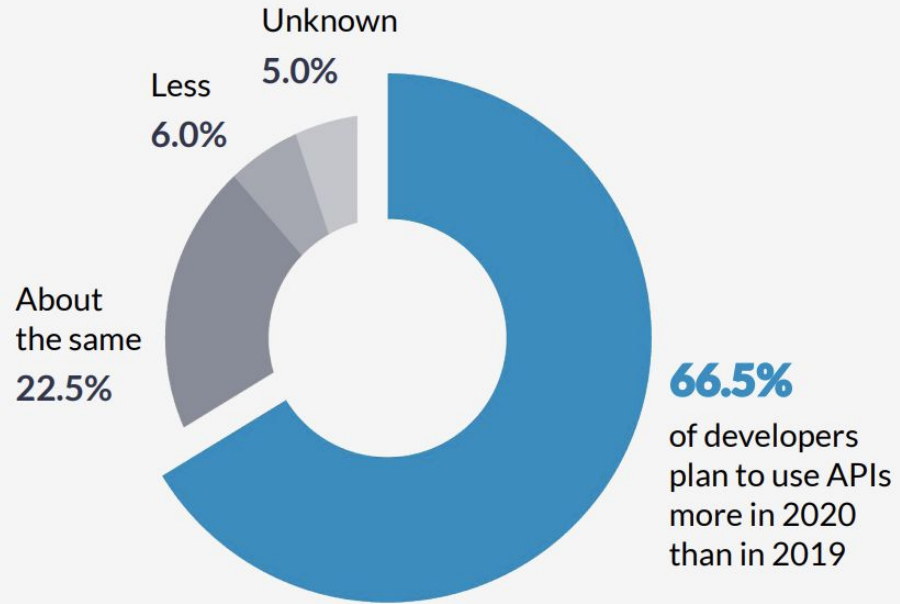
Internal

Internal - APIs are used by developers within enterprises

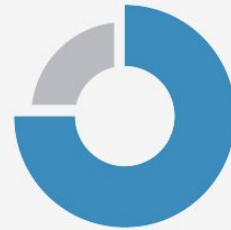
If it operates in a
request/reply model,
IT IS AN API



API - Trend



66.5%
of developers
plan to use APIs
more in 2020
than in 2019

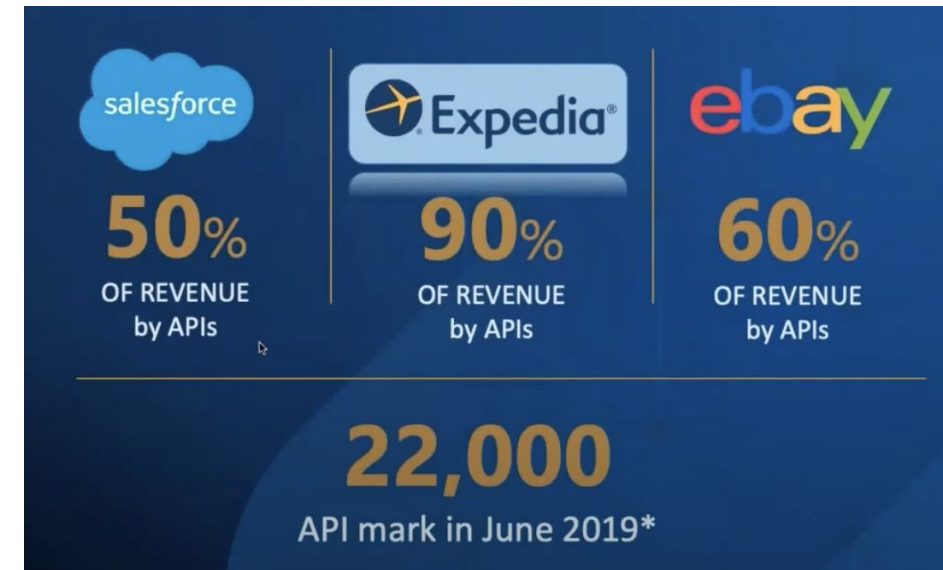


3/4
of enterprises are
using APIs today



60%
of enterprises
developed APIs in
the past 5 years

Statistics from RapidAPI Developer Survey and Insights



What Can Attackers Do With API Vulnerabilities

DEEP SEC

T-Mobile Alerts 2.3 Million Customers of Data Breach Tied to Leaky API



Author
Tom Spring
August 24, 2018
12:42 pm
2 minute read

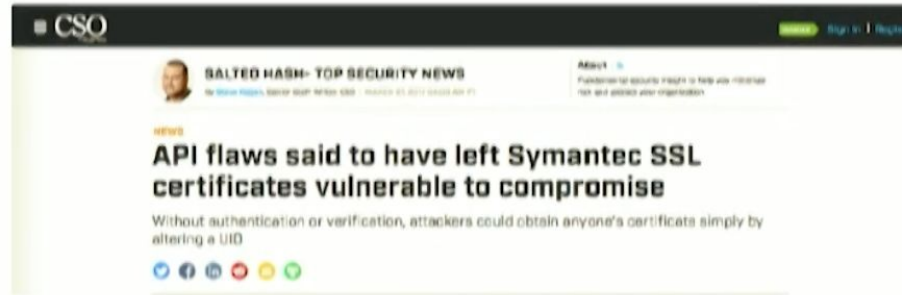
Google shuts down Google+ after API bug exposed details for over 500,000 users

Search giant says it found no evidence that any user data was misused.

News

Vulnerability Found in Venmo Public API Causing Massive Data Leak

By Nitish Singh · July 18, 2018



Salesforce Security Alert: API Error Exposed Marketing Data

Marketing Cloud Data Potentially Accessed or Corrupted Over 6-Week Period

Lessons Learned From Hacking The Tesla API



Hacking Starbucks and Accessing Nearly 100 Million Customer Records

June 20, 2020 · samwcyo

API Attacks Are getting Sophisticated
AI based BOT is becoming the weapon of choice

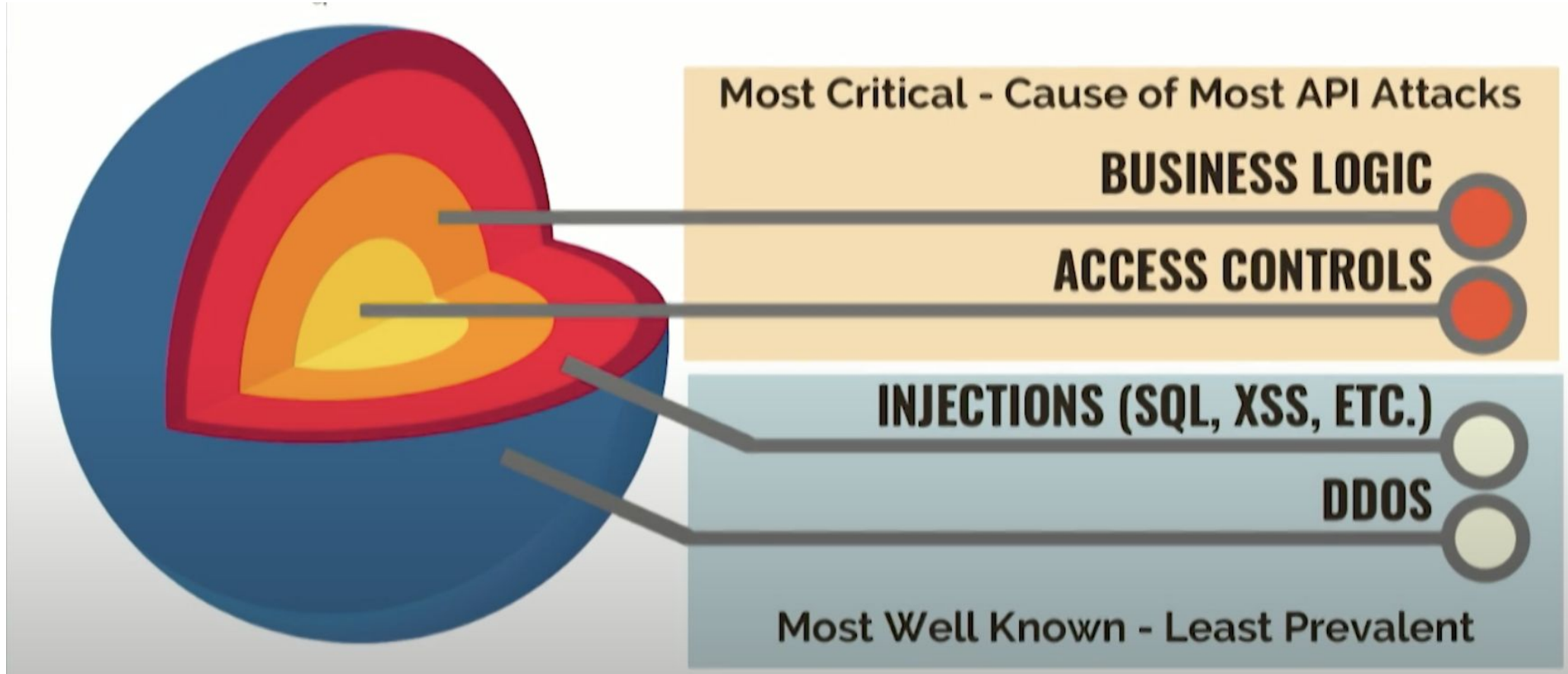


API Security - Common Problems

OWASP API Security Top 10	OWASP Top 10 (2017)
API1: Broken Object Level Authorization	A1: Injection
API2: Broken User Authentication	A2: Broken Authentication
API3: Excessive Data Exposure	A3: Sensitive Data Exposure
API4: Lack of Resources & Rate Limiting	A4: XML External Entities (XXE)
API5: Broken Function Level Authorization	A5: Broken Access Control
API6: Mass Assignment	A6: Security Misconfiguration
API7: Security Misconfiguration	A7: Cross-Site Scripting (XSS)
API8: Injection	A8: Insecure Deserialization
API9: Improper Assets Management	A9: Using Components with Known Vulnerabilities
API10: Insufficient Logging & Monitoring	A10: Insufficient Logging & Monitoring



What Can Attackers Do With API Vulnerabilities



MOST EXPLOITED API VULNERABILITIES

RBAC & ABAC VULNERABILITIES

Providing fine-grained access to resources can often lead to an explosion of RBAC roles or ABAC rules that can be easily exploited if not properly tested.

Privilege escalation vulnerabilities (RBAC) and unauthorized access to resources (ABAC) are extremely difficult to find. Such vulnerabilities have contributed to the most prominent API attacks and could cost companies extremely high fines for breaching regulatory guidelines.

BUSINESS LOGIC FLAWS

Technical security vulnerabilities (like SQL injections) come from coding errors. However, business logic vulnerabilities are due to mistakes in how the application was intended to function and APIs are the preferred point for hackers to exploit these vulnerabilities.

Since the code was correctly written, business logic vulnerabilities cannot be detected using App security tools or traditional source code analysis techniques like SAST and DAST vulnerability scanning solutions.



BOLA (Broken Object Level Authorization)

Example Attack Scenarios

Scenario #1

An e-commerce platform for online stores (shops) provides a listing page with the revenue charts for their hosted shops. Inspecting the browser requests, an attacker can identify the API endpoints used as a data source for those charts and their pattern `/shops/{shopName}/revenue_data.json`. Using another API endpoint, the attacker can get the list of all hosted shop names. With a simple script to manipulate the names in the list, replacing `{shopName}` in the URL, the attacker gains access to the sales data of thousands of e-commerce stores.

Scenario #2

While monitoring the network traffic of a wearable device, the following HTTP PATCH request gets the attention of an attacker due to the presence of a custom HTTP request header `X-User-Id: 54796`. Replacing the `X-User-Id` value with 54795, the attacker receives a successful HTTP response, and is able to modify other users' account data.

CITI ABAC VULNERABILITY FINANCIAL DATA OF 360,000 CUSTOMERS EXPOSED

How was this hack perpetrated?

- The attack relied on parameter tampering in the APIs.
- A missing ABAC validation or role assignment allowed any authenticated user in the application to request API resources belonging to any other user/customer just by knowing the other customer's account number.
- Predictable account numbers (e.g. incremental numbers 100034567, 100034568, etc.) allowed the attackers to type in repeated account numbers tens of thousands of times to access the account data.



API5:2019 Broken Function Level Authorization

Scenario #2

An API contains an endpoint that should be exposed only to administrators - GET /api/admin/v1/users/all. This endpoint returns the details of all the users of the application and does not implement function-level authorization checks. An attacker who learned the API structure takes an educated guess and manages to access this endpoint, which exposes sensitive details of the users of the application.

GOOGLE+ PRIVILEGE ESCALATION VULNERABILITY USER DATA FROM 52.5 MILLION ACCOUNTS EXPOSED

How was this hack perpetrated?

- "People: get" API endpoint was designed to let developers request basic information associated with a user profile.
- A software update in November 2018 introduced a privilege escalation (RBAC) vulnerability in the Google+ People API that allowed third-party app developers to view users' information even if a user profile was set to not-public.



What won't work in stopping these kinds of attacks

- SAST & DAST scanning solutions will not help detect these exploits as they focus on injection and fuzzing attacks rather than privilege escalation vulnerabilities.



What Can Attackers Do ...

- Try Resources names (admin, profile, accounts, search, pay etc ...)
- Try attributes names
- Try Content-Type
- Inject/Remove data
- Use answers to find data and plan next steps

<https://www.youtube.com/watch?v=qqmyAxfGV9c>

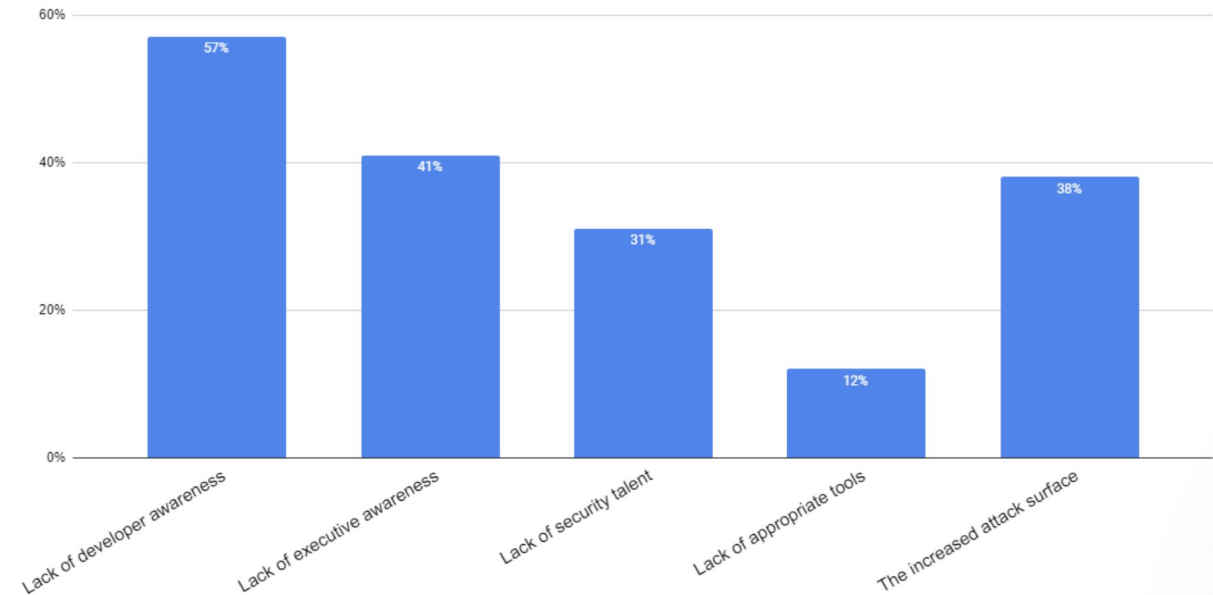


New “Old” AppSec World

Web Application Security is painful because the security is **not handled from beginning**

Developers **cannot define how the web application is built** and designed

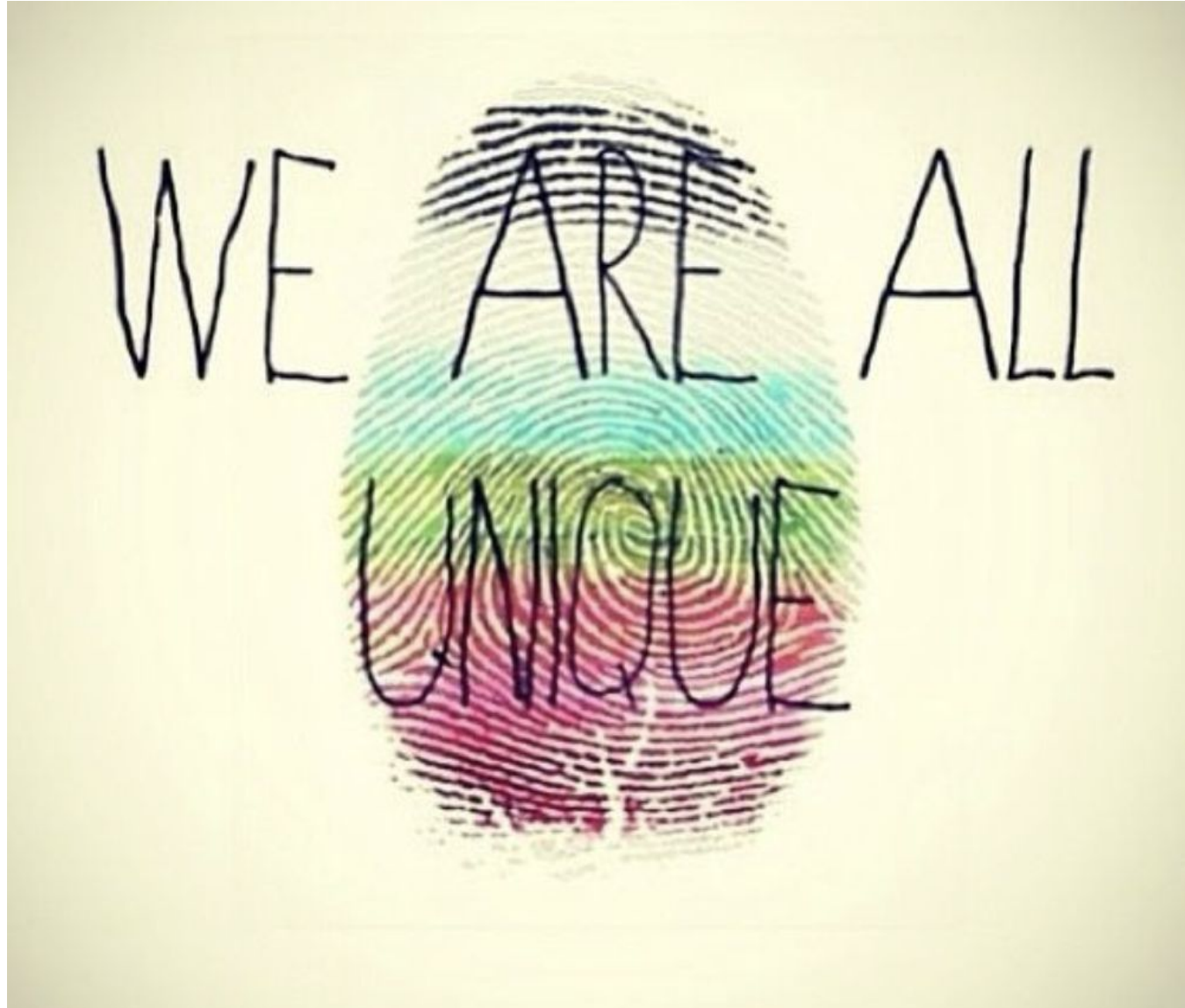
After 20 years of R&D, detection and protection tools have to **use AI to understand how the Web Application works...**

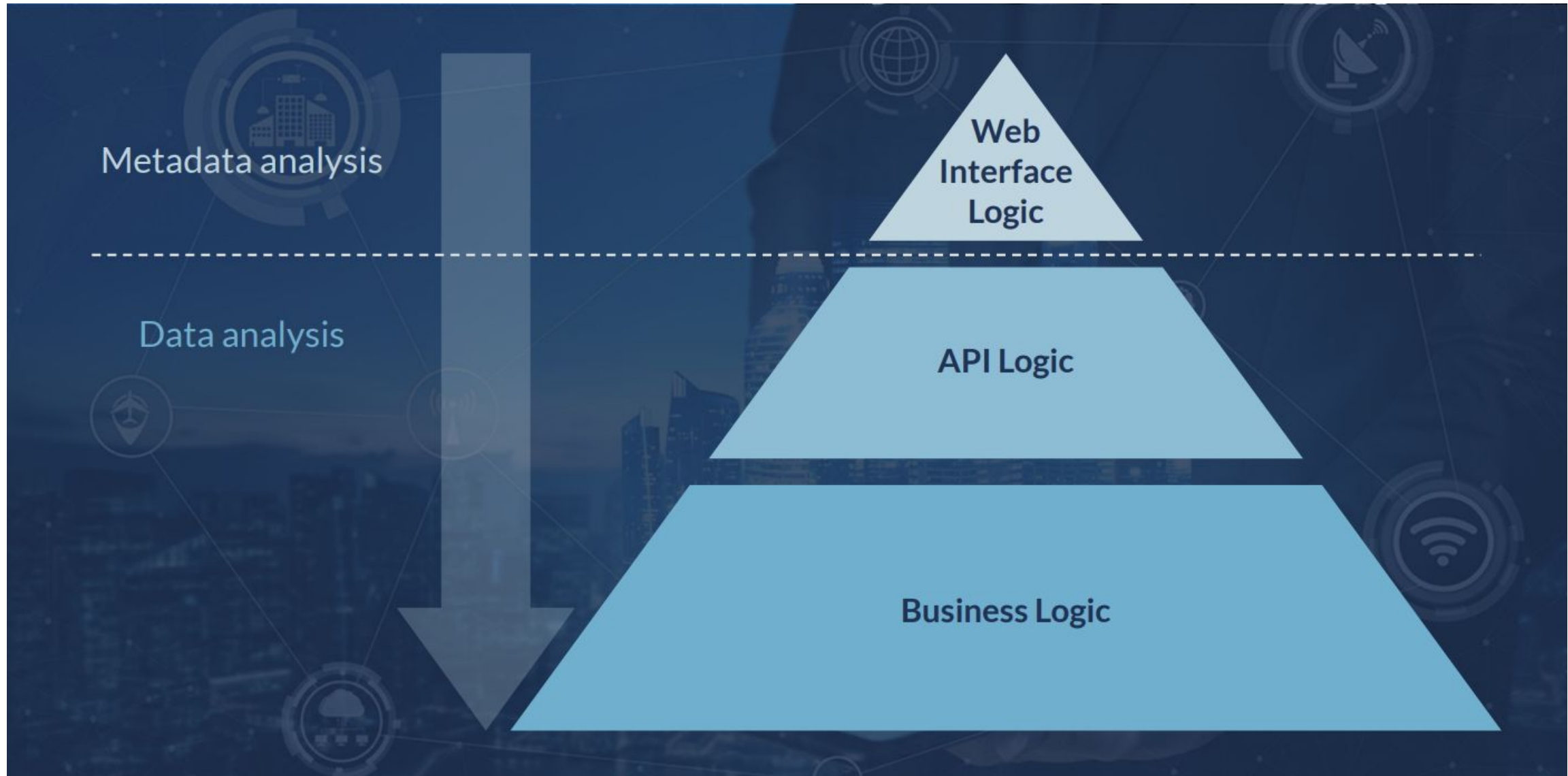


Security Requirements

DEEPSEC







Security Requirements - Why AI?

1. **High accuracy.** AI-based NLP technology automatically learns an API's business logic, going beyond metadata analysis by reviewing the actual API calls in the specific context. This approach focuses on prioritizing 'meaningful anomalies' - unusual behaviors with the potential of significantly impacting the business logic and indicating intent to manipulate the API.
2. **Prepares you for the unexpected.** While general-purpose application security solutions excel at detecting attacks that match known generic vulnerabilities, they fail when it comes to detecting zero-day, functional attacks. NLP-based security solutions learn each API's unique logic and detect any anomalous behavior that could be a functional attack.
3. **Capture different patterns in the API data.** Finding patterns in API data can be used for verifying whether any related transaction includes the required fields and alert when a transaction does not include one of these fields as an anomaly. Looking at API data as a dialogue enables users to look at the data from a sequential perspective, user clustering, and more. These are key patterns that help users understand the functional context in order to keep false positives down.
For example, NLP methods for [representation learning](#), in which words or phrases are mapped to some array of numbers taken as input categorical data, and learn a representation for each data value.
4. **Scale your protection.** Using statistical modelling to analyze the application behavior and spot deviations from baselines can be effective on a relatively small scale. But as the amount of traffic grows, it loses efficiency and false positives grow in proportion, thereby undermining scalability. Using NLP doesn't require comprehensive and ongoing maintenance to make sure all sensitive data is recognized and protected. It allows users to maintain very high accuracy at any scale because it discards that noise and focuses on meaningful anomalies.
5. **Knowing the right context.** NLP enables security analysts to explain the meaning of specific anomalies given the objects on which they occurred, their characteristics, the relationship being manipulated, the users, and more. Essentially, this results in faster remediation and better collaboration.



✓ Started from app filters as a security control.
Embedded WAF is an example of security filter

Filters

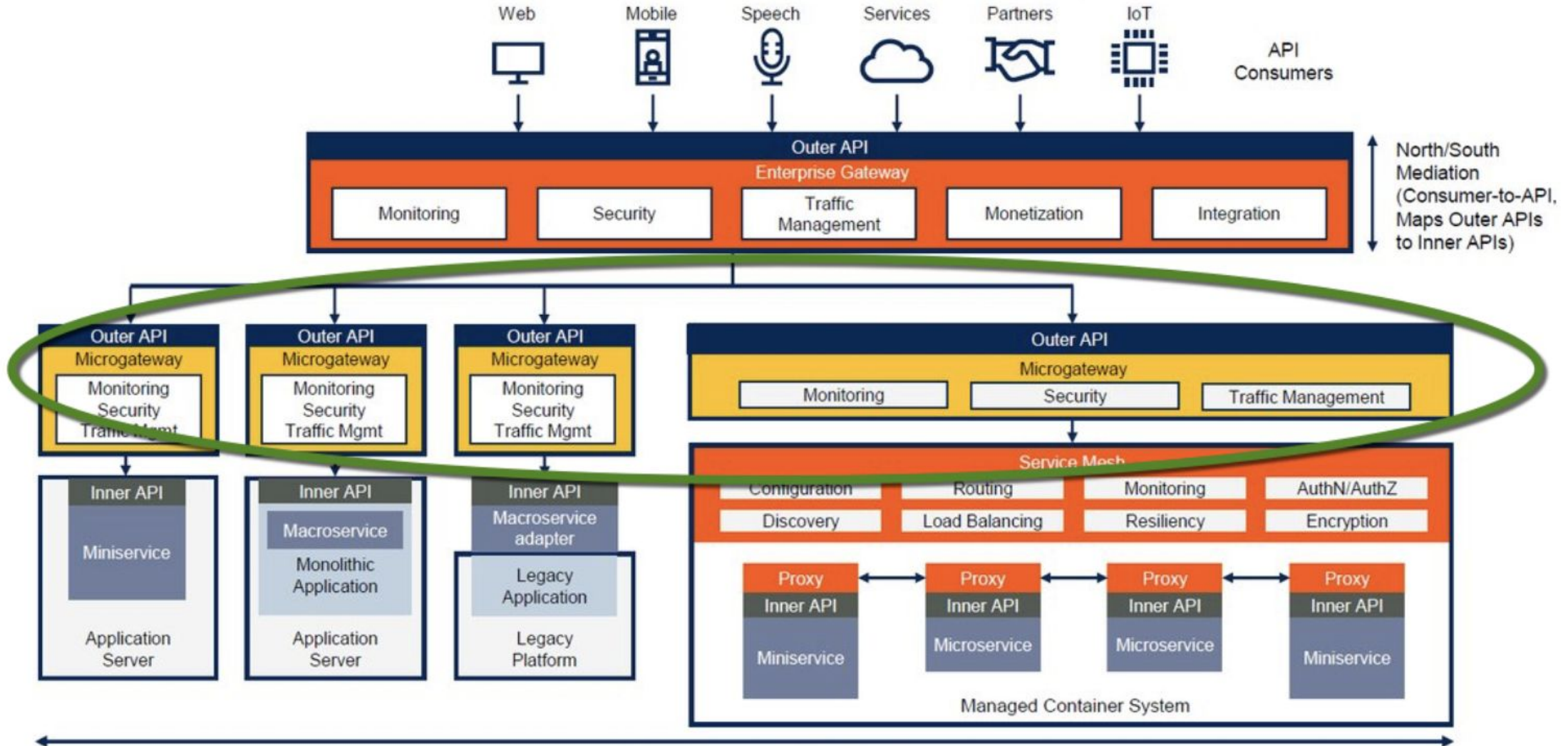
Single Gateway

Distributed
(Microgateways)

✓ Distributed Microgateways provides flexibility in
configuration, remove complexity from code e.c.t



Deployments (Where to protect)



What about API Gateway & WAF?



WAF - introduced back in the early 1990s, uses pattern recognition. based on manual rules, can't detect zero-day exploits. leading vendors are F5, Akamai, Imperva. Typical error rate >30%



Applicative DDoS protection - introduced back in the early 2000s, evolving from volumetric DDoS. Based on thresholds and rules, manual operations. Leading vendors are Radware, Akamai, Imperva, F5, Cloudflare. Typical error rate >30%



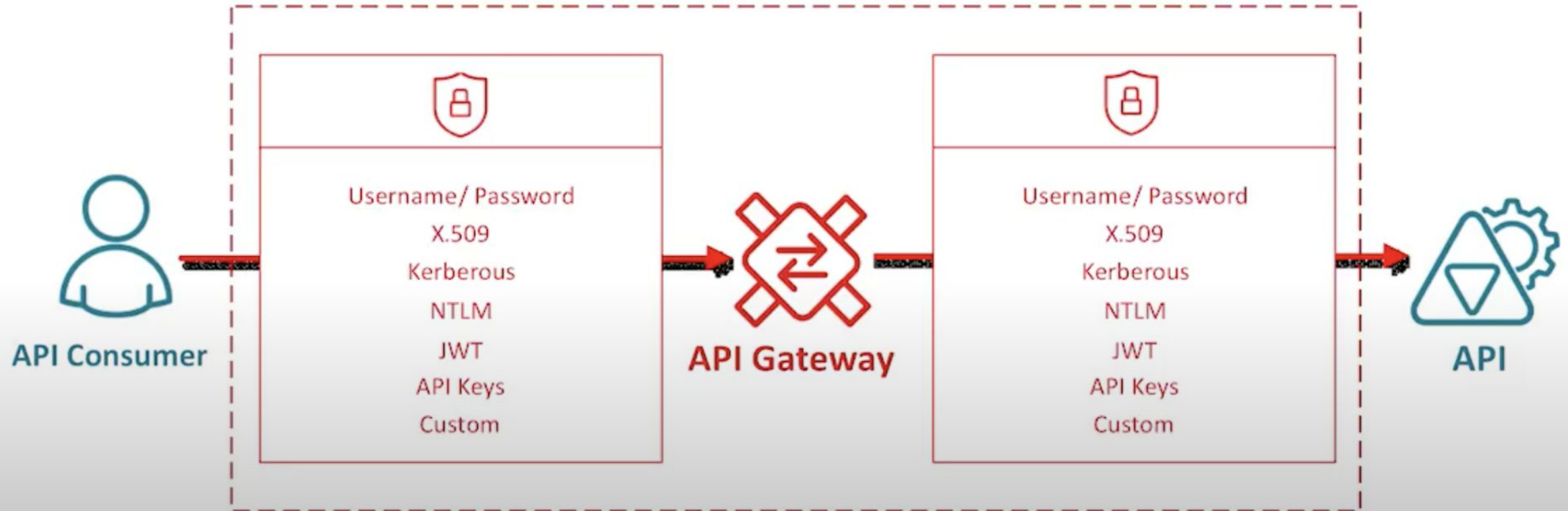
BOT protection - introduced back in the early 2010s, leading companies are Perimeter X, Distill networks. Akamai, client-side detection that can be easily bypassed.



API Gateway - Sits between a client and a collection of backend services performing user authentication, rate limiting, and statistics. "Control tower" like function

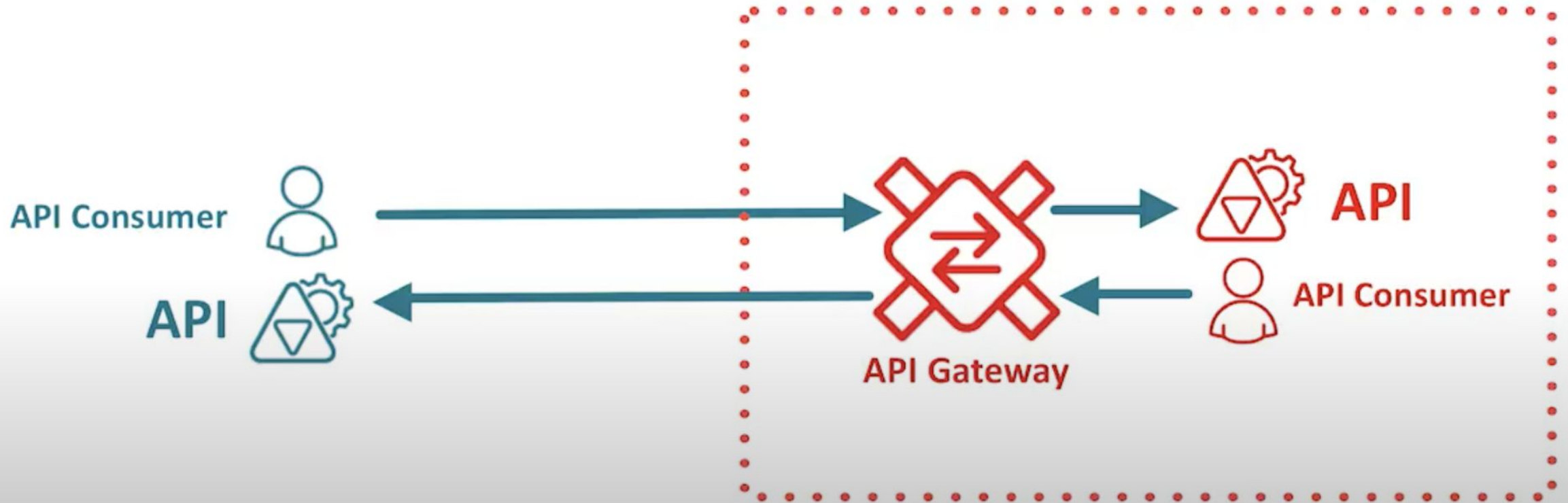


What about API Gateway



What about API Gateway

DEEPSEC



BOLA, Mass Assignment & API Gateway

DEEPSEC

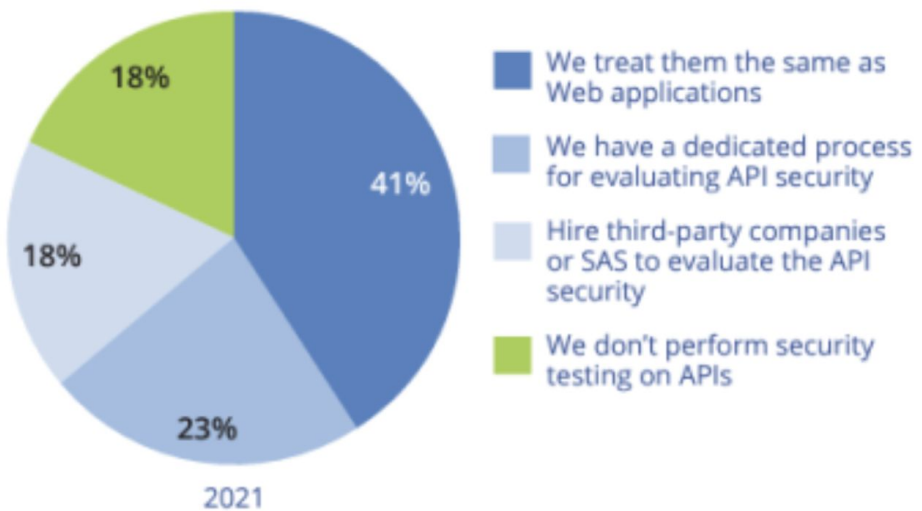
Legitimate - Client sends a legitimate request	Attack - Attackers sends the same request but adds the admin role in the request body
PUT /api/v2/users/5deb9097 HTTP/1.1 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36 X-Forwarded-For: 19.42.129.253 { "_id": "5deb9097", "address": "*****, NY City, NY", "company_role": "Investment Services", "email": "*****", "first_name": "*****", "full_name": "*****", "job_title": "Broker", "last_name": "*****", "phone_number": "*****" }	PUT /api/v2/users/5deb9097 HTTP/1.1 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36 X-Forwarded-For: 19.42.129.253 { "_id": "5deb9097", "address": "*****, NY City, NY", "company_role": "Investment Services", "email": "*****", "first_name": "*****", "full_name": "*****", "is_admin": true, "is_sso": true, "job_title": "Broker", "last_name": "*****", "permission_type": "admin", "phone_number": "*****", "role": "admin", "sso_type": "admin", "system_user_type": "admin", "system_user_type_cd": 2, "user_type": "admin", "user_type_cd": 10 }

Legitimate - userId matches in the query parameter and request	Attack - Attacker changes the userId in the query parameter
Request: GET /v1/customers/15981?userId=207939055 HTTP/1.1 Authorization: Bearer gwwh1Y4epjv9Y Cookie: _ga=GA1.3.630674023.1502871544; _gid=GA1.2.1579405782.1502871544;userId=207939055 Host: payments-api.dnssf.com X-Forwarded-For: 54.183.50.90	Request: GET /v1/customers/15981?userId=207938044 HTTP/1.1 Authorization: Bearer gwwh1Y4epjv9Y Cookie: _ga=GA1.3.630674023.1502871544; _gid=GA1.2.1579405782.1502871544;userId=207939055 Host: payments-api.dnssf.com X-Forwarded-For: 54.183.50.90
Response: 200 OK { "userId": 207939055, "firstName": "John", "lastName": "Smith", "email": "john.smith@acme.com", "phoneNumber": "+1650123123" }	Response: 200 OK { "userId": 207938044, "firstName": "David", "lastName": "Miller", "email": "david.miller@example.com", "phoneNumber": "+1912456456" }

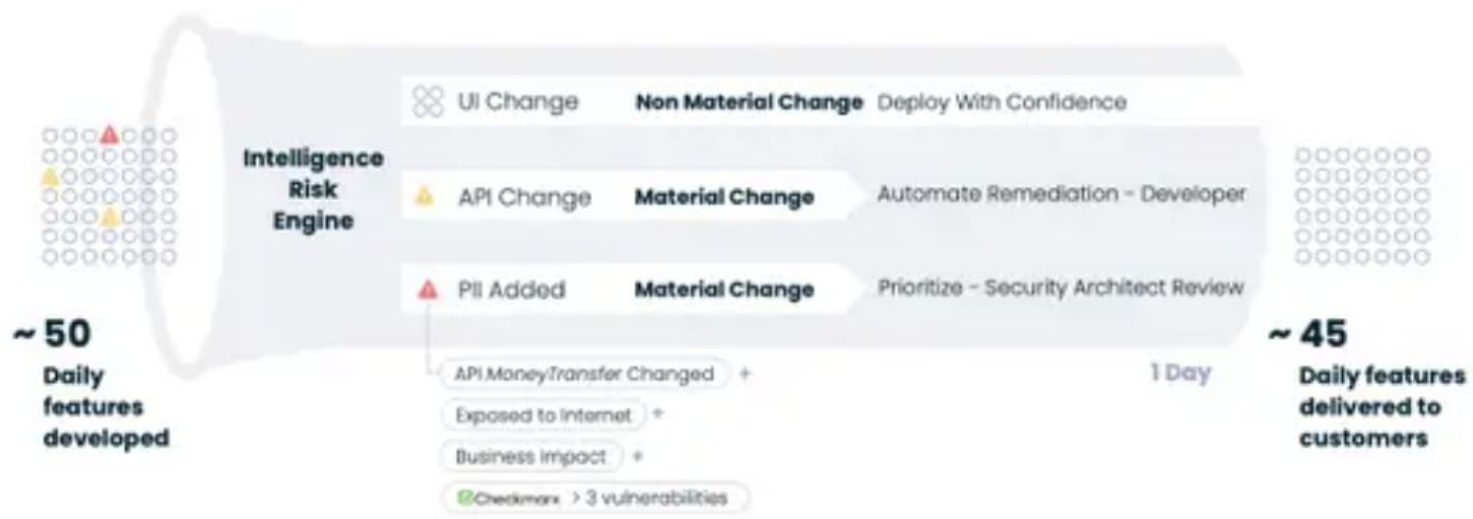
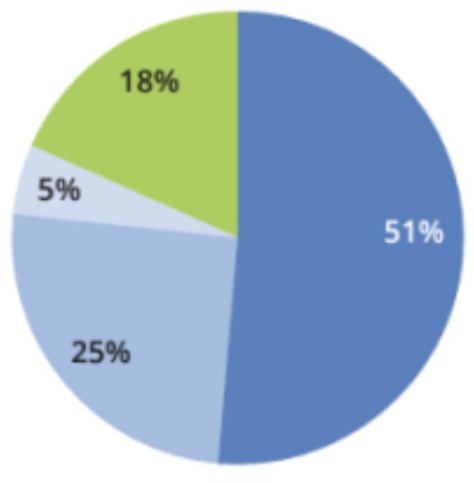


Security of APIs

How are you handling security of APIs?



Data: Dark Reading survey of 173 IT and cybersecurity professionals in Feb



“Positive Model”

DEEPSEC



Access **Denied** by
default



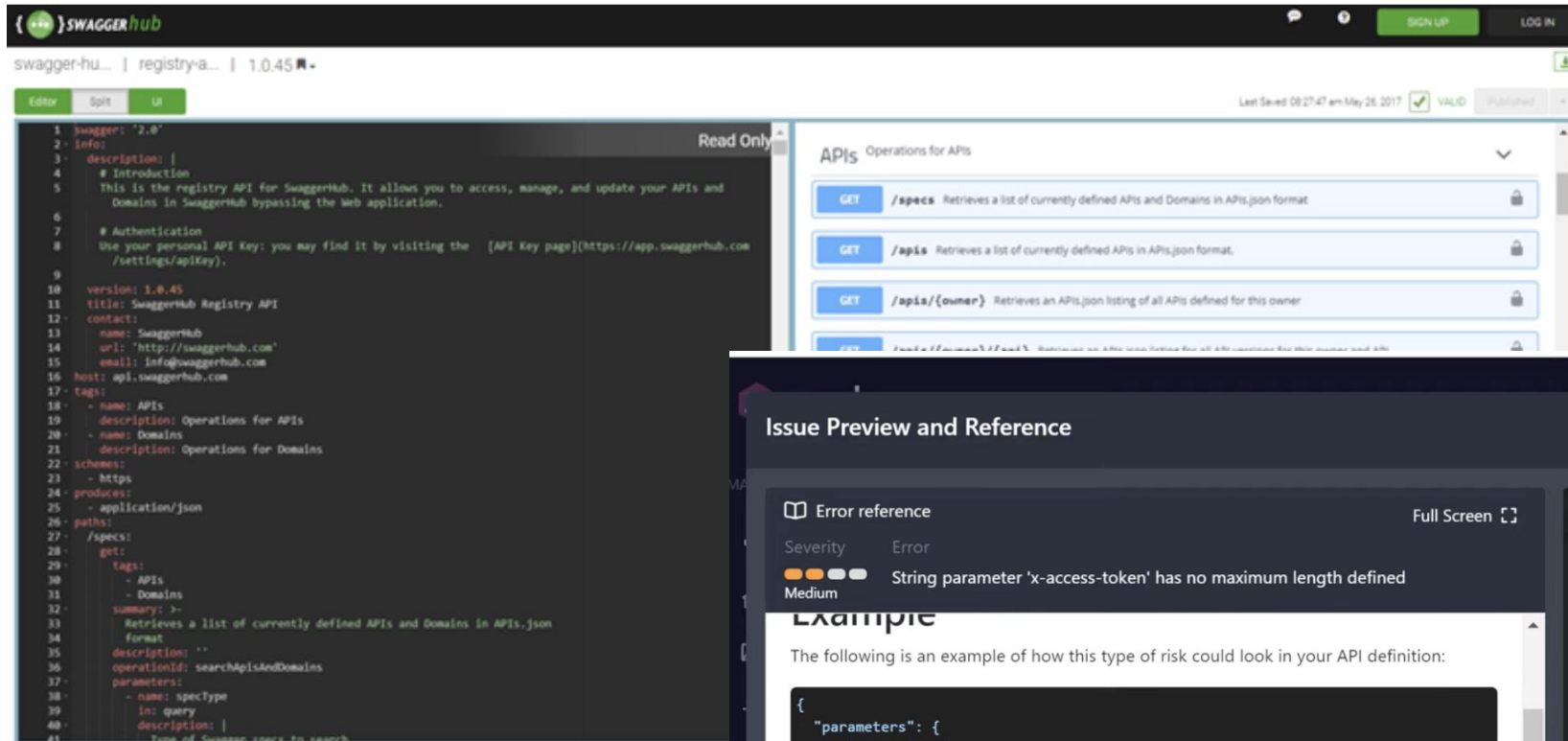
Allow Access only
to **approved**
traffic



Trust centric



OAS (OpenAPI Specification)



Issue Preview and Reference

Issue 1 of 7 < Previous Issue Next Issue > Close Window

Error reference

Full Screen

Severity Error

Medium String parameter 'x-access-token' has no maximum length defined

Example

The following is an example of how this type of risk could look in your API definition:

```
{  "parameters": {    "in": "query",    "name": "id",    "type": "string",    "description": "Identifier of the object to be extracted."  }}
```

Possible exploit scenario

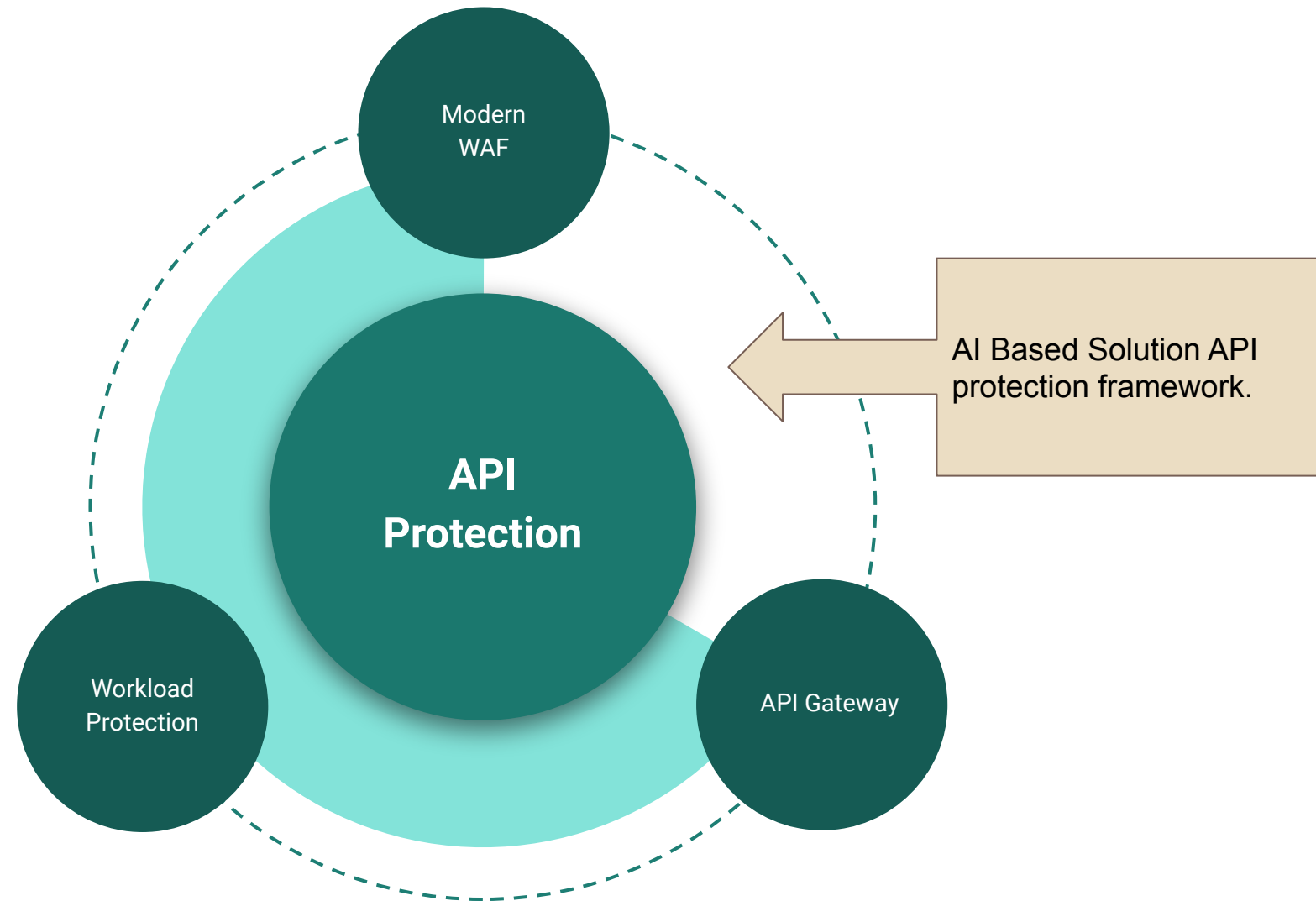
If you do not limit the length of strings, attackers can send longer strings to your API than what your backend server can handle. This could overload your backend server and make it crash. In some cases, this could cause a buffer overflow and allow for executing arbitrary code. Long strings are also more prone to injection attacks.

Your code

```
310      "description": "No token provided or invalid token."
311    },
312  },
313  "tags": [
314    "admins"
315  ],
316  "produces": [
317    "application/json"
318  ],
319  "parameters": [
320    {
321      "in": "header",
322      "name": "x-access-token",
323      "type": "string",
324      "required": true,
325      "pattern": "^[a-zA-Z0-9_]{4,}\\.[a-zA-Z0-9_]{4,}\\.[a-zA-Z0-9_]{4,}$"
326    },
327  ],
328  "summary": "Returns the list of ALL users. Must be admin to call.",
329  "description": "Returns the list of ALL users. Must be admin to call.",
330  "operationId": "adminallusers"
```

Score impact 4

Fix Issue in Editor



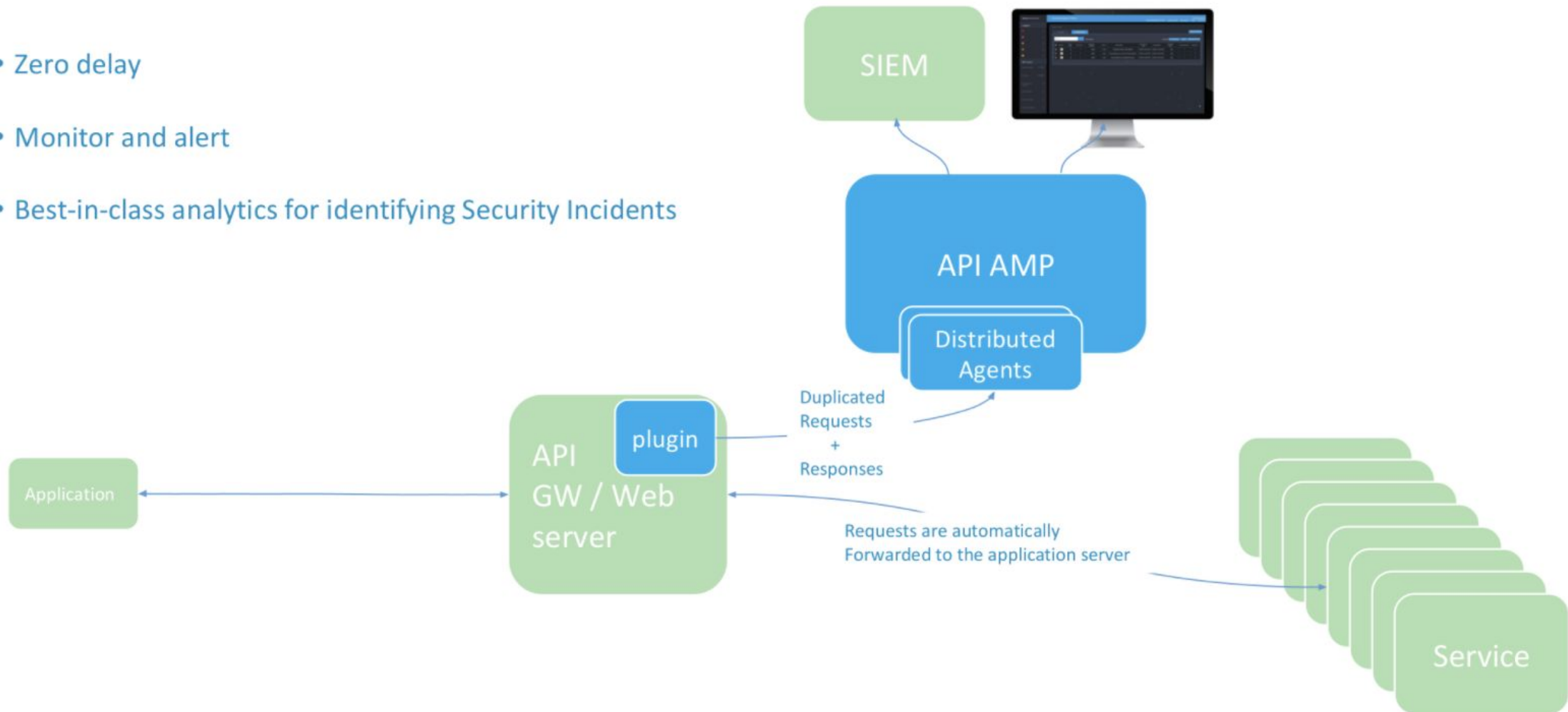
What we should worry about?

- It's all about statistic
- Learn through legitimate traffic (Cannot stop from the first bad request!)
- Policy creation/correction for Every API!
- Take an action (Protect) takes time
- Integrations



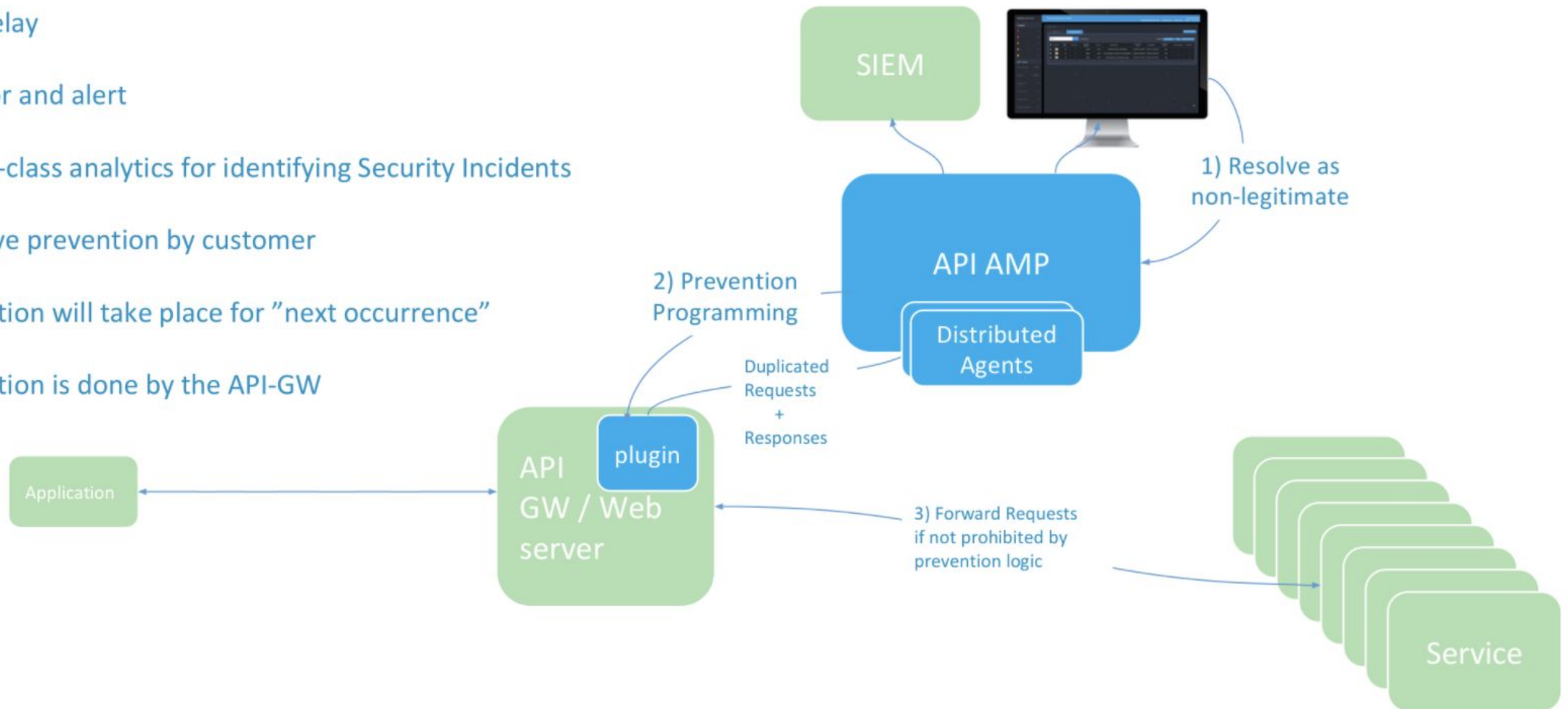
Monitoring only architecture

- Zero delay
- Monitor and alert
- Best-in-class analytics for identifying Security Incidents



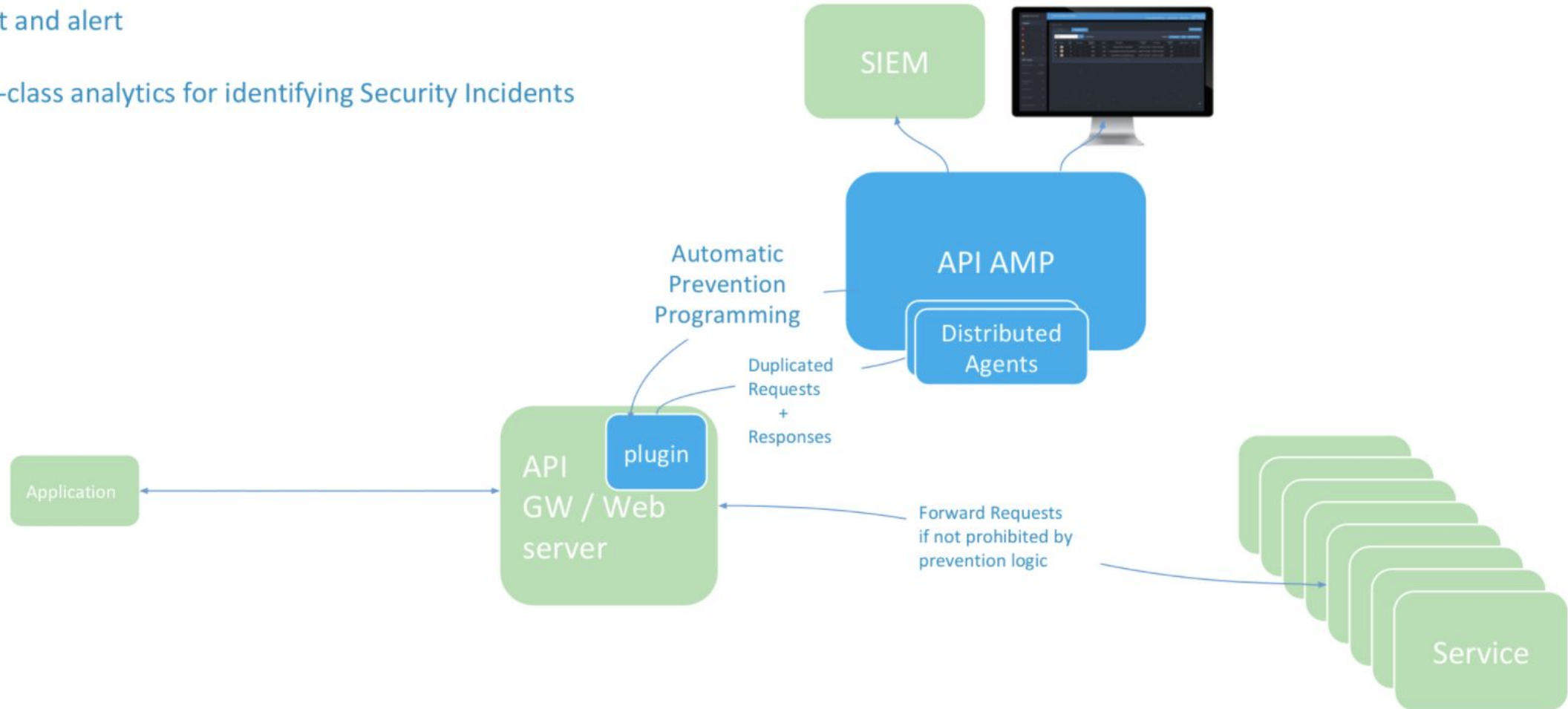
Semi-automatic prevention architecture

- Zero delay
- Monitor and alert
- Best-in-class analytics for identifying Security Incidents
- Selective prevention by customer
- Prevention will take place for "next occurrence"
- Prevention is done by the API-GW



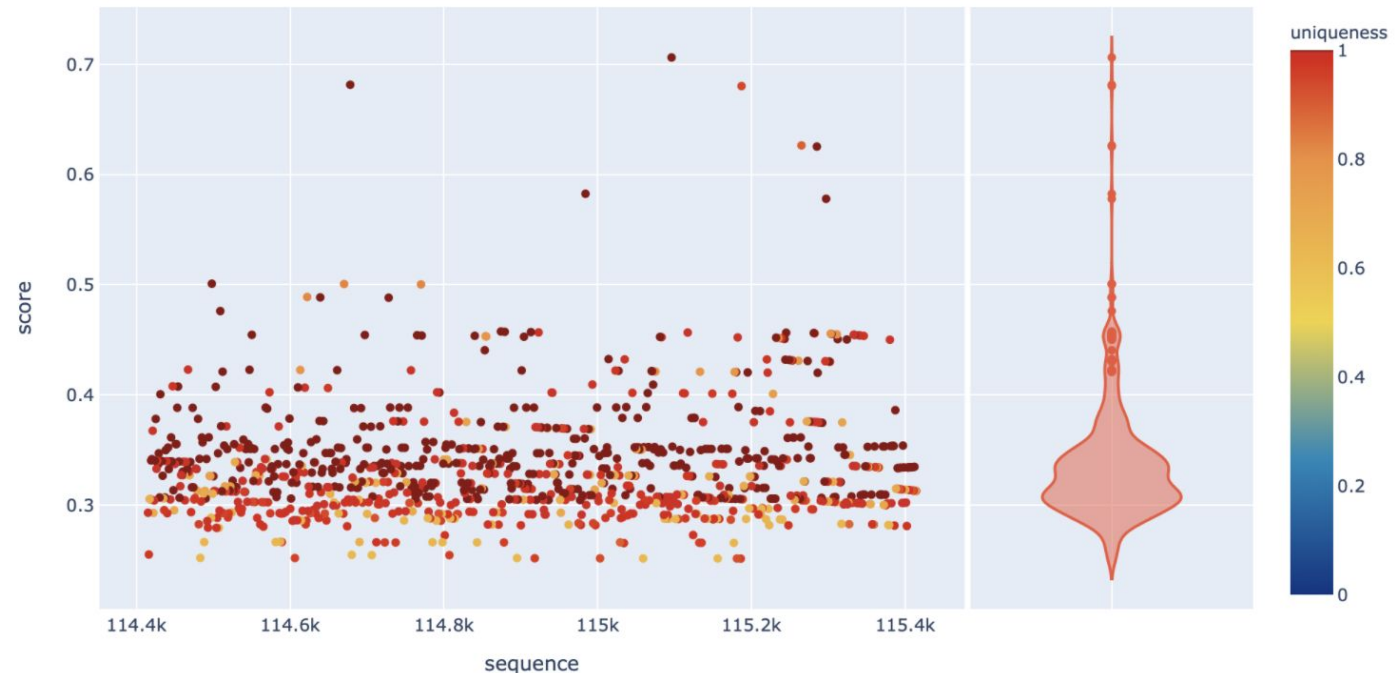
Automatic prevention architecture

- Prevent and alert
- Best-in-class analytics for identifying Security Incidents



Anomalies Detection

- Self-explanatory Anomaly Descriptions
- Use NLP to auto-generate self-explanatory report of the abnormal API anomaly event and incident
- Make it accessible/understandable for anyone with basic knowledge of the domain (e.g., Financial service)



"What Did You Say ?"

DEEPSEC

Anomaly Management Platform

Anomaly Management CenterDiscoveryManagementLogoutprogrammer

Home > Events > 67d0c8f7-8c04-40e3-8d4a-b6d44458e5d9

4

Event Number: 3

Detection Type: Inconsistent usage

Creation time: 11/25/2019 6:04:21 PM

Update date-time: 11/25/2019 6:04:21 PM

State: Active

Activity: Ended

Occurrences: 1

< Previous

Next >

Raw packet

Resolve

Delete

Event description:

API call description: [Submit account recovery code verify](#)
Anomaly description: [API usage inconsistency across sequence of 'submit account recovery code verify' API calls](#)
Detected behavior: ['Submit account recovery code verify' API call was used for the same device_id with 9 Inconsistent values](#)

Recommended counter measure:
Rate Limit 'submit account recovery code verify' API calls

Event details:

Application: [i.instagram.com/api](#)
Method: [POST](#)
Exchange: [Request](#)
Host: [i.instagram.com](#)
Endpoint: [/api/v1/accounts/account_recovery_code_verify](#)
Session ID: [N/A](#)
Consumer ID: [AKIAJZELQ2XJL0MWKRZA](#)

Inconsistent values: Too many different values of recover_code for device_id

No.	Key field name	Key value	Deviating field	Detected value	Response status
2	device_id	8C21DA6BC4CA07F2	recover_code	000003	403 Forbidden
3	device_id	8C21DA6BC4CA07F2	recover_code	000001	403 Forbidden
4	device_id	8C21DA6BC4CA07F2	recover_code	100002	403 Forbidden
5	device_id	8C21DA6BC4CA07F2	recover_code	999999	403 Forbidden
6	device_id	8C21DA6BC4CA07F2	recover_code	100003	403 Forbidden
7	device_id	8C21DA6BC4CA07F2	recover_code	999997	403 Forbidden
8	device_id	8C21DA6BC4CA07F2	recover_code	100001	403 Forbidden
9	device_id	8C21DA6BC4CA07F2	recover_code	000002	403 Forbidden

User Comments

Add



API Auto Discovery

Anomaly Management Platform

[Anomaly Management Center](#) [Discovery](#) [Management](#) [Logout](#) programmer

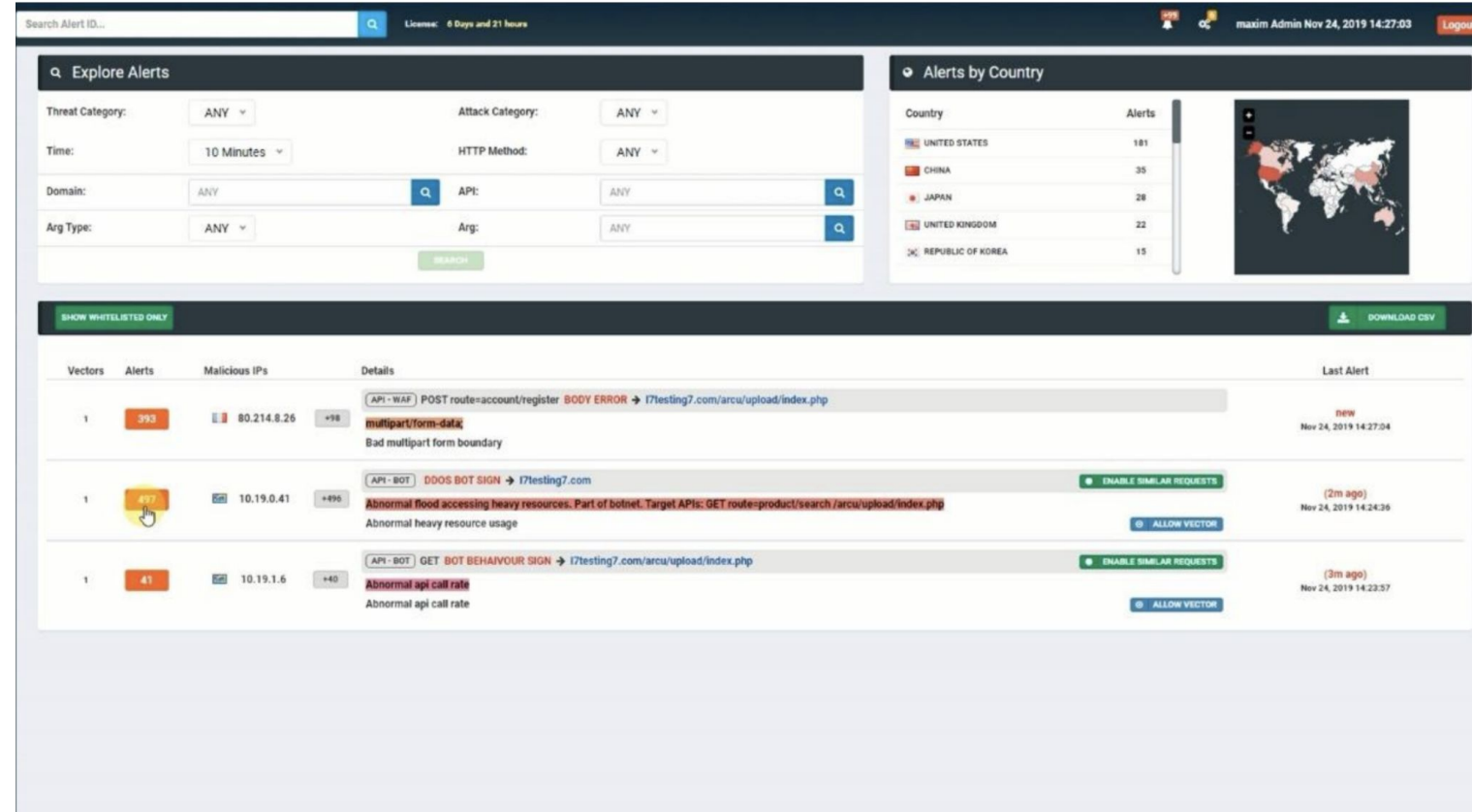
```
▼ api.openbankproject.com/obp/v3.1.0
  > /my/banks/{PARAM_1}/accounts/{PARAM_2}/transactions
  > /cards
  > /banks/{PARAM_1}/cards
  > /banks/{PARAM_1}/atms
  ▼ /banks/{PARAM_1}/accounts/{PARAM_2}/funds/transaction-request-types/COUNTERPARTY/transaction-r...
    ▼ POST
      Request Fields
      Response Fields
      Request Headers
      Response Headers
      Consumers
  > /banks/{PARAM_1}/accounts/{PARAM_2}/funds/transaction-request-types
  > /banks/{PARAM_1}/accounts/{PARAM_2}/funds/other_accounts
  > /banks/{PARAM_1}/accounts/{PARAM_2}/funds/funds-available
  > /banks/{PARAM_1}/accounts/{PARAM_2}/funds/checkbook/orders
  > /banks/{PARAM_1}/accounts/{PARAM_2}
  > /banks/{PARAM_1}/accounts/private
  > /banks/{PARAM_1}/accounts
  > /banks
```

Field name	Data type	Sensitive data
challenge.allowed_attempts	Number	<input type="checkbox"/>
challenge.challenge_type	String	<input type="checkbox"/>
challenge.id	String	<input type="checkbox"/>
charge.summary	String	<input type="checkbox"/>
charge.value.amount	Number	<input type="checkbox"/>
charge.value.currency	String	<input type="checkbox"/>
details.description	String	<input type="checkbox"/>
details.to_counterparty.counterparty_id	String	<input type="checkbox"/>
details.to_sandbox_tan.account_id	String	<input type="checkbox"/>
details.to_sandbox_tan.bank_id	String	<input type="checkbox"/>
details.to_sepa.iban	String	<input checked="" type="checkbox"/> Sensitive
details.to_transfer_to_account.description	String	<input type="checkbox"/>
details.to_transfer_to_account.future_date	Number	<input type="checkbox"/>
details.to_transfer_to_account.to.account.iban	String	<input checked="" type="checkbox"/> Sensitive
details.to_transfer_to_account.to.account.number	String	<input checked="" type="checkbox"/> Sensitive
details.to_transfer_to_account.to.bank_code	String	<input type="checkbox"/>
details.to_transfer_to_account.to.branch_number	String	<input type="checkbox"/>
details.to_transfer_to_account.to.name	String	<input checked="" type="checkbox"/> Sensitive
details.to_transfer_to_account.transfer_type	String	<input type="checkbox"/>
details.to_transfer_to_account.value.amount	Number	<input type="checkbox"/>
details.to_transfer_to_account.value.currency	String	<input type="checkbox"/>
details.to_transfer_to_atm.description	String	<input type="checkbox"/>
details.to_transfer_to_atm.from.mobile_phone_number	String	<input checked="" type="checkbox"/> Sensitive



"Connect The Dots" - Visibility

- Smart analytic layer which group and classify API anomalies into intent-driven Attack incident.
- Convert anomalies into the actionable incident by grouping multiple-anomalies with a common attack denominator into an incident.



Attacker "Signature"

< Attacker Details

All Attackers > 60a5c0514c000033d8ea1f3e

MAIN SOURCE TYPE	LAST ACTIVITY	API NAME
IP	2 Minutes ago	Fishhatchery 🔗
MAIN SOURCE ID	ATTEMPTS	
107.2.118.249	14	

Open

Take Action

DESCRIPTION

Write a comment...

Summary Sources Timeline

Endpoints Most Attempted

/api/user/{userId}	7	(50%)	
/api/cart/{cartId}	2	(14%)	
/api/cartItems	1	(7%)	
/api/order	1	(7%)	
/api/cartItems/{cartItemId}	1	(7%)	
/api/customerAddress	1	(7%)	

Server Responses

400 Bad Request	5
404 Not Found	4
500 Internal Server Error	2
200 OK	1
401 Unauthorized	1
405 Method Not Allowed	1

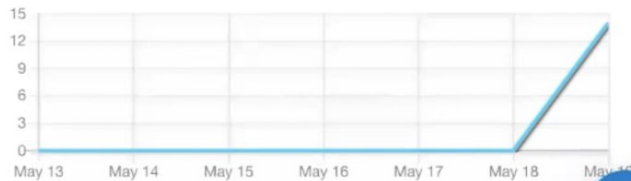


Attack Types ⓘ

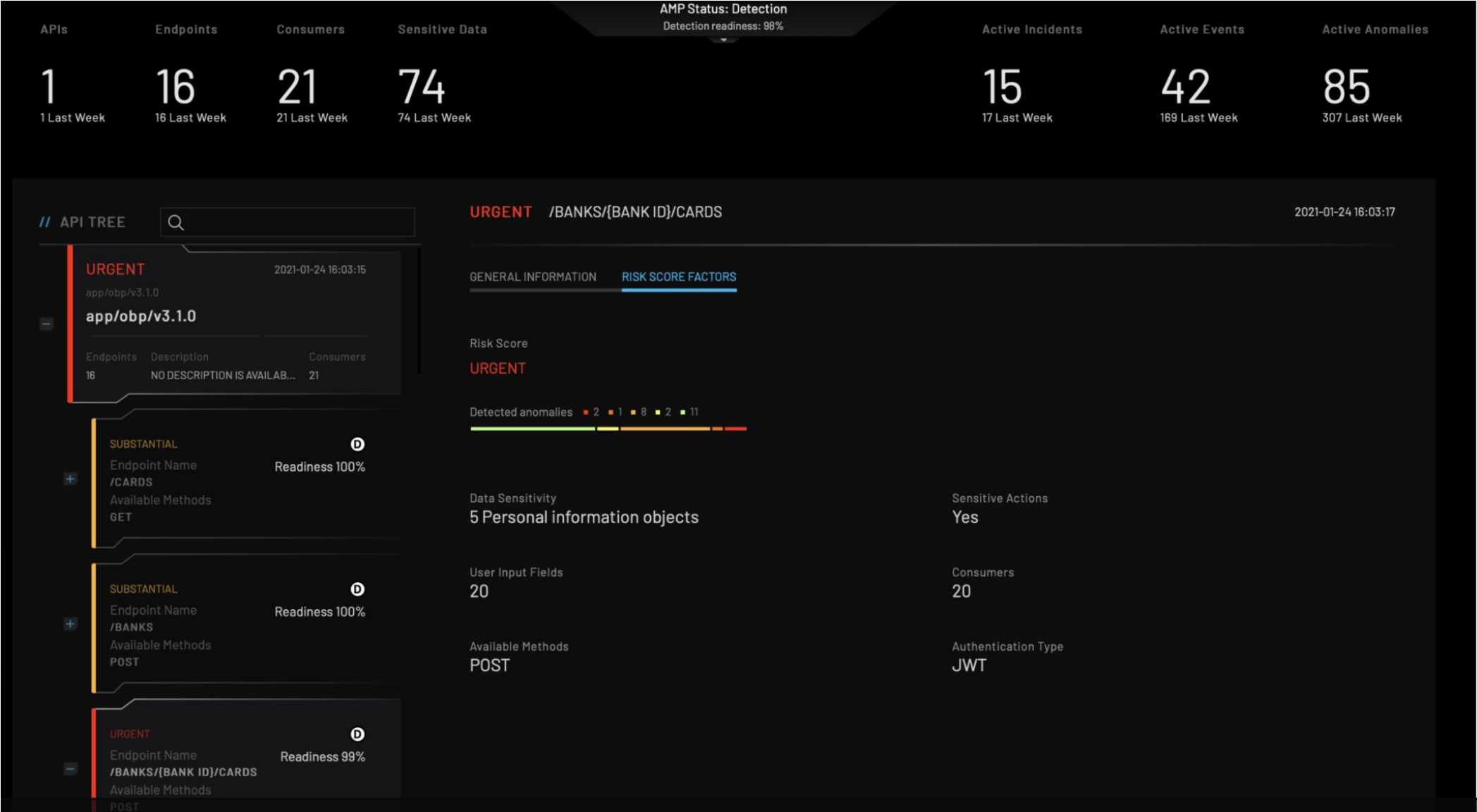
OWASP API1 - Potential Broken Object Level Authorization	3
OWASP API2 - Broken Authentication	1
OWASP API5 - Potential Broken Function Level Authorization	2
OWASP API6 - Mass Assignment	5
OWASP API8 - Potential Code Execution Attempt	1
Parameter Tampering	8



Number Of Attempts

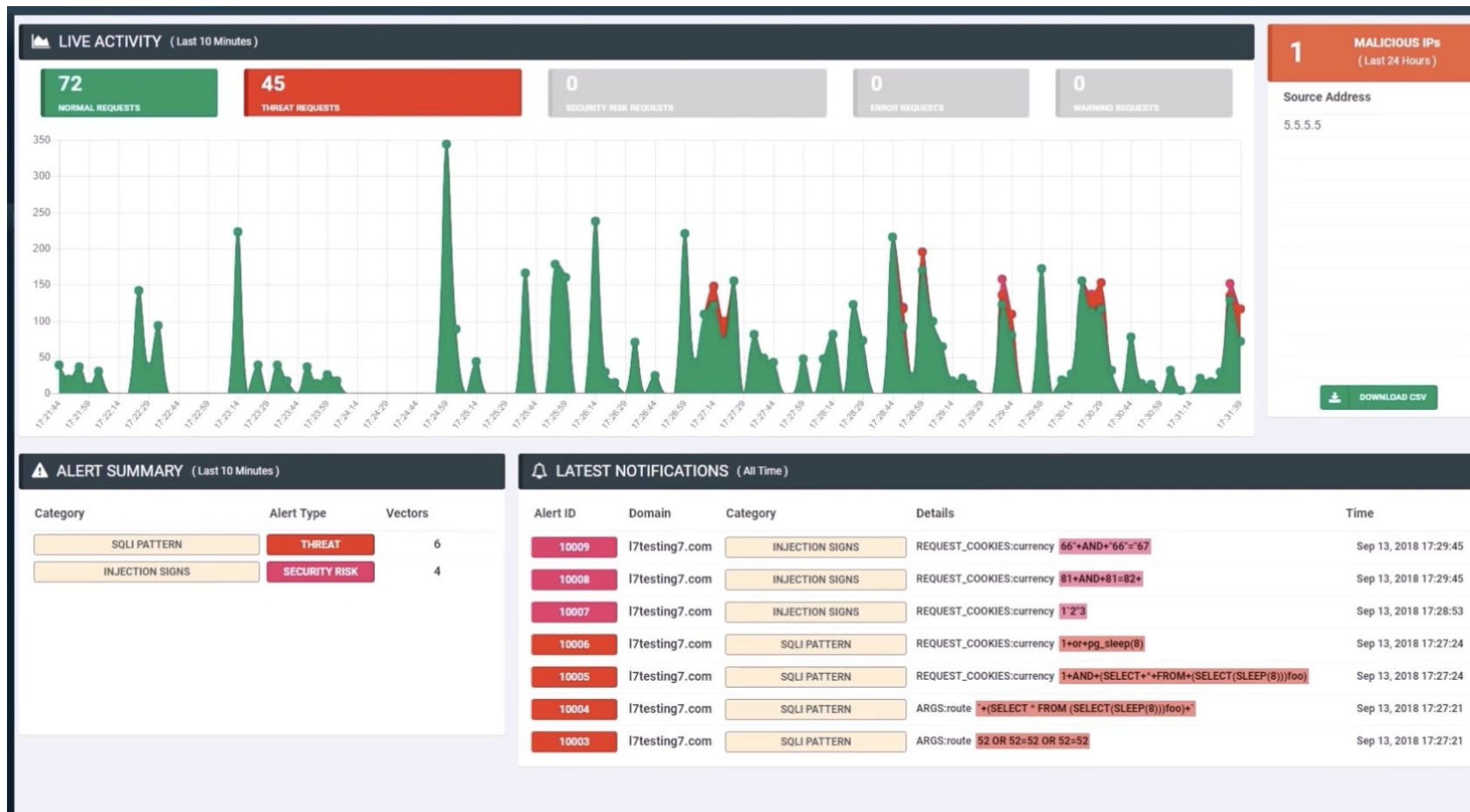


Anomalies Visibility (Find the Risks)

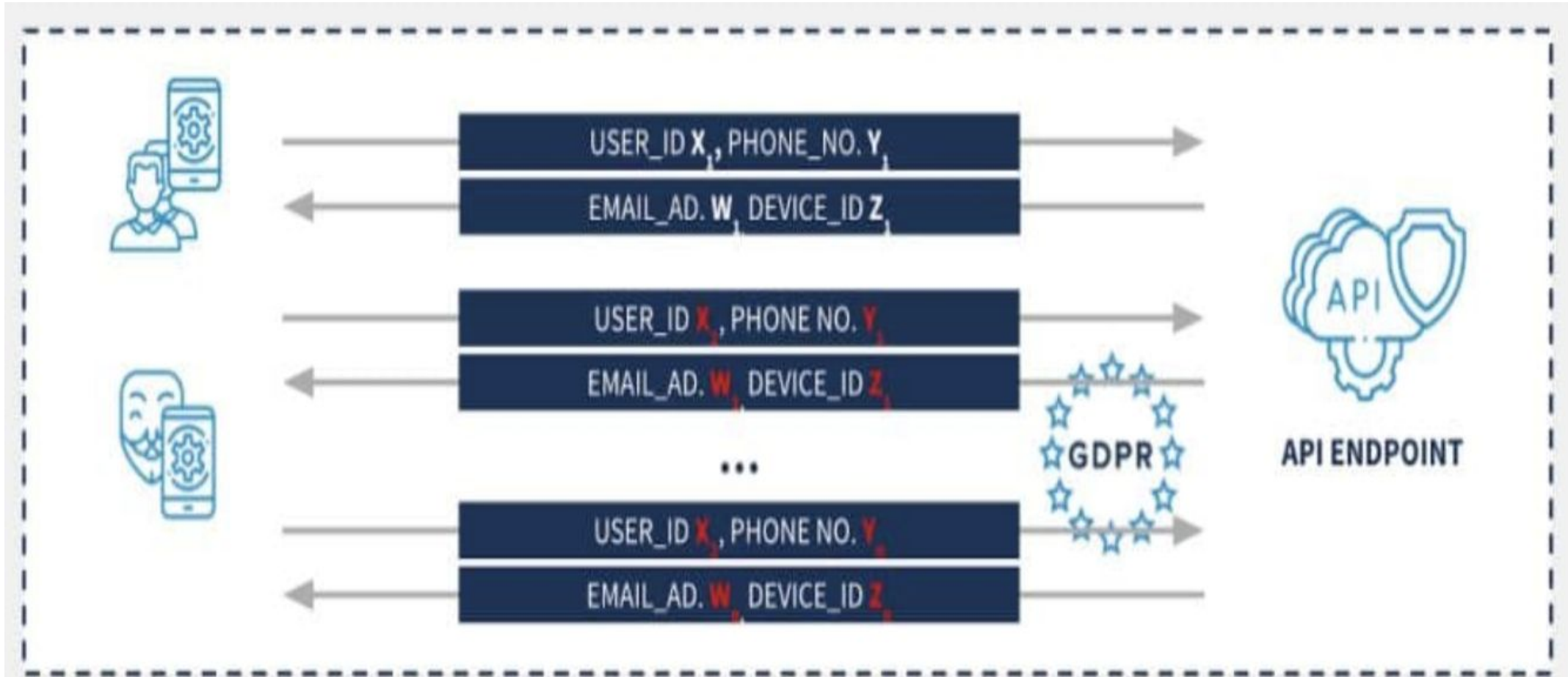


Early detection - Probe Phase

- Taking advantage of “trial and error” hacking patterns
- Identify “attempts-to-attack” patterns and alert/block the root cause before a successful attack was conducted



“Important” Content Detection



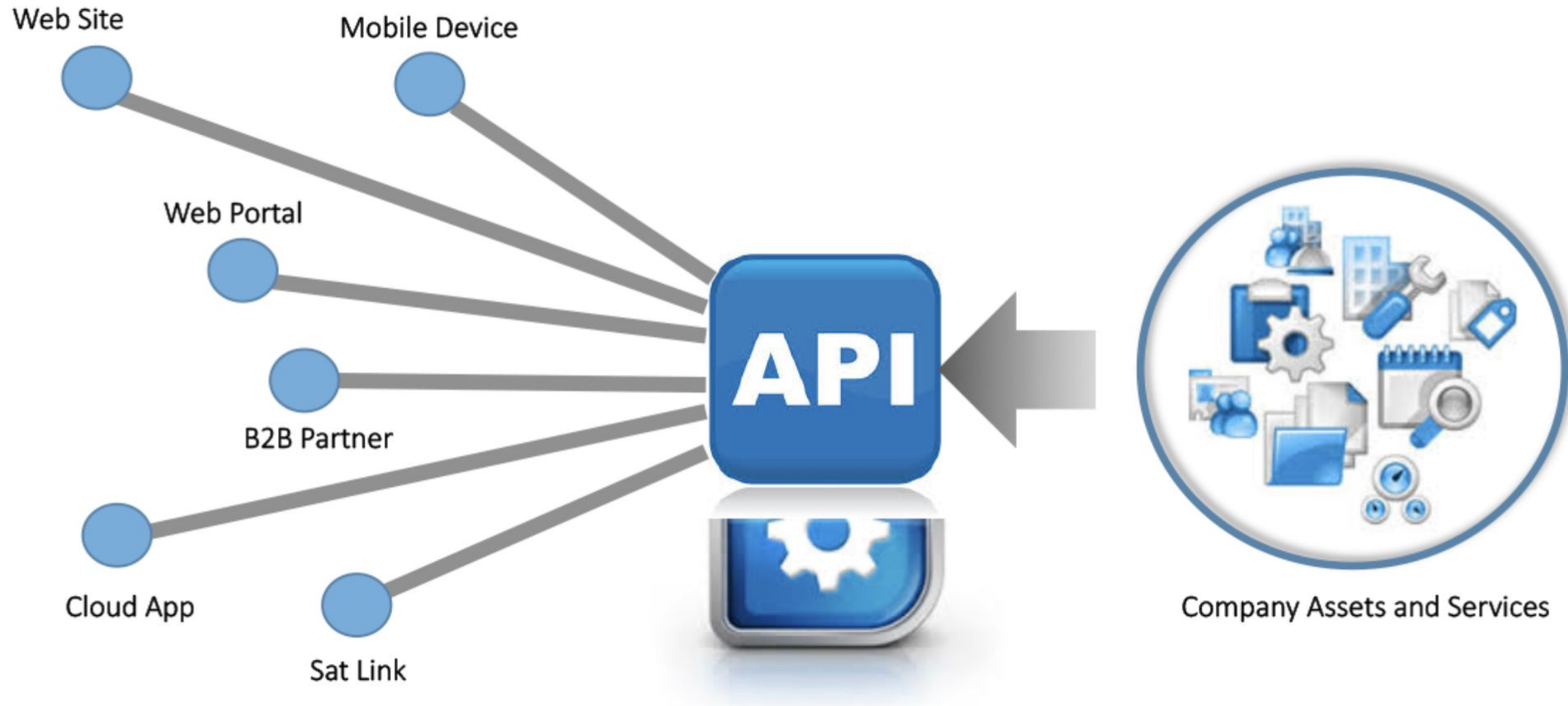
Security Policy as a Code

```
var curr_api = getApiData("/v1/hg654/users/*")÷  
var metadata = curr_api.getMetadata()÷  
if(metadata.containsPII(0)){  
    dashboard.alert("Add PII data is forbidden for this endpoint : " + curr_api.getMetadata().getId());  
}else{  
  
    //Continue to validate  
    this.Continue(curr_api);  
  
}●  
|
```



Bi-Directional Security (Next Step)

DEEPSEC



Vendors (Example)

DEEPSEC



IMVISION

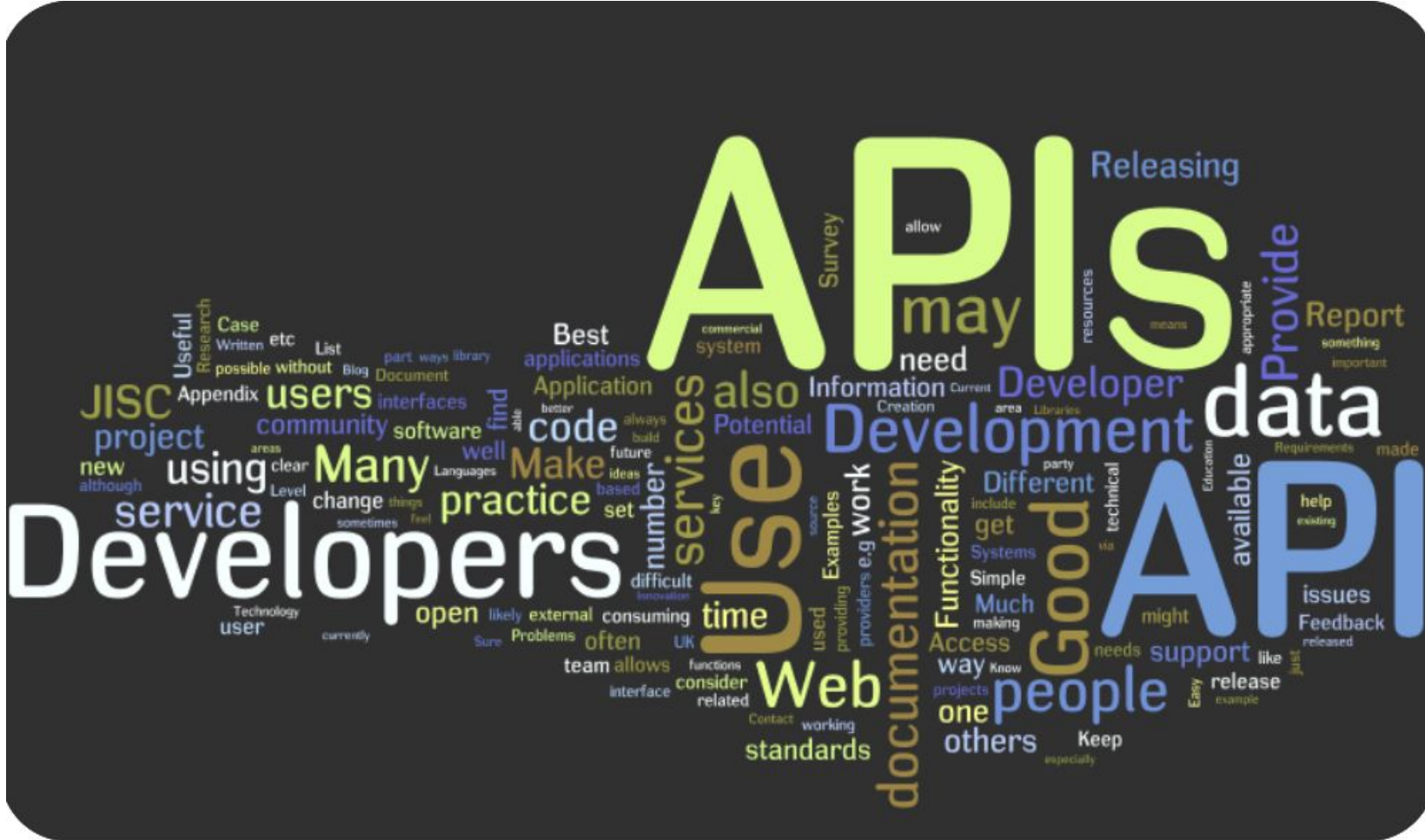


42 crunch

 **Reblaze**



Conclusion

The logo for DEEPSEC, featuring the word "DEEP" in white on a dark red background and "SEC" in dark blue on a white background.

Thank You!

vitalyd@jfrog.com

LinkedIn: <https://www.linkedin.com/in/vitaly-davidoff-07039a1>

