Agile + DevOps EAST

A TECHWELL EVENT

DT11

DevSecOps

Thursday, November 8th, 2018 3:00 PM

Serverless Security: Overcome Architectural Security Challenges

Presented by:

Eric Sheridan

WhiteHat Security

‘

Brought to you by:

TECHWELL™

Eric Sheridan

As chief scientist at WhiteHat Security, Eric oversees research and development for Sentinel Source and related products. Eric leads the WhiteHat Certified Secure Developer (WCSD) program, a free training program designed to educate and certify developers on secure coding and application security best practices. Prior to joining WhiteHat, Eric cofounded Infrared Security, specializing in application security and next-generation static analysis technologies that were ultimately integrated within WhiteHat Sentinel Source. He earned a bachelor of science degree in computer science with a track in security from Towson University.

Serverless Security! Wait, What?

Eric Sheridan
Chief Scientist
eric.sheridan@whitehatsec.com

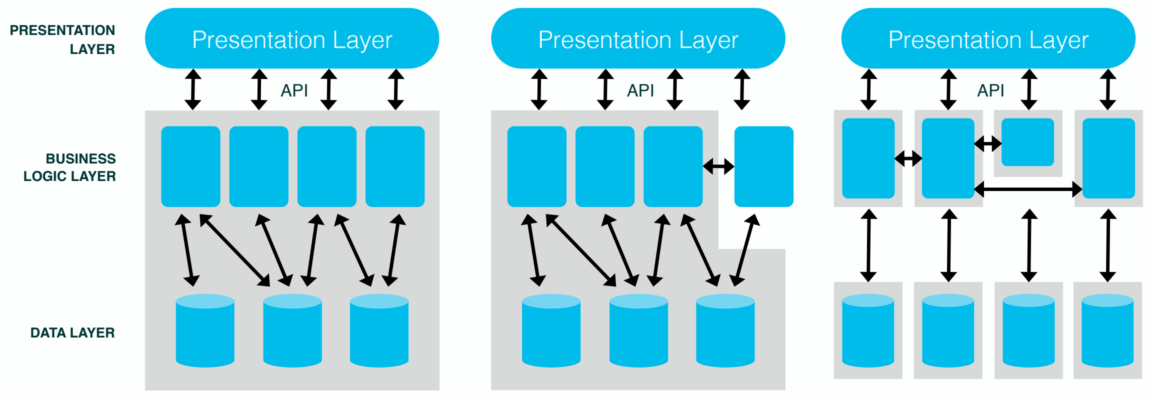
1

Agenda

- Overview of Serverless Architecture
- Key Security Considerations
- “Sec” in “DevSecOps”
- Q and A

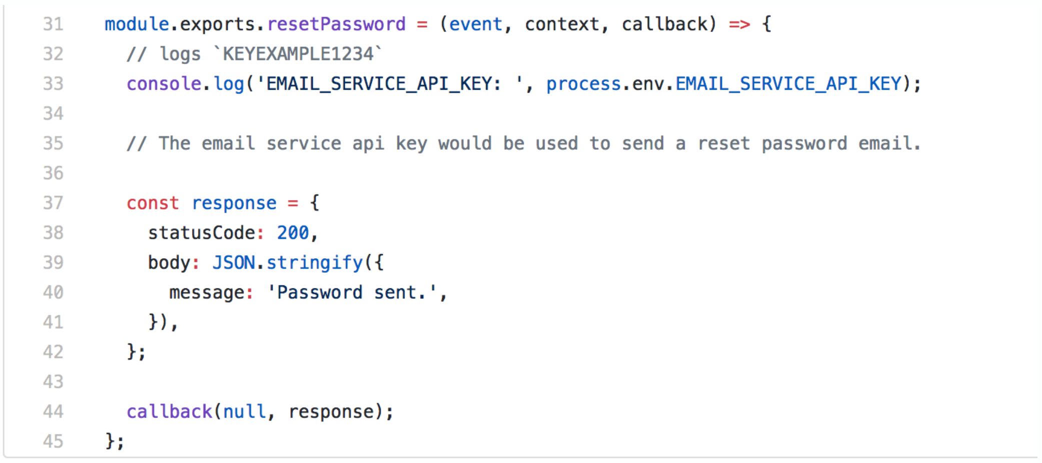
2

Transition to Serverless



WhiteHat SECURITY

Serverless Framework - Example



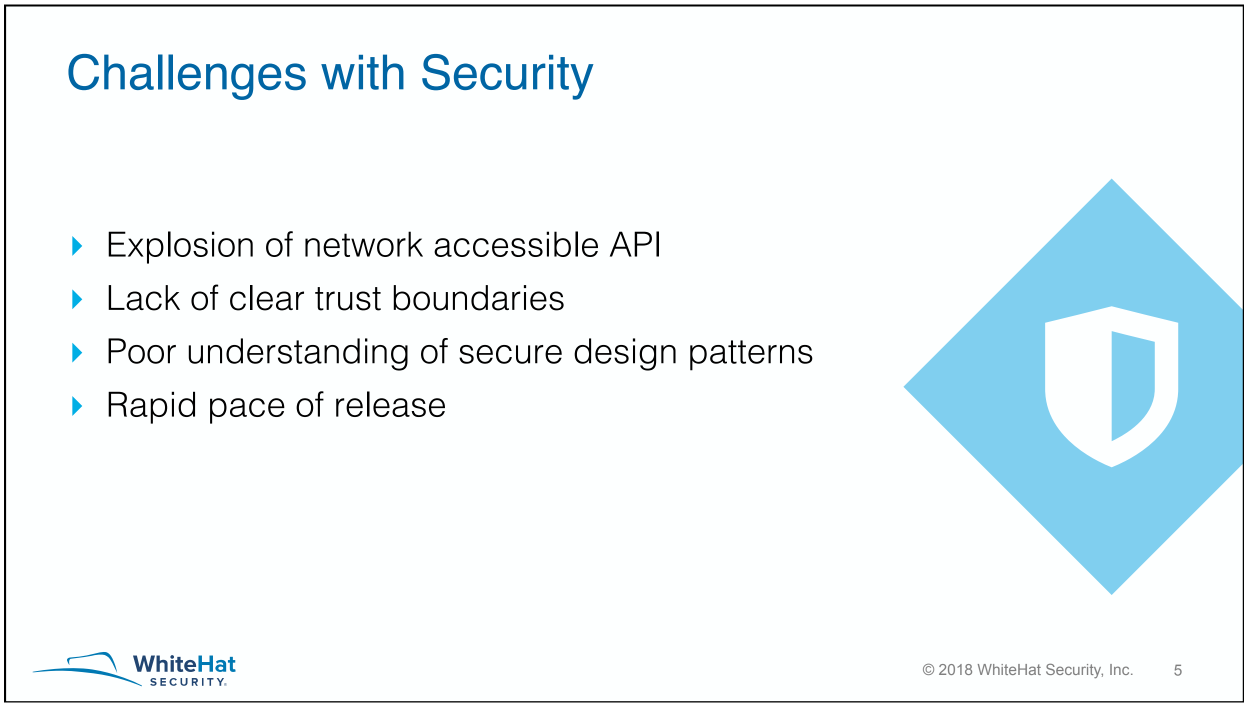
```
module.exports.resetPassword = (event, context, callback) => {
  // logs `KEYEXAMPLE1234`
  console.log('EMAIL_SERVICE_API_KEY: ', process.env.EMAIL_SERVICE_API_KEY);

  // The email service api key would be used to send a reset password email.

  const response = {
    statusCode: 200,
    body: JSON.stringify({
      message: 'Password sent.',
    }),
  };

  callback(null, response);
};
```

WhiteHat SECURITY

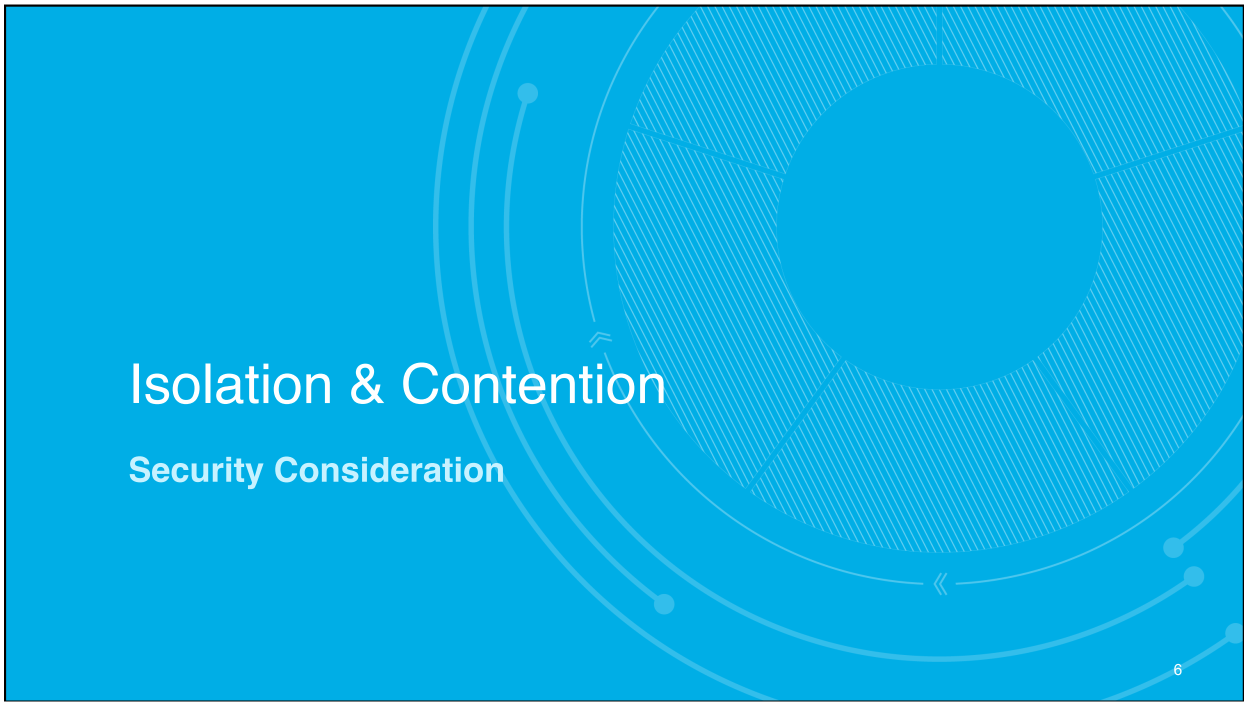


Challenges with Security

- Explosion of network accessible API
- Lack of clear trust boundaries
- Poor understanding of secure design patterns
- Rapid pace of release

WhiteHat SECURITY.

5



Isolation & Contention

Security Consideration

6

Serverless Isolation

Understanding isolation in across vendors…

- **AWS provides you with dedicated VM instances**
  - No clear way to run your functions on a VM dedicated to another User
- **Azure may share your VM instance with other users**
  - Results in shared resources with other users!
- **Google's strategy for isolation is unclear**
  - No clear way to uniquely identify underlying VM / host

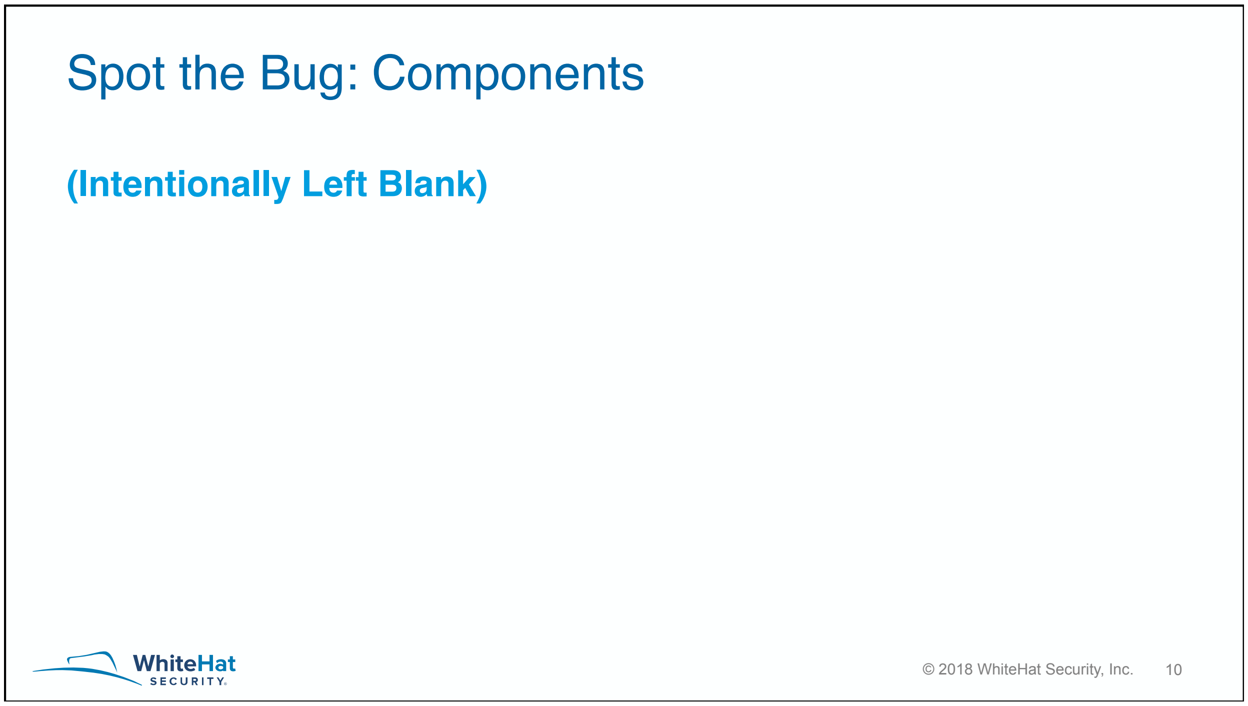
Spot the Bug: ReadLine

(Intentionally Left Blank)

3rd Party Components

Security Consideration

9

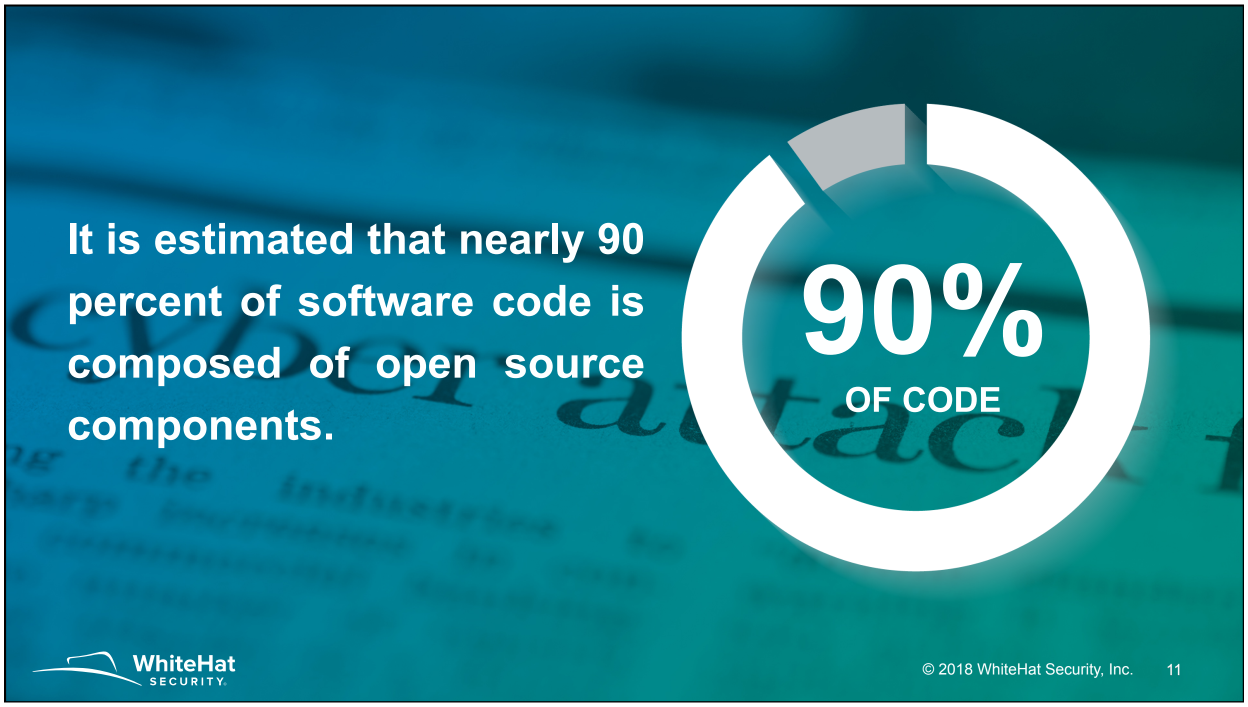


Spot the Bug: Components

(Intentionally Left Blank)

WhiteHat SECURITY

© 2018 WhiteHat Security, Inc. 10



It is estimated that nearly 90 percent of software code is composed of open source components.

90%
OF CODE

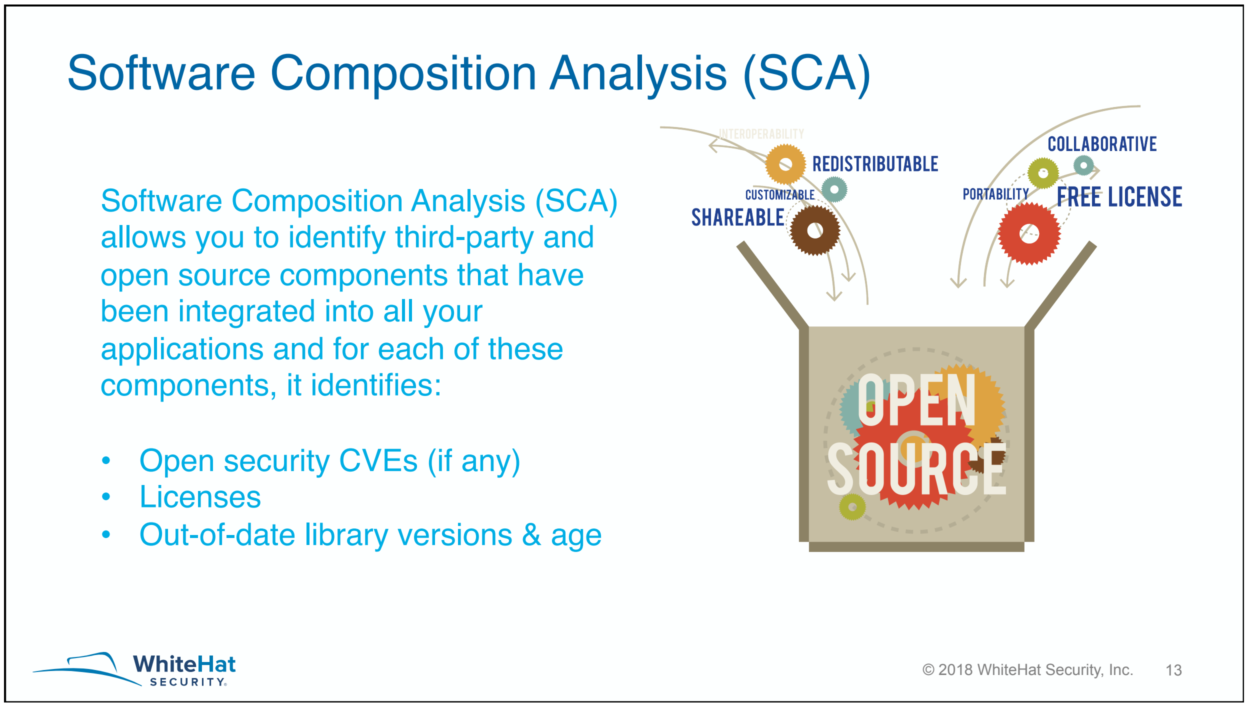
© 2018 WhiteHat Security, Inc. 11

The Equifax Hack: Struts Vulnerability

Apache Struts 2 is an open-source web application framework for developing Java EE web applications.

Highlights huge need for Software Composition Analysis

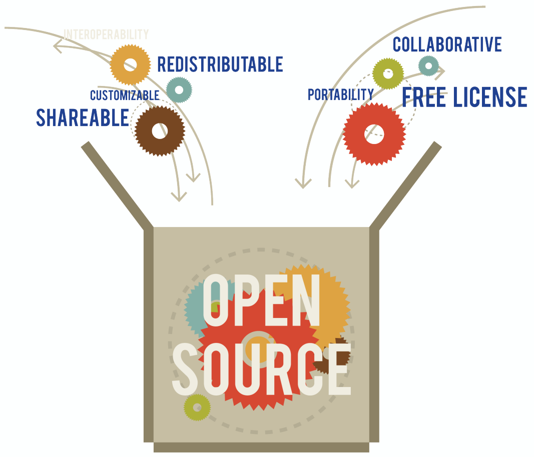
© 2017 WhiteHat Security, Inc. 12



Software Composition Analysis (SCA)

Software Composition Analysis (SCA) allows you to identify third-party and open source components that have been integrated into all your applications and for each of these components, it identifies:

- Open security CVEs (if any)
- Licenses
- Out-of-date library versions & age



WhiteHat SECURITY.

© 2018 WhiteHat Security, Inc. 13

Sensitive Data

Security Consideration

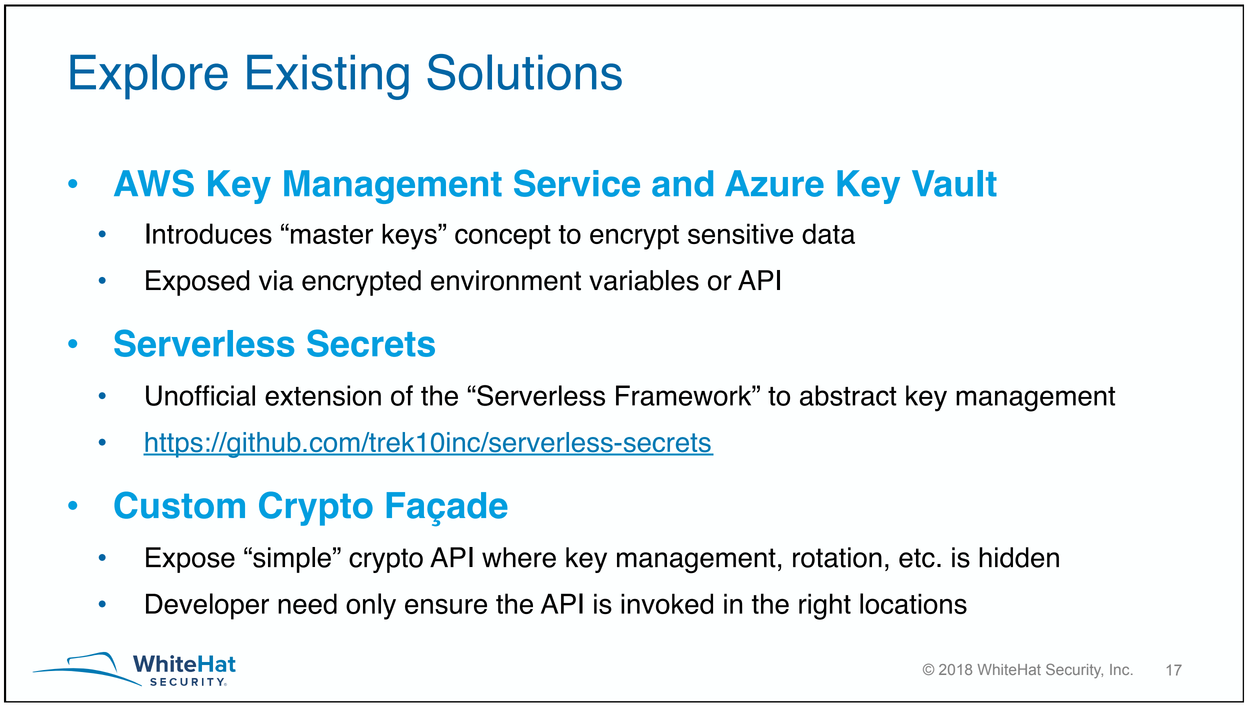
14

Spot the Bug: Tokens

(Intentionally Left Blank)

Spot the Bug: Cryptography

(Intentionally Left Blank)

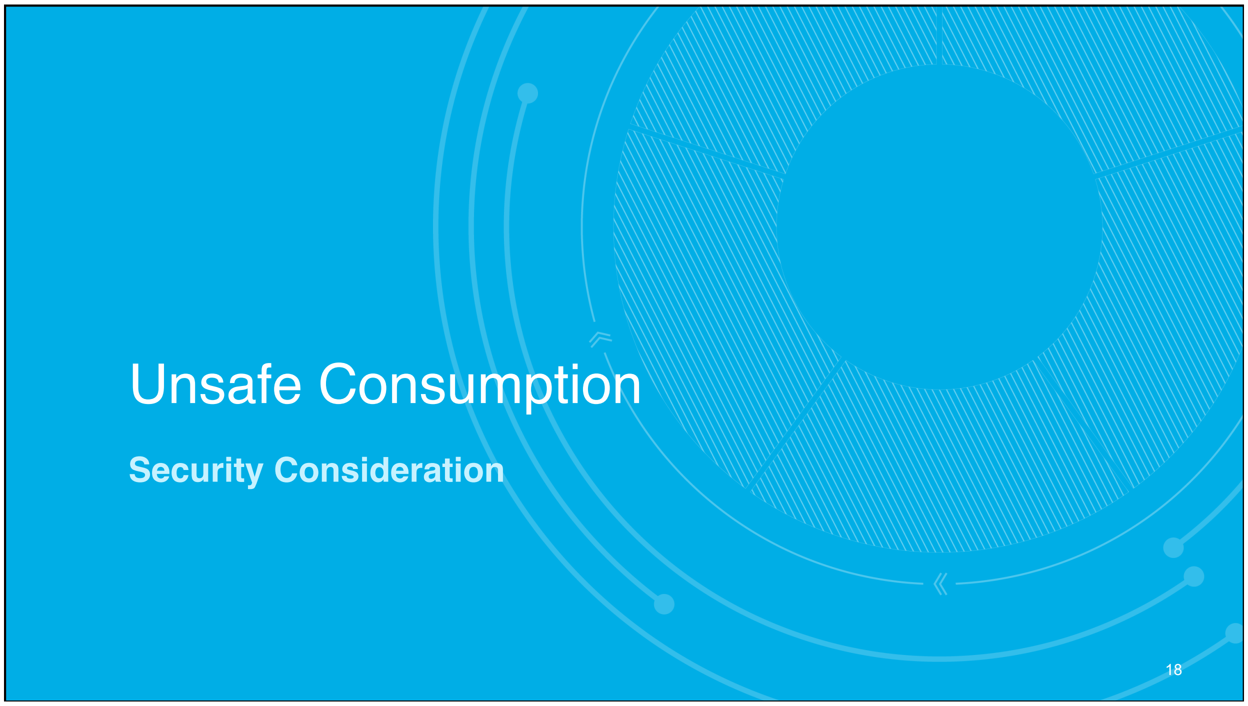


Explore Existing Solutions

- **AWS Key Management Service and Azure Key Vault**
  - Introduces “master keys” concept to encrypt sensitive data
  - Exposed via encrypted environment variables or API
- **Serverless Secrets**
  - Unofficial extension of the “Serverless Framework” to abstract key management
  - https://github.com/trek10inc/serverless-secrets
- **Custom Crypto Façade**
  - Expose “simple” crypto API where key management, rotation, etc. is hidden
  - Developer need only ensure the API is invoked in the right locations

WhiteHat SECURITY

17



Unsafe Consumption

Security Consideration

18

Spot the Bug: Interpreter

(Intentionally Left Blank)

Variable Binding

Explicitly differentiate for the SQL parser commands vs data

- "Pre-compiles" the SQL query before introducing untrusted data

Rewrite dynamic SQL queries into literals with placeholders

- **(before) "SELECT * FROM USERS WHERE name = '" + name + "'";**
- **(after) "SELECT * FROM USERS WHERE name = ?";**

Available in all modern development languages

- Java, .NET, NodeJS, etc.

Examples in NodeJS Using Sequelize

BAD

```
let name = req.params.name;
let sql = "SELECT * FROM USERS WHERE NAME ='" + name + "'";
let result = sequelize.query(sql, { type: sequelize.QueryTypes.SELECT });
```

GOOD

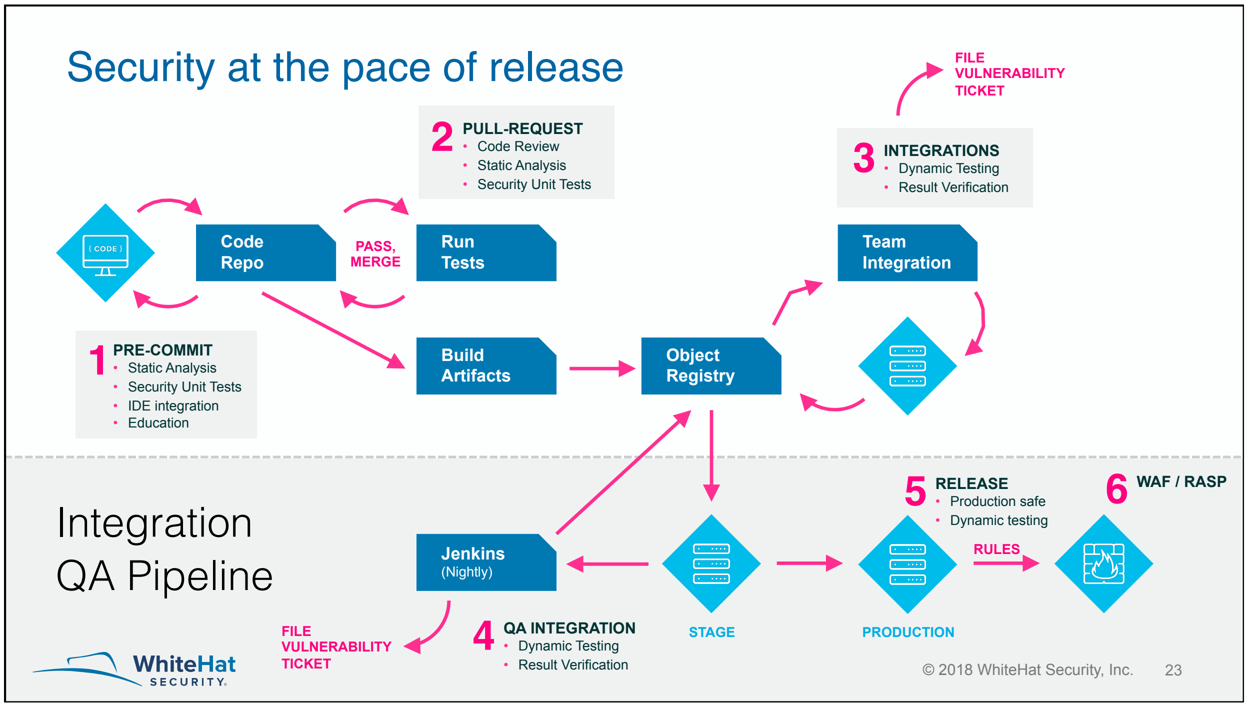
```
let name = req.params.name;
let sql = "SELECT * FROM USERS WHERE NAME = ?";
let result = sequelize.query(sql, {
    type: sequelize.QueryTypes.SELECT, replacements: [ name ]
});
```

GOOD

```
let User = require('models').User;
let result = User.findOne({ where: { name: req.params.name } });
```

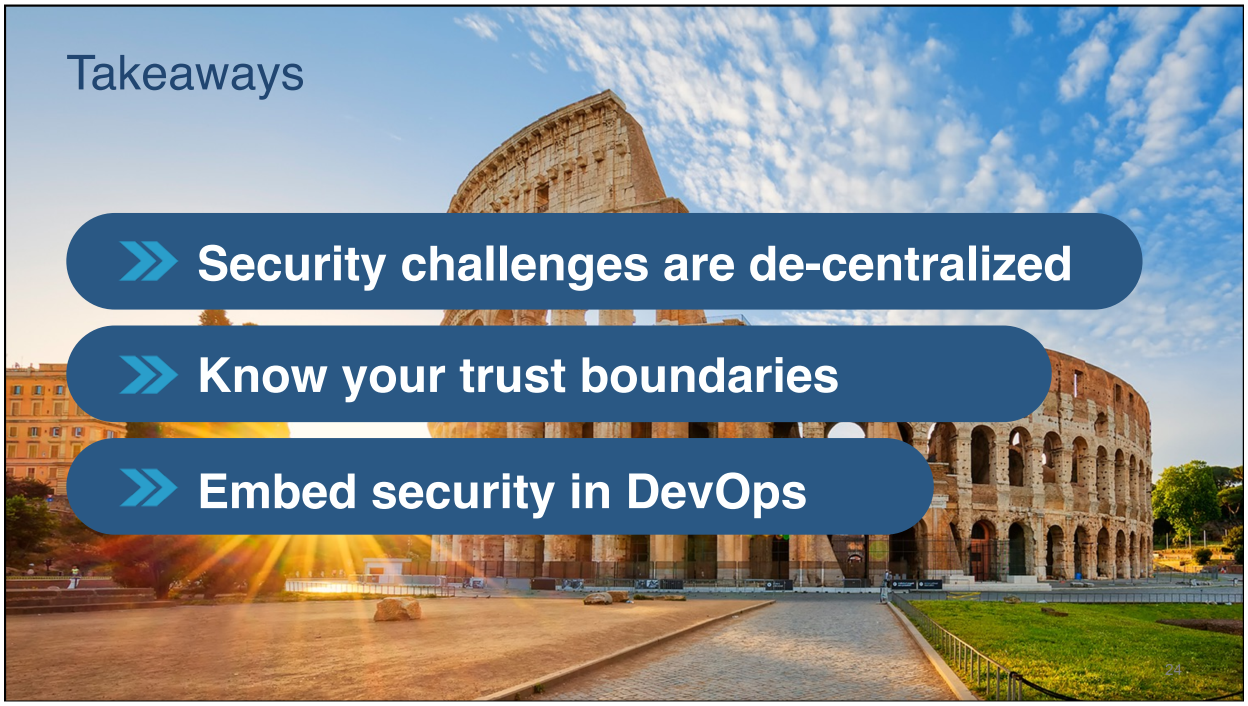
Spot the Bug: Service

(Intentionally Left Blank)



Security at the pace of release

© 2018 WhiteHat Security, Inc. 23



Takeaways

- Security challenges are de-centralized
- Know your trust boundaries
- Embed security in DevOps

24

Thank you

Eric Sheridan
Chief Scientist
eric.sheridan@whitehatsec.com

25