Pushing the SOC Left To Achieve Nash Equilibrium

Presented By: O’Shea Bowens of Null Hat Security
O’Shea Bowens

• Grew up in the SOC
• Founder & CEO of Null Hat Security
• CSOC Manager at Toast
• Founder of SKICON
• Co-Founder of IDS Podcast

Affiliations
• Boston Security Meetup Organizer
• DC617
• ISSA Board of Directors
• Twitter:@sirmudbl00d
• In game theory, the Nash equilibrium, named after the mathematician John Forbes Nash Jr., is a proposed solution of a non-cooperative game involving two or more players in which each player is assumed to know the equilibrium strategies of the other players, and no player has anything to gain by changing only their own strategy.

• In terms of game theory, if each player has chosen a strategy, and no player can benefit by changing strategies while the other players keep theirs unchanged, then the current set of strategy choices and their corresponding payoffs constitutes a Nash equilibrium.
Agenda

• Primary Objectives of the SOC

• Where Did The SOC Go Wrong

• How Do We PSH Left

• Questions
Primary Objectives of the SOC

People- Every employee needs to be aware of their role in preventing and reducing cyber threats, and specialised technical cyber security staff needs to stay fully up to date with the latest skills and qualifications to mitigate and respond to cyber-attacks.

Process- Processes are crucial in defining how the organisation’s activities, roles and documentation are used to mitigate the risks to the organisation’s information. Threats change quickly, so processes need to be continually reviewed to be able to adapt with them.

Technology is essential to giving organizations and individuals the computer security tools needed to protect themselves from cyber-attacks. By identifying the cyber risks that an organization faces appropriate controls and technology can be put in place.
Primary Objectives of the SOC

Detection
- Ability to detect malicious activity
- Incorporate solutions
- Baseline, baseline

Response
- Incorporate incident response
- RE of malware
- Can you recover. Honestly ask this
Where Did We Go Wrong in the SOC?
Where did we go wrong in the SOC

- We didn’t. The Times They Are A-Changin
- Lose expectations from red teamers
- Devops- not incorporated into the SOC
- Application security- typically functions outside the SOC
Where Did We Go Wrong in the SOC-DevOps

What is Devops

**DevOps** is a set of practices that combines software development (*Dev*) and information-technology operations (*Ops*) which aims to shorten the systems development life cycle and provide continuous delivery with high software quality.

-Wiki
Where Did We Go Wrong in the SOC-DevOps In Plain English

- A process of removing barriers between software development & IT Operations

- Accomplished with tools you (as a blue teamer) should learn about
Where Did We Go Wrong in the SOC - DevOps

DevOps Pipeline

1. CI/CD Framework
2. Source Control Management (SCM)
3. Build Automation Tool
4. Web Application Server
5. Code Testing Coverage

<table>
<thead>
<tr>
<th>Company</th>
<th>Deployment Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon</td>
<td>23,000 per day</td>
</tr>
<tr>
<td>Google</td>
<td>5,500 per day</td>
</tr>
<tr>
<td>Netflix</td>
<td>500 per day</td>
</tr>
<tr>
<td>Facebook</td>
<td>1 per day</td>
</tr>
<tr>
<td>Twitter</td>
<td>3 per week</td>
</tr>
<tr>
<td>Typical enterprise</td>
<td>1 every 9 months</td>
</tr>
</tbody>
</table>

How the heck do you implement security around this?
Where Did We go Wrong in the SOC

At a time in history in a land far far away a planet named “Pawnage” was the home to a hostile race of hackers.

This race was unlike other hackers. They sought to bring corporations to their knees.

They feed on capitalism and as more organizations offered web services, their appetite only grew.

Every packet was sniffable.

The security industry needed to make a stand and a new disciple was born.
Where Did We Go Wrong in the SOC-AppSec

Application Security
What is Application Security?

What sort of problems?
Attackers like to cause chaos and must be stopped before they crumble the world.

Why we care?
As defenders it’s our job to protect those that can’t protect themselves.

How can we tackle it?
Embrace the threat landscape and understand we must keep pace with attacker.
Where Did We Go Wrong in the SOC-AppSec

1/4 of breaches reported in 2019 were due to web application attacks

ElasticSearch Server Breach 100 Million Records

IAM Policy Misnomer

CapitalOne
Where Did We Go Wrong in the SOC-AppSec

Primary Responsibilities

• Static Code Testing
• Dynamic Code Testing
• Threat Modeling
• Penetration Testing
Where Did We Go Wrong in the SOC-AppSec

**DAST**
- Dynamic Application Security Tool
- Test for vulnerabilities on running application

**SAST**
- STATIC Application Security Tool
- Identify vulnerabilities in code
How Do We PSH Left

Stage 1 - Sprinkle of Identification & A Dash of Threat Modeling
The App!

• Yowl!

• A restaurant review site that really shouldn’t be in prod yet.

• Allows you to register a user, add reviews, and search existing reviews.

• We need to know what we’re concerned about before we can protect against it.

• We need to look at the big picture, and then the details.
Based on our threat model ... 

• We need to know what we’re concerned about before we can protect against it.

• This means looking at each application individually, but also cataloging and categorizing what we have. What do we actually care about?

• What would need to be fixed to actually solve the problem?

• How would we fix it?
Prioritize by concerns

- Public
- Internal
- Weird cloud services no one owns
- SaaS products
What Are Our concerns?

Alpha product

Behind authentication

Allows users to submit reviews

Has search functionality
How Do We PSH Left

Stage 2- Vulnerability Detection
How Do We PSH Left

• Let’s start with focusing on the vulnerabilities themselves

• This is familiar territory to us blue team folks

• Exploits>vulnerabilities>signature creation> alerting

• Exploits>vulnerabilities>huerstics>alerting
Is This Real life?

Sort of

Actual vulnerabilities, not simulated

Doesn’t use modern frameworks or client side processing
Our Application, YOWL!

- Super small app
- Super vulnerable
- It intends to be an image upload
- Does it stop you from uploading something else?
What’s Wrong With the App?

- Static Analysis
- Dynamic Analysis
- Code Review
- Manual testing
Quick Aside

- Product security, and application security in general, are focused on prevention.
- They are trying to find and fix any problems before they are found and exploited by an attacker.
First, Static Findings

<table>
<thead>
<tr>
<th>Level</th>
<th>Issue Type</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIGH</td>
<td>Static Scan</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SQL Injection</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>2</td>
</tr>
<tr>
<td>MEDIUM</td>
<td>Static Scan</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Credentials Management</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Cross-Site Scripting</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>Cryptographic Issues</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Directory Traversal</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>13</td>
</tr>
<tr>
<td>LOW</td>
<td>Static Scan</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Information Leakage</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>1</td>
</tr>
</tbody>
</table>
Dynamic Analysis

Not everything found by static is easily exploitable, and sometimes what’s wrong with your app isn’t visible in the code.
Now that We’ve Looked Through the Reports

What do you think is the biggest problem with this application?

What would you prioritize to fix or monitor?
Based on our Threat Model ...

What do we actually care about?

What would need to be fixed to actually solve the problem?

How would we fix it?

How long will the vulnerability live?
How Do We Push Left

Blue Team Brothers and Sisters…. It’s Our Time To Shine
How Do We PSH Left

Stage 3- Purple Teaming aka Useful Pen-Testing For The SOC?
Scenario

#SOC OPSEC

- Stage 1 - Local file inclusion vulnerability running on Apache client for Yowl. Upload payload
- Stage 2 - Navigate to payload directory to leverage system calls for further exploitation. In our case, creating a backdoor. YAY
- Stage 3 - Upload backdoor. Circumvent defensive tools e.g. ossec, create persistence
- Stage 4 - Find database credentials and dump them
- Stage 4 - Exfiltration of data
How Do We PSH Left

• Don’t let your pen-testers off the hook
• Create dialog to understand their capabilities

Objectives
• Fool me once
• Plan and coordinate engagements together
• Where is the SOC being defeated
Can We Get In
• Web server accepted the payload
• Time to grab hashes and dump’em
• Reverse the hash & write passwords to file
• Move to exfil, then leave
• Simple.
#SOC OPSEC-Migration

- Attacker methods to blend in
  - Odd cron jobs
  - Process hi-jacking
  - DLL Sideload
  - Out of place powershell scripts

- Hint: Don’t trust the timestamps
How Do We PSH Left

• Focus your efforts on replicating identified vulnerabilities

• Tools like OWASP ZAP or Burpsuite can be of use

• Now its time to re-visit the Identification pillar

• We hunt for artifacts related to the testing

• Remember what I said about **logging practices**
How Do We PSH Left

• Focus efforts on creating detection capabilities

• With identification squared away this is a step up

• Can you do this alone or do you need the support from your vendor?

• Continuous testing of your security posture is key

• Demo or graphical explanation of detection
How Do We PSH Left

Of Course We Have to Mention MITRE ATT&CK
How Do We PSH Left

Where Are Remediation Efforts Focused?
How Do We PSH Left

Stage 4- DevOPS Visibility
How Do We PSH Left

---

Do you understand the builds?

Jenkins — Ansible — Packer — AWS-CLI — AWS-EC2

Ansible — Vagrant — Packer — Terraform — Azure

Base Image — Package — Code — Container
How Do We PSH Left

Access control review for developers in GitHub

• Who can commit
• How secure is repo access
• GitHub Audit
How Do We PSH Left
How Do We PSH Left
How Do We PSH Left

The Golden Rule

Golden Image
• A Windows or Linux operating system installation
• New security patches and updates
• Configuration specific to your environment
• Software specific to your environment
• Security hardening configs

Golden Pipeline
• Store your golden image as code in a centralized git repository
• Allow pull requests to make changes to the golden images
• Automatically test and validate that the changes will work with the applications relying on them
• Automatically deploy the created images, so they can be consumed by applications
How Do We PSH Left

Access control review for developers in GitHub
- Who can commit
- How secure is repo access
- GitHub Audit

Microcontainers
- Ensure you scan dependencies
- Vulnerabilities
- Correct patch level
- Defined VPC and IAM Polices
- SOC working with DevOps to develop vulnerability sliding scale.
How Do We PSH Left

IF
• Microcontainters
• Golden Pipeline
• Golden Image

THEN
• Out of compliance
• Introduces serve vulnerability
• Patch level issues

BREAK THE BUILD
Okay, I encoded the AWS Creds

Smooth work brah
Conclusion

• Lets try to jump out ahead of the change

• It's your mission to understand appsec practices within your organization

• You need to get involved with devops

• Most of the tools covered today are open source, so go test them out.

• Questions
Access control review for developers in GitHub

• Who can commit
• How secure is repo access
• GitHub Audit
How Do We PSH Left
How Do We PSH Left