DevSecOps

Module 1: DevSecOps Drivers and Challenges

compliance->iso 27001,27017(service provider,cloud),27018(additionally)

Cloud security alliance(CSA),Cloud Controls Matrix (CCM)

Consensus Assessments Initiative Questionnaire(CAIQ),Google Vendor Security Assessment Questionnaires (VSAQ)

Federal Information Processing Standards (FIPS)->crypto->OWASP Cryptographic Storage Cheat Sheet. OWASP Guide to Cryptography OWASP Key Management Cheat Sheet

asymmetric->AES(>128 bit)

symmetric->RSA(>1024 bit)

hash->SHA256

Digital signature->RSA (>=2048 bits) DSA (>=2048 bits) ECDSA (>=256 bits)

Hellman key exchange (DH)->DH (>=2048 bits) ECDH(>-256 bits)

GDRP:

Design stage->Design Privacy Impact Assessment (PIA)

Coding stage->Data masking library, Anonymous toolbox, RAPPOR—privacy-preserving reporting algorithms, Encryption storage (RSA, ASE), Secure erasure, Secure communication protocol (such as TLS v1.2, SSH v2, SFTP, SNMP v3), Cookie consent, Data Vault, Key management

Testing->OWASP testing for weak cryptography, testing for error handling, testing for configuration, and so on

Deployment->OWASP configuration and deployment management testing, CIS secure environment configuration, Sensitive information in Git

Monitoring->ELK for log analysis, Integrity monitoring (IDS/IPS) to monitor any unauthorized changes, CIS secure configuration monitoring, Sensitive information leakage in Git

Virtualization

1. Limit informative messages from the VM to the VMX file
2. Limit sharing console connections
3. Disconnect unauthorized devices (USB, DVD, serial devices, and so on)

4. Disable BIOS Boot Specification (BBS)
5. Disable guest-host interaction protocol handler
6. Disable host guest filesystem server
7. Disable VM console paste operations
8. Disable virtual disk shrinking
9. Do not send host information to guests

Docker

1. Separate partition for containers
2. Updated Linux kernel
3. Only allow trusted users to control the Docker daemon
4. Audit the Docker daemon, files, and directories
5. Restrict network traffic between containers
6. TLS authentication for the Docker daemon
7. Do not bind Docker to another IP/port or a Unix socket
8. Docker daemon configuration files permissions
9. Container runtime (Linux Kernel capabilities, SSH, ports, memory, CPU, IPC)

Infrastructure as Code

Puppet, Chef, Ansible, and SaltStack

This will help both operation or development teams to build security configuration baselines such as file permissions, firewall rules, web configurations, or MySQL connections. Once the security configuration baseline is defined, the operation team can monitor any unauthorized changes or roll back the configuration to previous specific versions

Cloud services hacks/abuse

- Data breaches
- Weak identity, credentials, and access management
- Insecure APIs
- System and application vulnerabilities
- Account hijacking
- Malicious insiders
- Advanced Persistent Threats (APTs)
- Data loss
- Insufficient due diligence
- Abuse and nefarious use of cloud services
- Denial of service
- Shared technology issues

Rapid release

1-Continuous integration

Jenkins, Git, Unit testing

2-Continuous delivery

IaC(Puppet), Docker

3-Continuous deployment

IaC (puppet), Docker Automated acceptance testing, Configuration

Module 2: Security Goals and Metrics

- Organization goal
- Development goal/metrics
- QA goals/metrics
- Operation goal/metrics

Application Security Verification Standard(ASVS):

Authentication, Session Management, Access Control, Malicious Input handling, Output encoding/escaping, Cryptography, Error handling and logging , Data Protection, Communication Security, Http Security configuration, Security configuration, Malicious, Internal Security, Business logic, Files and resources, Mobile, Web services

Design review

- Security compliance checklist
- Security requirement checklist (OWASP ASVS)
- Top 10 security design issues
- Security issues in the previous release
- Customer or marketing feedback on security issues

Implementation review

- Secure coding
- Selection of reliable and secure third-party components
- Secure configuration

Third-party components

1. A third-party software evaluation checklist:
2. Recommended third-party software and usage by projects:
3. CVE status of third-party components:

Code Review

Static Application Security Testing (SAST)->FindSecbugs, Fortify, Coverity, klocwork.

Dynamic Application Security Testing (DAST)->OWASP ZAP, BurpSuite

Interactive Application Security Testing (IAST)->CheckMarks Varacode

Run-time Application Security Protection(RASP)->OpenRASP

https://www.owasp.org/index.php/Category:OWASP _Code_Review_Project SEI CERT Coding
https://wiki.sei.cmu.edu/confluence/display/seccode/SEI+CERT+Coding+Standards

Software Assurance Marketplace (SWAMP): https://www.mir-swamp.org/

Environment Hardening

- Secure configuration baseline
- Constant monitoring mechanism

Constant monitoring mechanism

1-Common vulnerabilities and exposures (CVEs)

OpenVAS, NMAP

2-Integrity monitoring

OSSEC

3-Secure configuration compliance

OpenSCAP

4-Sensitive information exposure

No specific open source tool in this area. However, we may define specific regular expression patterns

Module 3: Security Assurance Program and Organization

SDL (Security Development Lifecycle)

Training:

Core security training

Requirements:

1. Establish security requirements
2. Create quality gates/bug bars
3. Perform security and privacy risk assessments

Design:

1. Establish design requirements
2. Perform attack surface analysis reduction
3. Use threat modeling

Implementation:

1. Use approved tools
2. Deprecate unsafe functions
3. Perform static analysis

Verification:

1. Perform dynamic analysis
2. Perform fuzz testing
3. Conduct attack surface review

Release:

1. Create an incident response plan
2. Conduct final security review
3. Certify, release, and archive

Response:

1. Execute incident response plan

OWASP SAMM

OWASP SAMM categorizes security practices into four key business

Governance:

1. Strategy and metrics
2. Policy and compliance
3. Education and guidance

Construction:

1. Threat assessment
2. Security requirements
3. Secure architecture

Verification:

1. Design review
2. Implementation review
3. Security testing

Operations:

1. Issue management
2. Environment Hardening
3. Operational enablement

Security guidelines and processes

1- Security training:

Security awareness, Security certification program, Case study knowledge base, Top common issue, Penetration learning environment

OWASP top 10, CWE top 25, OWASP VWAD

2- Security maturity assessment

Microsoft SDL, OWASP SAMM self-assessment for maturity level

Microsoft SDL, OWASP SAMM

3- Secure design

Threat modeling templates (risks/mitigation knowledge base), Security requirements for release gate, Security design case study, Privacy protection

OWASP ASVS, NIST, Privacy risk assessment

4- Secure coding

Coding guidelines (C++, Java, Python, PHP, Shell, Mobile), Secure coding scanning tools, Common secure coding case study

CWE, Secure coding, CERT OWASP

5- Security testing

Secure compiling options such as Stack Canary, NX, Fortify Source, PIE, and RELRO, Security testing plans, Security testing cases, Known CVE testing, Known secure coding issues, API-level security testing tools, Automation testing tools, Fuzz testing, Mobile testing, Exploitation and penetration, Security compliance

Kali Linux tools, CIS

6- Secure deployment

Configuration checklist, Hardening guide, Communication ports/protocols, Code signing

CIS Benchmarks, CVE

7- Incident and vulnerability handling

Root cause analysis templates, Incident handling process and organization

NIST SP800-61

8- Security training

Security awareness by email, Case study newsletter, Toolkit usage hands-on training, Security certificate and exam

NIST 800- 50, NIST 800- 16, SAFECode security engineering training

Stage 1 – basic security control

- Leverage third-party cloud service provider security mechanisms (for example, AWS provides IAM, KMS, security groups, WAF, Inspector, CloudWatch, and Config)
- Secure configuration replies on external tools such as AWS Config and Inspector
- Service or operation monitoring may apply to AWS Config, Inspector, CloudWatch, WAF, and AWS shield

Stage 1 – building a security testing team

Vulnerability assessment:

NMAP, OpenVAS

Static security analysis:

FindBugs for Java, Brakeman for Ruby on Rails, Infer for Java, C++, Objective C and C

Web security:

OWASP dependency check, OWASP ZAP, Archni-Scanner, Burp Suite, SQLMap, w3af

Communication:

Nmap, NCAT, Wireshark, SSLScan, sslyze

Infrastructure security:

OpenSCAP, InSpec

VM Toolset:

Pentest Box for Windows, Kali Linux, Mobile Security Testing Framework

Security monitoring:

ELK, MISP—Open source Threat Intelligence Platform, OSSCE—Open source HIDS Security, Facebook/osquery—performant endpoint visibility, AlienValut OSSIM—opensource SIEM

Stage 3 – SDL activities

- Security shifts to the left and involves every stakeholder
- Architect and design review is required to do threat modeling
- Developers get secure design and secure coding training
- Operation and development teams are as a closed-loop collaboration
- Adoption of industry best practices such as OWASP SAMM and Microsoft SDL for security maturity assessment

Stage 4 – self-build security services

Take Salesforce as an example—the Salesforce Developer Center portal provides security training modules, coding, implementation guidelines, tools such as assessment tools, code scanning, testing or CAPTCHA modules, and also a developer forum. Whether you are building

an application on top of salesforce or not, the Salesforce Developer Center is still a good reference not only for security knowledge but also for some open source tools you may consider applying.

Stage 5 – big data security analysis and automation

Key characteristics at this stage are:

- Fully or mostly automated security testing through the whole development cycle
- Applying big data analysis and machine learning to identify abnormal behavior or unknown threats
- Proactive security action is taken automatically for security events, for example, the deployment of WAF rules or the deployment of a virtual patch

Typical open source technical components in big data analysis frameworks include the following:

- Flume, Log Logstash, and Rsyslog for log collection
- Kafka, Storm, or Spark for log analysis
- Redis, MySQL, HBase, and HDFS for data storage
- Kibana, ElasticSearch, and Graylog for data indexing, searching, and presentation

The key stages in big data security analysis are explained in the table:

Data collection:

Collects logs from various kinds of sources and systems such as firewalls, web services, Linux, networking gateways, endpoints, and so on.

Data normalization:

Sanitizes or transforms data formats into JSON, especially, for critical information such as IP, hostname, email, port, and MAC.

Data enrich/label:

In terms of IP address data, it will further be associated with GeoIP and WhoIS information. Furthermore, it may also be labeled if it's a known black IP address.

Correlation:

The correlation analyzes the relationship between some key characteristics such as IP, hostname, DNS domain, file hash, email address, and threat knowledge bases.

Storage:

There are different kinds of data that will be stored —the raw data from the source, the data with enriched information, the results of correlation, GeoIP mapping, and the threat knowledge base.

Alerts:

Trigger alerts if threats were identified or based on specified alerting rules.

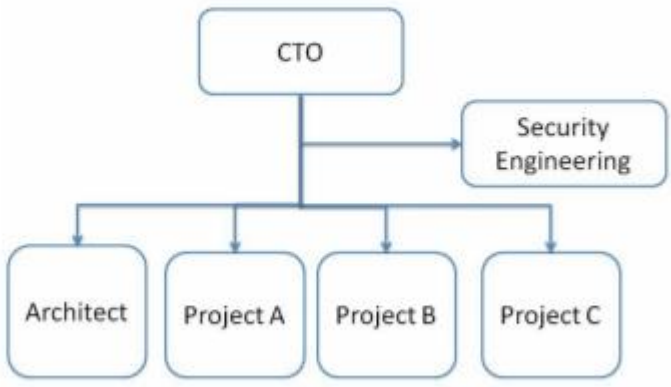
Presentation/query:

Security dashboards for motoring and queries. ElasticSearch, RESTful API, or third-party SIEM.

Role of a security team in an organization

1- Security office under a CTO

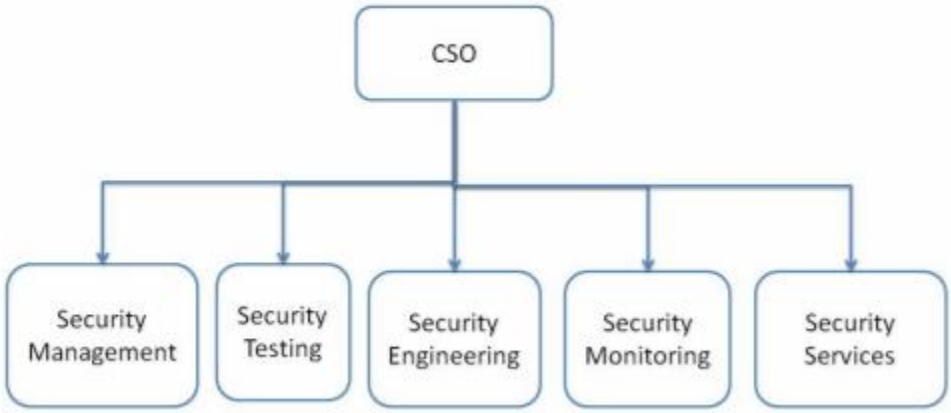
- No dedicated Chief Security Officer (CSO)
- The security team may not be big—for example, under 10 members
- The security engineering team serves all projects based on their needs
- The key responsibility of the security engineering team is to provide security guidelines, policies, checklists, templates, or training for all project teams
- It's possible the security engineering team members may be allocated to a different project to be subject matter experts based on the project's needs
- Security engineering provides the guidelines, toolkits, and training, but it's the project team that takes on the main responsibility for daily security activity execution



2-Dedicated security team

- **Security management:** The team defines the security guidelines, process, policies, templates, checklist, and requirements. The role of the security management team is the same as the one previously discussed in the Security office under a CTO section.

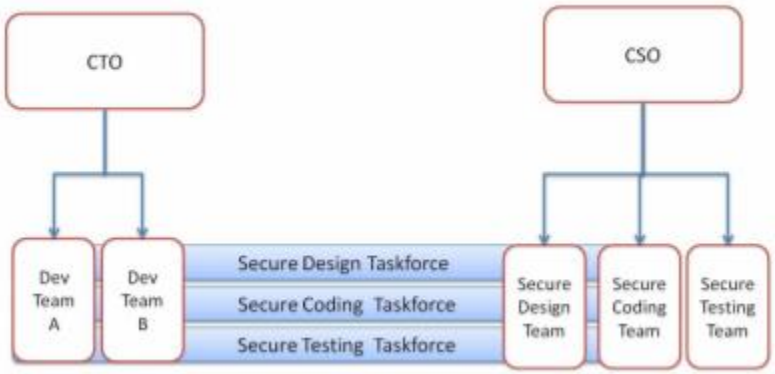
- **Security testing:** The team is performing in-house security testing before application release.
- **Security engineering:** The team provides a common security framework, architecture, SDK, and API for a development team to use
- **Security monitoring:** This is the security operation team, who monitor the security status for all online services.
- **Security services:** This is the team that develops security services such as WAF and intrusion deference services.



Security technical committee (taskforce)

The secure design taskforce will have a weekly meeting with all security representatives—from all project teams— and security experts from the security team to discuss the following topics (not an exhaustive list):

- Common secure design issues and mitigation (initiated by security team)
- Secure design patterns for a project to follow (initiated by security team)
- Secure design framework suggestions for projects (initiated by security team)
- Specific secure design issues raised by one project and looking for advice on other projects (initiated by project team)
- Secure design review assessment for one project (initiated by project team)



Module 4: Security Requirements and Compliance

Release gate examples

1-Design

- Threat modeling activities were performed for highrisk modules.
- The uses of third-party component versions was reviewed without major vulnerability.
- The top common secure design issues were reviewed without major issues.

2-Coding

- The static code analysis tool was used to identify major security risks.
- High severity issues in the code scanning results were all checked.
- No sensitive information was found in the source code (such as password, IP, email, encryption key).

3-Build

Toolchain (compiler and linker) hardening configurations such as **Position Independent Executables (PIE)**, or **Address Space Layout Randomization (ASLR)**, or **Data Execution Prevention (DEP)** were correctly configured.

4-Testing

- No high-severity security issue. The severity is measured by the Common Vulnerability Scoring System (CVSS) version 3.0
- OWASP testing cases were followed and executed.

- All protocols were tested with a fuzzer.

5-Production delivery

- The secure configuration definition was delivered.
- The communication ports, interface, and protocols were documented.

6-Monitoring

- The readiness of services and the configuration list for security scanning
- The readiness of service logs for security analysis.

Common Vulnerability Scoring System (CVSS)

Therefore, to get a more objective standpoint on the severity and impact of a security issue, it's suggested to apply CVSS 3.0. CVSS 3.0, https://www.first.or g/cvss/calculator/3.0, evaluates a security issue by answering the following eight questions:

- Attack Vector (AV): Does the attack require physical access, or can it be done through a network?
- Attack Complexity (AC): Can the attack be done at any time, or at only under specific conditions?
- Privileges Required (PP): Does the attack require administrator privileges?
- User Interaction (UI): Does the attack require user interaction (such as a click) to be successful?
- Scope (S): Does the attack only impact the vulnerable component, or all other components and the whole system?
- Confidentiality (C): Will any confidential information be stolen?
- Integrity (I): Will there be any integrity impact, such as tampering or changes to system information?
- Availability (A): Will there be any availability impact, such as a performance impact or services unavailable?

Security requirements for web applications

https://www.owasp.org/index.php/Category:OWASP_Application_Security _Verification_Standard_Project.

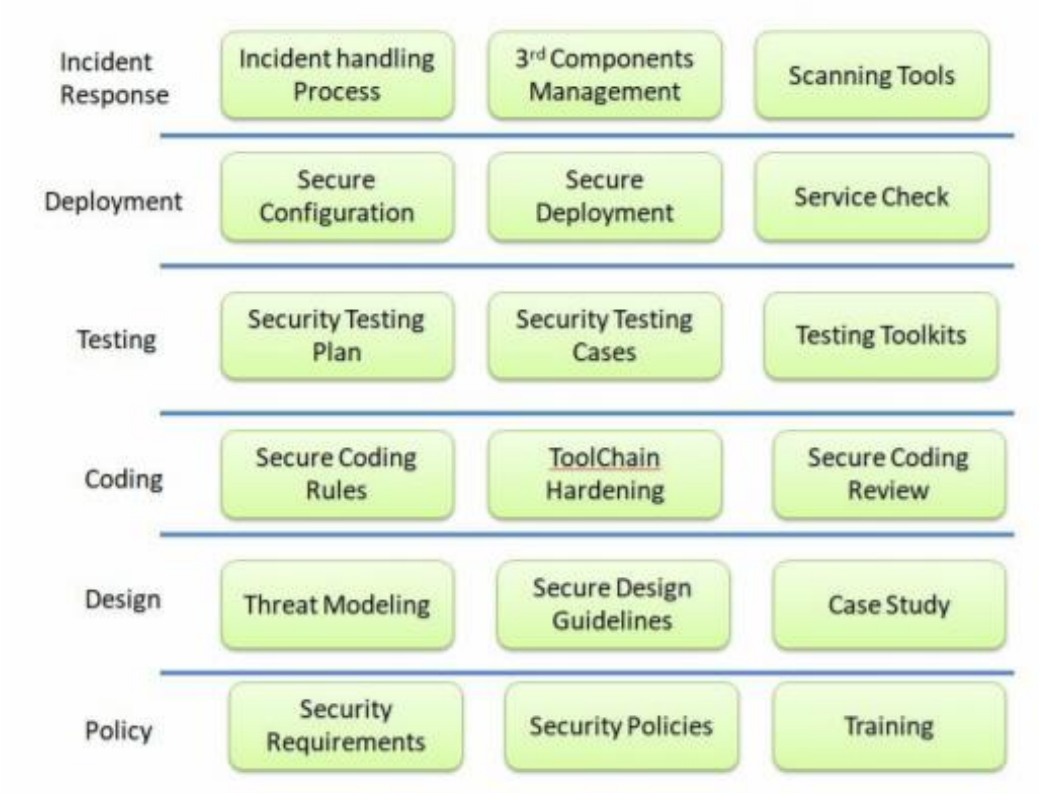
- ASVS V1 Architecture
- ASVS V2 Authentication
- ASVS V3 Session Management
- ASVS V4 Access Control
- ASVS V5 Input Validation and Output Encoding
- ASVS V7 Cryptography
- ASVS V8 Error Handling
- ASVS V9 Data Protection
- ASVS V10 Communications
- ASVS V13 Malicious Code
- ASVS V15 Business Logic Flaws
- ASVS V16 Files and Resources
- ASVS V17 Mobile

- ASVS V18 API
- ASVS V19 Configuration
- ASVS V20 Internet of Things

The OWASP ASVS defines three levels of security requirements. Take V7: Cryptography at rest as examples; in level-1 applications, it may only require that cryptographic modules fail securely. For level 2/3 applications, whose security requirements surpass level 1, additionally requires the use of an approved random number generator in the application

Security knowledge portal

The project team can also share their best practices or tools on the portal, to increase experience sharing across business units. An ideal security knowledge portal may cover the following areas, as shown in the following figure:



https://github.com/blabla1337/skf-flask

Big data security requirements

1-Infrastructure Security

- Database and service availability
- Protection against DDOS and a huge volume of data
- Secure data transmission, such as TLS 1.2

2-Data Privacy

- Data classification and protection
- Unauthorized access auditing and logging
- Data masking for sensitive or personal information
- Compliance with privacy laws or regulations

3-Data Management

- Secure database storage, such as secure configurations, encryption, and hardening
- Data governance during data life cycle processes
- Tell the users how the data is collected and used
- Explicit user consent for any collection of personal data
- Allow a user to edit, update, or delete the collected data

4-Integrity and Reactive Security

- Security analysis of logs to identify abnormal data access
- Prevent data from being tampered with
- Inform users when a security incident occurs

Big data technical security frameworks

1-Centralized security administration and management, Authorization and permissions control, Centralized audits and reports

- Apache Ranger
- Apache Sentry

2-Operation monitoring and audits

- Apache Ambari

3-Enforcement of REST API security, Perimeter security

- Apache Knox

4-Secure transmission

- TLS v1.2 instead of HTTP
- SSH v2 instead of Telnet
- SFTP instead of FTP

5-Authentication

- Kerberos

6-Secure configuration and deployment

- Kerberos, and Knox secure configuration such as file permissions, daemon users, NTP, certificates, and TLS

7-Data governance, Data life-cycle management, Data classification such as PII, classified, authorization/datamasking based on classifications

- Apache Atlas

The following are some further recommended references for big data privacy and security:

- SP.1500-4 big data interoperability framework: Volume 4, Security and Privacy
- ENISA: Privacy by design in big data
- CSA Expanded top ten big data security and privacy challenges
- Information Commissioner's office Guide to data protection
- ENISA: Big data security

Privacy data attributes

1-Privacy data type

Describe collected or processed privacy data, such as name, address, phone

2-Purpose of collection

Describe the objective of the data collection and the business

3-Is it a must?

Is the data collection essential to keep the business application running?

4-Ways of collection

How the personal data is collected, such as API, email, or web form registration

5-Lawful basis

Is the data collection based on user agreement, contract, or legal compliance?

6-Rights of data subject

Can the data subject edit or delete the data?

7-Transmission

How the data is transmitted, such as FTP, email, or API

8-Storage country

Which country is the data stored in?

9-Storage format

In what format is the data stored, such as big data, relational database, or paper-based?

10-Expiration period

Any specified expiration period of the data usage?

11-Cross-border transfer

Will the data be transferred out of or into the EU?

12-Third-party involvement

Is any third party involved with the data processing?

13-Owner

Who/which team is the owner of the data?

Module 5: Security Architecture and Design Principles

Security architecture design principles

	Security by design	Privacy by design
Primary concerns	Unauthorized access to the system.	Authorized process of privacy data.
	According to OWASP, security by design principles are the following: • Minimize attack surface area • Establish secure defaults • Principle of least privilege • Principle of defense in depth • Fail securely • Don't trust services • Separation of duties	Referring to OECD Privacy Principles, the term privacy by design is defined by eight principles: • Collection Limitation Principle • Data Quality Principle • Purpose Specification Principle • Use Limitation Principle • Security Safeguards Principle • Openness Principle • Individual Participation Principle • Accountability Principle

	• Avoid security by obscurity • Keep security simple • Fix security issues correctly	
Examples of controls	• Access control • Unsuccessful login attempts • Session control • Timestamps • Non-repudiation • Configuration change control • Audit security events • Cryptographic module • Incident monitoring • Error handling	• Cookie • Anonymity • Consent • Obfuscation • Restrict • Notify and inform • Authentication • Minimization • Separation • Encryption • Data masking

The following industry references may help you to build a secure architecture:

- Open Security Architecture (OSA) Patterns: http://www.opensecurityarch itecture.org/
- CSA CAIQ (Consensus Assessment Initiative Questionnaire): https:/ /cloudsecurityalliance.org/group/consensus-assessments
- Google VSAQ (Vendor Security Assessment Questionnaires): https:// github.com/google/vsaq
- PCI Self-Assessment Questionnaire (SAQ): https://www.pcisecuritystanda rds.org/pci_security/completing_self_assessment
- NIST 1500-4 v4 Big Data Interoperability Framework Security and Privacy: https://www.nist.gov/publications/nist-big-data-interoperability-fram ework-volume-4-security-and-privacy
- NIST 800-122 Guide to Protecting the Confidentiality of Personally Identifiable Information (PII): https://csrc.nist.gov/publications/detail/sp /800-122/final

Cloud service security architecture reference

The Open Security Architecture (OSA) Patterns SP-011: Cloud Computing Pattern and SP-008: Public Web Server Pattern provide an overview diagram of the whole system. In addition, SP-001: client module and SP-002 Server module are also a good reference. Take a look at the components of the cloud computing pattern in the following link: http://www.op ensecurityarchitecture.org/cms/library/patternlandscape/251-pattern-cloud-computing

Java web security framework

1-Spring Security

- The Spring Security framework is only for Java- and Spring-based applications. It provides lots of out-ofbox security controls such as user authentication, CSRF attack protection, session fixation protection, a HTTP security header, and URL access control. Also, it supports various kinds of authentication such as Oauth2.0, CAS, and OpenID.

2-Shiro

- Shiro is a more lightweight and simple framework compared to Spring Security. The key difference between Shiro and Spring Security is that Shiro doesn't require a Spring-based application, and it can run standalone without tying into any web framework or a non-web environment.

3-Object Access Control (OACC)

- OACC primarily provides authentication and authorization. The key characteristic of OACC is that it provides a security relationship with application resources while Spring Security defines authorization by URL, methods, and roles.
- A security relationship example definition in OACC may be: (Sara) has (READ, EDIT) permissions on (TimeSheet.xls). Being able to establish the application resource (TimeSheet.xls) in a security relationship is a unique authorization model in OACC.

Non-Java web security frameworks

1-Node.JS

- Passport framework is an authentication module for Node.JS.

2-Ruby on Rails

- Devise Security: This is a security module for Ruby. It provides security features such as password complexity, CAPTCHA, user account inactivity checks, verification code, and session control for the web

3-ASP.NET

- ASP.NET Core provides security features such as authentication, authorization, anti-XSS, SSL enforcement, anti-request forgery, encryption, and also APIs to support GDPR.

4-Python

- Yosai is a security framework for Python applications
- Flask Security: It provides common security controls to Flask applications such as authentication, password hashing, and role management.

Web readiness for privacy protection

- **TLS for secure data transmission:** The misconfiguration of TLS may result in insecure data transmission or man-in-the-middle attacks.
- **Referrer Policy:** The Referrer Policy defines how the browser should handle Referrer information, which reveals the user's original visiting web site. The website visiting history is also considered to be personal privacy information.
- **Cookie Consent Disclaimer:** To comply with the GDPR, the collection of cookie information and the use of any third-party cookies will require explicit cookie consent.
- **HTTP Security Headers:** The HTTP protocol itself provides web security controls. Please also refer to the following table for the suggested HTTP security header configurations.

The following table summarizes the technical parts of privacy security requirements and suggested tools to assess and build the web:

1-Secure Communication: HTTPS by default and secure configuration of TLS.

- SSLyze, SSLScan, and TestSSLServer included in Pentest Box or Kali Linux

2-The origins of a visiting website source should not be leaked to other websites by the referrer header

- Referrer Policy defines how the referrer can be used. The configuration of the Referrer Policy depends on the requirements
- no-referrer will ensure the browser never sends the referer header.
- If the information is needed, it's suggested to configure sending information over HTTPS by using 'strictorigin'.

3-If Google Analytics is used, enable privacy extension to anonymize IPs.

- Enable IP masking for Google Analytics

4-Third-party cookies or embeds services (such as Google Analytics), with user consent.

- Cookie Consent
- Cookie Consent JavaScript plug-in: https ://github.com/insites/cookieconsent

5-HTTP Security Headers

The following are the suggested mandatory examples of secure http headers.

- Content-Security Policy (CSP) "defaultsrc 'self' "
- Referrer-Policy "no-referrer"
- Strict-Transport-Security "maxage=31536000"
- X-content-Type-options "nosniff"
- X-Frame-Options "SAMEORGIN"
- X-Xss-Protection "1;mode=block"
- Cookie "Secure"

Refer to OWASP Secure Headers Project for details of each security headers definition.

Privacy Score Assessment: https://privacyscore.org.

Login protection

Login protection can be seen as the first defense layer of the application.

If the number of login failures reaches a certain threshold level, the system should take action, such as banning the IP source:

Tools/modules for login protection are summarized in the table:

1-Detect the number of login failures in logs and take action

Fail2Ban

2-CAPTCHA solution to prevent machine brute-force login attacks

VisualCaptcha to build your own CAPTCHA service, Google reCAPTCHA

Cryptographic modules

The recommended encryption modules that the development team may need are shown here:

1-OpenSSL

- Full-featured and most widely used cryptography and SSL/TLS toolkit

2-Bouncy Castle Crypto APIs

- Lightweight cryptography Java API

3-mbed TLS

- OpenSSL alternative
- Cryptographic and SSL/TLS in embedded products
- Cryptography C API

4-SSLyze

- Verify the secure TLS configuration of the web server

In addition, an operation team may care more about the configuration of encryption on servers such as web servers, SSH, Mail, VPNs, database, proxy, and Kerberos.

Refer to Applied Crypto Hardening: https://betterCrypto.org/static/applied-crypto-hardening.pdf.

Input validation and sanitization

Input validation is like the perimeter security control of the whole application. The input not only includes data input from users but also covers the parameters passing between function calls, methods, APIs, or systems. The concept of validation covers various kinds of technical approaches:

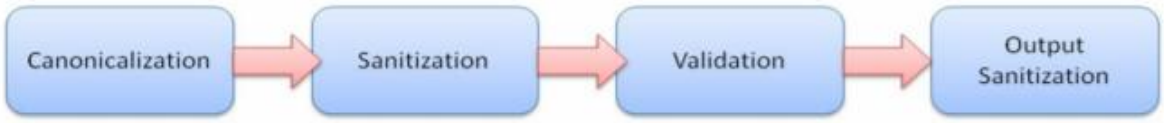
Techniques	**Purpose**	**Example**
Canonicalization Normalization	Process input data into known or expected form.	• URL decode/encode • File path or names handling
Sanitization	Sanitization is to remove illegal characters or make potentially risky	• Escape: replace < > ' " & with HTML entities.

	data safe. Always sanitize an output to avoid XSS.	
Validation	To check if the input is valid or within the constraint data type, length, and so on.	• IsAlpha, isCreditCard, isDecimal, isIP

Secure coding requires the following:

- Normalize strings before validating them
- Canonicalize path names before validating them
- Perform any string modifications before validation
- Canonicalize a URL before it is used

For example sanitization to prevent XSS:



For general canonicalization, sanitization, and validation, we can apply the APIs provided by the mature security framework, while the development team can focus more on business logic validation:

1-Java

- Java OWASP Java HTML Sanitizer

2-Ruby on Rails

- Active Record Validations

3-Node.js/JavaScript

- Validators

4-JavaScript

- DOMPurity to sanitize HTML and prevents XSS attacks

5-python

- Cerberus

Data masking

Data masking is the process of obfuscating original/sensitive data to protect it. There are five typical key scenarios that require data masking. Different tools are required based on different roles or usage scenarios:

Scenario	Involved roles	Required tools/modules
1-The application receives data and will do data masking based on defined policies	Developer	• Data masking modules • Data masking policies
2-Define the PII data tag and access policies	DBA	• PII metadata definition • PII access policies
3-Query results with data masking based on defined PII tags and access policies	Data query users	• Dynamic data masking
4-The operation team may monitor and check if there is any PII in data, files, configuration, or any unstructured data	Operation team	• PII data discovery
5-Any PII in the logs or files must be masked before further processing	Support team	• Data Anonymizer tools

Third-party open source management

An organization should set up its own internal open source and third-party software database and selection criteria. The database keeps records of open source or in-house developed components adopted in projects. It will provide a good framework selection reference for similar projects such as the web security framework we discussed earlier. If you are looking for an open source component search database, try Open Hub. You may search open source projects and find what you need for the project: https://www.openhub.net/.

Furthermore, the open source selection criteria help to reduce legal and quality risks. A typical criteria checklist is listed in the following table:

Selection criteria	Example and description
Does the open source community fix the security issue in a timely manner?	• High-security risks fixed within 1 month.
Adoption of latest and stable releases	• Official and stable release by the community.

Availability of technical support?	• The open source community provides official technical support and issues feedback.
Software licenses with GPL and LGPL are less preferred.	• Licenses with GPL and LGPL are not preferred. If any GPL software components are used, custom-developed source code may also need to be available as open source. • The binary analysis tool (BAT) is suggested for license scanning based on binary files: htt p://www.binaryanalysis.org/.
Vulnerability status and fixes	Search for the vulnerability status of the components. For more details, please visit https://nvd.nist.gov/vuln/search.
Software release or update frequency	Was the latest version released within 6 months or several years ago?
Software architecture	Is it using the latest software technologies or legacy framework?

For the security of open source components, the recommended security practices and tools during the DevOps stages are summarized in the table:

Stage	**Activities**	**Recommended Tools/Practices**
Design and Selection	Selection of Open Source.	• http://www.openhub.net/ • Open Source selection checklist • In-house Open source database
Package Delivery	Identify all the dependencies in the project and check known vulnerabilities.	• OWASP dependency check • OWASP dependency Track
Package Deployment	On-line services monitoring and scanning of CVE.	• CVE database (https://n vd.nist.gov/vuln/search) • NMAP or OpenVAS scanning

Also, refer to SAFECode Managing Security Risks Inherent in the Use of Third-party Components: https://www.safecode.org/wp-content/uploads/2017/05/SAFECode_TPC_Whitepaper.pdf.

Module 6: Threat Modeling Practices and Secure Design

Threat modeling with STRIDE

The STRIDE threat model defines threats in six categories, which are spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privilege. It's normally used to assess the architecture design.

The threat STRIDE model and general security mitigation are summarized in the following table. In addition to STRIDE, it's also suggested to include privacy in the analysis:

STRIDE threats	**Mitigation**
Spoofing	Authentication such as credentials, certificates, and SSH
Tampering	Integrity (HASH256, digital signature)
Repudiation	Authentication, logging
Information Disclosure	Confidentiality (encryption, ACL)
Denial of Service	Availability (load balance, buffer, message queue)
Elevation of Privilege	Authorization (ACL)
Privacy (additionally included)	Data masking, access control, user consent, removal

Refer to OWASP Application Threat Modeling for more examples based on the DFD diagram: https://www.owasp.org/index.php/Application_Threat_Modeling.

Diagram designer tool

The Microsoft Threat Modeling tool, OWASP Threat Dragon, and Mozilla SeaSponge are the tools in this category that allow you to draw DFD diagrams with threat analysis:

- Microsoft Threat Modeling Tool: https://www.microsoft.com/en-us/download/de tails.aspx?id=49168
- OWASP Threat Dragon: https://www.owasp.org/index.php/OWASP_Threat_Dragon
- Mozilla SeaSponge: http://mozilla.github.io/seasponge/

Threat library references

1-CAPEC

It lists 508 attack patterns in a tree view. The attack patterns are also available in CSV and XML format. Each attack pattern is labeled with a CAPEC-ID number.

2-ATT&CK

The threats are categorized by platform (Linux, Windows, Mac, mobile) with specific attack techniques. Each threat is also discussed with technical mitigation and detection approaches. It includes lots of practical hacker and malware attack techniques.

3-CWE

CWE is a list of software weaknesses. Each CWE is categorized into a threat tree view and presented with both insecure and secure source code implementations. It's also a very good reference for secure coding

Module 7: Secure Coding Best Practices

Secure coding industry best practices

Depending on programming languages, secure coding standards are summarized in the following table:

1-CERT Secure Coding

This provides secure coding standards for C, C++, Java, Perl, and Android

2-Find Security Bugs

This provides bug patterns with samples of vulnerable code and solution for Java.

3-CWE

This provides vulnerable source code samples from the perspective of common software weaknesses. The coding samples cover C, C++, Java, and PHP.

4-Android

Android Application Secure Design and Secure Coding Guidebook

5-OWASP SKF

- OWASP Security Knowledge Framework.
- It can be used as an internal security knowledge base, which includes OWASP ASVS and secure coding knowledge.

6-PHP Security

- OWASP PHP Security Cheat Sheet

7-OWASP Code Review

- OWASP Code Review Project

8-Apple Secure Coding Guide

- Apple Secure Coding Guide

9-Go

- Go Secure Coding Practices for GO language

10-JavaScript

- JavaScript Secure Coding Practices

11-Python

- OWASP Python Security Project

Secure coding awareness training

A case study or scenario-based vulnerable source code example will have better training effects than simply secure coding rules. The following are good references in this area and provide a lot of vulnerable and secure best practice code samples:

- OWASP Security Knowledge Framework: https://www.securityknowledgef ramework.org/
- Android Application Secure Design and Secure Coding Guidebook: http://www.jssec.org/dl/android_securecoding_en.pdf
- Find Security Bugs Patterns for Java: https://find-sec-bugs.github.io/
- OWASP Teammentor: https://owasp.teammentor.net/angular/user/index

High-risk module review

The following table lists high-risk modules that require further review:

1-Authentication

- Accounts registration
- Login and CAPTCHA
- Password recovery or reset
- Password changes
- Identity and password storage and access control
- Account lockout control after multiple failures

2-Authorization

- Sensitive resource access
- Administration management

3-Configuration

There are two kinds of review in the configuration:

- Secure configurations of the applications, such as turning off debug mode and enabling TLS communication.
- The impact of the configuration for each software release

4-Finance

- Payment functions
- Order and shopping carts

5-File handling

- File upload
- File download

6-Database

- Database query
- Database add, update, and delete

7-API interface

- Restful API interfaces
- Third-party integration interfaces

8-Legacy

- Modules that don't support secure communication
- Modules that may still use weak encryption algorithms
- Uses of banned or dangerous APIs

9-Encryption

- Uses of banned encryption algorithms
- Hardcoded sensitive information or comments in the source code during development, such as IP, email, password, or hidden hotkey

10-Session

- Concurrent session control and detection
- The randomness of the session ID and expiration period

Secure code scanning tools

Here are some commonly used open-source secure coding analysis tools, as in 2018. Note that we only list open source tools here:

Tools	Background and key characteristics of the scanning tool
C/C++	• Infer • CPP Check • Flawfinder • Clang Static Analyzer
Java	• Infer • SpotBugs • PMD

Android	• MobSF
PHP	• Phan • PHPMD
Ruby	• DawnScanner
Python	• Pylint
JavaScript	• ESLint • JSHint • Retire.JS • PMD
Dependencies vulnerabilities	• OWASP Dependency check • PHP Security Checker • Retire.JS
Language-independent	• SonarQube • DREK • Graudit • VisualCodeGrepper

Secure compiling

The common secure options are summarized in the following table:

Protection techniques	**Secure options**	**OS/Compiler**
Address Space Layout Randomization (ASLR)	echo 1 >/proc/sys/kernel/randomize_va_space	Android, Linux OS
Stack-based buffer overrun protection	-fstack-protector –fstack-protector-all	gcc
GOT Table Protection	-Wl,-z, relro	gcc
Dynamic link path	-Wl,--disable-new-dtags,--rpath [path]	gcc
Nonexecutable stack	-Wl,-z,noexecstack	gcc
Image randomization	–fpie –pie	gcc
Insecure C runtime function detection	–D_FORTIFY_SOURCE=2 –Wformat-security	gcc
Stack-based buffer overrun defenses (Canary)	/GS	MS Visual C++
Address Space Layout Randomization (ASLR)	/DYNAMICBASE	MS Visual C++

CPU-level NoeXecute (NX) support. Data Execution Prevention (DEP)	/NXCOMAT	MS Visual C++
Safe-structured exception handling	/SAFESEH	MS Visual C++
Enable additional security check	/SDL	MS Visual C++

For further reference and a description of each protection technique, here are some references:

- SAFECode Development Practices: https://www.safecode.org/publication/SAFECode_Dev_Practices0211.pdf
- OWASP C-based ToolChain Hardening: https://www.owasp.org/index.php/C-Based_Toolchain_Hardening
- Linux Audit ASLR: https://linux-audit.com/linux-aslr-and-kernelrandomize_va _space-setting/
- MS Security Best Practice for C++: https://msdn.microsoft.com/en-us/library/k3a3hzw7.aspx
- Secure Compiler and linker flags for GCC: https://developers.redhat.com/blog/2018/03/21/compiler-and-linker-flags-gcc/

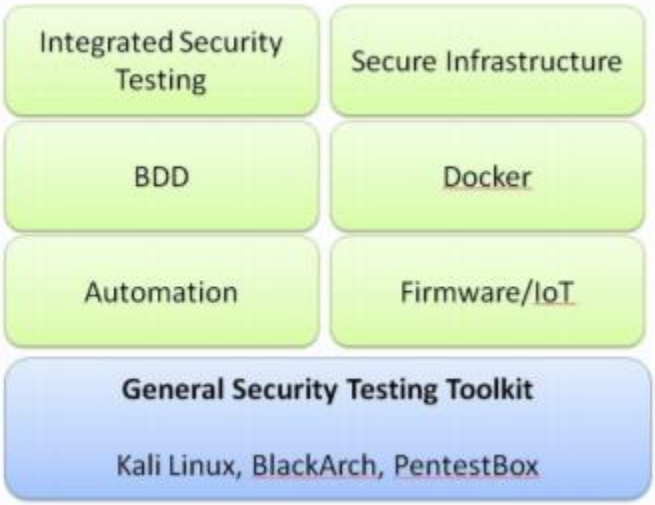
To verify whether the application or the environment has been configured with secure options, the following tools are useful:

- CheckSec: http://www.trapkit.de/tools/checksec.html
- BinScope: https://www.microsoft.com/en-us/download/details.aspx?id=44995

Module 8: Security Testing Toolkits

General security testing toolkits

The objective of providing security testing toolkits is for project teams to understand what tools are available and apply the tools that they judge to be appropriate based on the business application scenario. There are many kinds of security testing tools. An organization may define one general testing toolkit for all projects, and also suggest other security testing tools based on those specific domains, such as automation, infrastructure, Docker, and BDD:



The following table shows the recommended minimum security testing toolset (only open source or free tools are listed here):

1-WhiteBox review

GraudIT or GREP-IT

These tools are recommended because they don't require a whole buildable source code to identify the security issue for different programming languages:

- GraudIT: https://github.com/wireghoul/graudit
- GREP-IT: https://github.com/floyd-fuh/crass/blob/master /grep-it.sh

2-Web

BurpSuite, OWASP ZAP, Vega, SQLmap, Arachni

3-Vulnerability

Nessus, OpenVAS, OpenSCAP, NMAP

4-Networking

NMAP, WireShark, TCPDump, Hping, SSLScan, SSLyze, masscan

Automation testing criteria

The web security testing can be automatically triggered every time the build is submitted. To be able to integrate web security testing tools with Jenkins, there are several key criteria that we need to consider:

- **Command console:** Most security testing tools provide a command console or GUI interface to operate the security testing procedures. It would be ideal for the tool to provide both interfaces. The command console can be used for Jenkins to trigger the execution of the security testing, and the GUI can help the human testing. From the automated testing point of view, the command-line interface (CLI) is a minimum

requirement to integrate with Jenkins. The CLI interface also helps us to integrate with the unit test framework or BDD framework.

- **API interface:** The web security testing can be executed in a standalone attacker mode or a proxy mode. The API interface will allow us to interact with the testing tool programmatically during runtime. For example, the OWASP ZAP provides a REST API to automate all the operations using Python and also the ZAP CLI to interact with ZAP from the command line.
- **Output formats:** Most web security testing tools provide different kinds of reporting formats, such as HTML, PDF, XML, CSV, JSON, or console output. CSV, JSON, and XML are considered the basics if we would like to import the testing results together. Because of the various security tools and large quantities of results in the daily report, it's suggested that you apply integrated security testing tools, such as OWASP DefectDojo, to consolidate all the testing results in one dashboard (this option will be discussed later). In addition, some tools may provide the Jenkins plugin, which can help you to output the results in the Jenkins management console.

Just be aware that the web security automated test can't complete all web security tasks. Some testing scenarios still require a human security tester to guide the tool and perform further verification, such as authentication, web page authorization, business logic-related tests, and multiple order submissions. The following table displays the tools and their features:

	Web GUI	**CLI**	**REST API**
OWASP ZAP	Yes	ZAP CLI	ZAP API
Arachni	Yes	Yes	Yes (It also provides Ruby libraries.)
W3af	Yes	Yes	Yes
Nikto	n/a	Yes	n/a
Wapiti	n/a	Yes	n/a

Android security testing

Generally, the following are considered common testing tools when it comes to Android security testing:

1-ApkTool

ApkTool is used to perform reverse engineering for Android APK files

2-ByteCode View

ByteCode View is a Java Bytecode viewer and GUI Java decompiler.

3-Dex2JAR

Dex2JAR converts the DEX to a CLASS file.

4-JADX

JADX converts the DEX to a Java decompiler.

5-JD-GUI

JD-GUI is a GUI viewer that is used to read the source code of CLASS files.

6-Drozer

Drozer is an interactive security and attacks framework for the Android app.

7-Baksmali

Baksmali is an assembler/disassembler for the DEX format.

8-AndroBugs

AndroBugs takes an APK file as input and performs an APK security vulnerabilities scan.

9-AndroGuard

AndroGuard is a Python framework that can perform reverse engineering and malware analysis of the APK.

10-QARK

Quick Android Review Kit (QARK) works similarly to AndroBugs. It detects security vulnerabilities for APK files.

11-AppMon

AppMon can monitor API calls for both iOS and Android apps.

To install and configure the tools separately can be very time-consuming, so it is suggested that you use the following toolkits, which have most of the Android security testing tools preinstalled:

1-AndroL4b

2-Appie

3-PentestBox

Securing infrastructure configuration

Securing the infrastructure configuration is vital in ensuring that the infrastructure configurations and system hardening are compliant with industry security best practices, such as CIS benchmarks, PCI-DSS, and the **National Checklist Program (NCP)**.

If the DevOps team have applied infrastructure tools, such as Chef or Puppet, it's highly recommended that you define the security configuration on top of these tools to achieve the goal of **infrastructure security as code**.

This helps to move the infrastructure security from the operation stage to the development stage. The Inspec, Hardening Framework, and ServerSpec tools are tools that are used for checking infrastructure security configurations. You can learn more about them at the following links:

- Inspec: https://www.inspec.io/
- Hardening Framework: https://Dev-Sec.io
- Serverspec: https://serverSpec.org/

For an infrastructure environment that is not deployed with configuration

management tools (Puppet, Chef, Ansible, or SaltStack), the following

scanning tools are suggested:

- Lynis Security Auditing: https://github.com/CISOfy/lynis
- OpenSCAP: https://www.open-scap.org/
- CIS Benchmarks: https://www.cisecurity.org/cis-benchmarks/

For a sample of the scanning result of OpenSCAP, go to https://www.open-scap.o rg/wp-content/uploads/2015/09/ssg-rhel7-ds-xccdf.report.html.

Docker security scanning

Generally speaking, there are three kinds of Docker security tools that do one of three different things:

- Scan for Docker security best practices based on CIS (Docker Bench, Actuary)
- Scan for known common vulnerabilities and exposures (CVEs) (Claire, Anchor Engine)
- Runtime threat analysis (Falco, Dagda)

Here are the open source security testing tools for Docker security:

1-Docker Bench

Docker Bench is an automated script that checks the Docker security best practices compliance.

The scanning rules are based on the CIS Docker Security Benchmark.

- Docker Bench Security: https://github.com/docker/docker-benc h-security/
- Docker Benchmark: https://benchmarks.cisecurity.org/

2-Actuary

Actuary works similarly to Docker Bench. Additionally, Actuary can scan based on user-defined security profiles from the Docker security community.

- Actuary: https://github.com/diogomonica/actuary/

3-Clair

Clair is a container image security static analyzer for CVEs.

- Clair: https://github.com/coreos/clair

4-Anchor Engine, Anchor Cloud

Anchor Cloud and Anchor Engine scan the Docker images for CVEs. Anchor Engine is a hosted tool and Anchor Cloud is a cloud-based tool.

- Anchor Engine: https://github.com/anchore/anchore-engine
- Anchor Cloud: https://Anchore.com/cloud/

5-Falco

Falco is a Docker container runtime security tool that can detect anomalous activities.

- Falco: https://sysdig.com/opensource/falco/

6-Dagda

Dagda is an integrated Docker security tool that provides runtime anomalous activities detection (Sysdig Falco), vulnerabilities (CVEs) analysis (OWASP dependency check, Retire.JS), and malware scanning (CalmAV).

- Dagda: https://github.com/eliasgranderubio/dagda/

Integrated security tools

If you are looking for such an integrated security testing management tool, here are some of the open source and free tools to consider:

1-JackHammer

JackHammer, provided by Ola, is an integrated security testing tool. It provides you with a dashboard to consolidate all the testing results. The key difference is that JackHammer includes mobile app security scanning and source code static analysis tools. The supported open source security scanners include Brakeman, Bundler-Audit, Dawnscanner, FindSecurityBugs, PMD, RetireJS, Arachni, Trufflehog, Androbugs, Androguard, and NMAP. The following screenshots show a typical example of its integrated interface.

- JackHammer: https://github.com/olacabs/jackhammer
- Additional information: https://jch.olacabs.com/userguide/

2-Faraday

Faraday is an integrated penetration testing environment, and provides a dashboard for all the testing results. It integrates with over 50 security tools.

- Faraday: https://www.faradaysec.com/#why-faraday
- Additional information: https://github.com/infobyte/farada y/wiki/Plugin-List

3-Mozilla Minion

Mozilla Minion is also an integrated security testing tool that includes the following plugins by default:

- ZAP
- Nmap
- Skipfish
- SSLScan

You can find Mozilla Minion at https://github.com/mozilla/min ion/

4-Penetration testing toolkit

Penetration testing toolkit provides a unified web interface for many Linux scanning tools, such as nmap, nikto, WhatWeb, SSLyze, fping, URLCrazy, lynx, mtr, nbtscan, automater, and shellinabox.

- Penetration testing toolkit: https://github.com/veerupandey/ Penetration-Testing-Toolkit

5-Seccubus

The key advantage of using Seccubus is that it integrates with various kinds of vulnerability scanner testing results, and also compares the differences between each scan. It includes the following scanners:

- Nessus
- OpenVAS
- NMAP
- Nikto
- Medusa
- SSLyze
- SSL Labs
- TestSSL.sh
- SkipFish
- ZAP

You can find Seccubus at https://github.com/schubergphilis/Sec cubus.

6-OWTF

Offensive Web Testing Framework (OWTF) is an integrated security testing standards OWASP testing guide and includes PTES and NIST tools.

- OWTF: https://owtf.github.io/
- Additional information: https://owtf.github.io/online-pass ive-scanner/

7-RapidScan

RapidScan is a mult-itool that includes a web-vulnerability scanner. The security scanning tools that it contains include nmap, dnsrecon, uniscan, sslyze, fierce, theharvester, and golismero.

8-DefectDojo

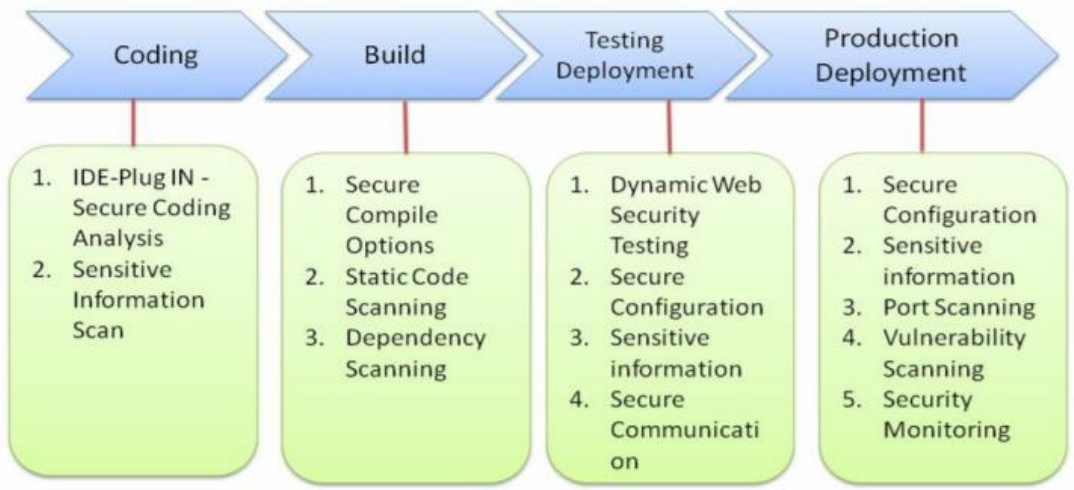
The OWASP DefectDojo is a security tool that can import and consolidate various security testing tool outputs into one management dashboard

DefectDojo: https://github.com/DefectDojo/django-DefectDojo

Module 9: Security Automation with the CI Pipeline

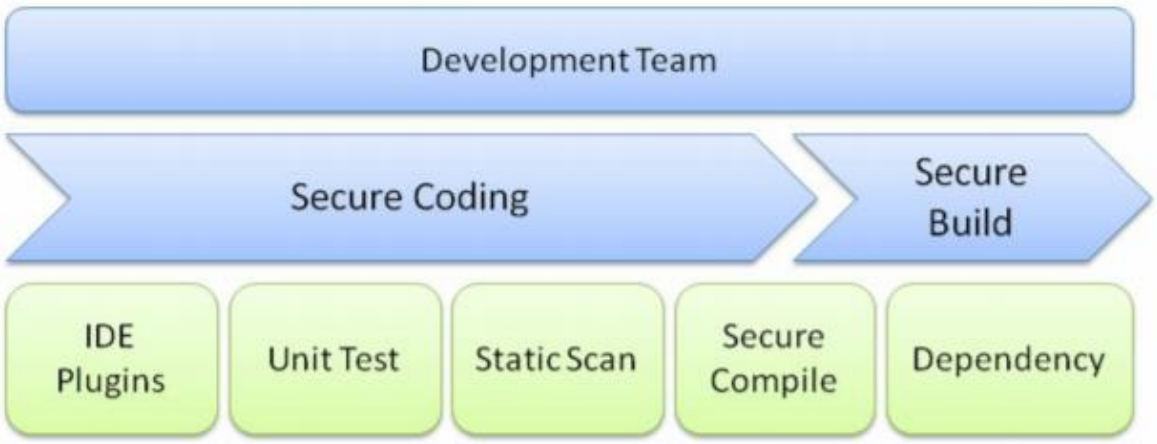
Security in continuous integration

The following diagram shows the security practices in each phase, namely, coding, build, testing, and production deployment:



Security practices in development

The following diagram shows the overall security practices we can plan into the development stage. We will introduce some of the open source security tools and practices for these security activities in the upcoming sections:



Dependency check

The following tools will help you scan for vulnerable components:

1-OWASP Dependency Check

The OWASP Dependency Check scans for dependency vulnerabilities in Java, Ruby, PHP, JavaScript, Python, and .NET.

2-Retire.JS

Retire.JS scans for vulnerable JavaScript libraries

3-Snyk

Snyk scans for the JS, Ruby, Python, Java vulnerabilities.

Web automation testing tips

The following table contains some suggested tips to improve the testing efficiency and effectiveness for uses of web automation testing tools, such as ZAP or Arachni:

Testing tips	Description

Integration	To do automated integration, try to understand that the web security tools provide the following: • Headless execution mode • Command-line interface • REST API • Jenkins plugin (this may be optional as long as one of te preceding tools is provided) For example, the OWASP ZAP (https://github.com/Grunny/z ap-cli/) provides the ZAP CLI interface, which also helps make the integration easier.
Authorization testing	To test the guest, user, and admin permissions for every web service's URL or resources will require proper predefined navigation workflows. The testing scenario may include the following: • Session fixation, reuses, expiration • User, role, guest, administration permissions • Login, logout, and reauthentication behaviors There are two main approaches for the security testing: • Use Selenium or Robot Framework to do the authentication and use OWASP ZAP to detect the security issue • Preconfigure the pages or URLs that require authentication in OWASP ZAP or Arachni
Scanning scope	Dynamic web testing may take a very long period of time. Properly configure the scanning scope to include or exclude the URLs that are being tested. Only apply a complete full scan when the application passes the smoke testing. A complete scan can be scheduled to be done on a nightly basis
API fuzz	The web service may provide several REST JSON or SOAP XML APIs. It's suggested that you get a complete API list and specifications. Do the fuzz testing on the parameters of each API. Once this has been done, run the OWASP ZAP or the Arachni in proxy mode to intercept all the API calls. Then, investigate these API calls for further fuzz testing with the parameters in the payload. For the fuzz security payload test, consider replacing the value of the parameters with the following data in the fuzzDB: • https://github.com/fuzzdb-project/fuzzdb/ • https://github.com/minimaxir/big-list-of-naughty-strings/ Radamsa can be used to automatically generate fuzzing data: • https://github.com/aoh/radamsa
Business logic	Some web UI workflows need to be operated in order, such as shopping for items, ordering the items, and payment. Here are some approaches to help you handle this kind of security testing: • Use existing functional Selenium automation UI testing and run the OWASP ZAP or Arachni in proxy attack mode. • Use the script provided by OWASP ZAP to integrate with Selenium. Refer to the Zap webdriver (https:/github.com/continuumsecurity/zap-webdriver) as an example • Apply the BDD Security framework (https://github.c om/continuumsecurity/bdd-security/).

	• Manually operate the web pages to navigate the flow and save the ZAP sessions for further security scanning.

Security automation in Jenkins

The following table shows an example of how to configure the command-line ZAP, which can be triggered periodically and remotely by a predefined URL:

Steps	**Configuration steps**
New item	New Item | Enter an Item Name | "Security Testing" | Freestyle Project | OK
General	Project Name: "Security Testing"
Build Trigger	The automation testing can be triggered by the schedule in the following ways. The Build Trigger defines how the tasks can be triggered. There are two modes supported: one is the scheduled mode and the other is the remote trigger by the REST API: Build Periodically: 45 9-17/2 * * 1-5 The automation testing can also be triggered remotely by sending the HTTP request: Trigger builds remotely: ZAP Once it's defined, this will be the URL that can be triggered remotely to kick off the automation execution: https:///job/Security Testing/build?token=ZAP
Build	Build | Add Build Step Execute the Windows batch command: echo ---- the execution of OWASP ZAP for Active Scan---- zap cli active-scan http://targetWeb/ echo ---- The end of OWASP ZAP active Scan ----

The following table lists the common Jenkins plugins that are related to software security assessment:

Jenkins Security plugins	**Description**
ZAP	ZAP is a dynamic web scanning tool.
Arachni Scanner	Arachni Scanner is a dynamic web scanning tool.
Dependency Check plugin	The Dependency Check plugin detects vulnerable dependency components.
FindBugs	FindBugs is a static code analysis tool for Java.

SonarQube	SonarQube is a code quality analysis tool.
360 FireLine	360 FireLine is a static code scanner for Java.
HTML Publisher Plugin	The HTML Publisher plugin generates the testing results in HTML
Log Parser Plugin	The Log Parser plugin parses the testing results of the security testing tools, such as the number of XSS detected or the number of errors.
Static Analysis Collector	The Static Analysis Collector plugin can consolidate the results from all other static code analysis plugins, such as Checkstyle, Dry, FindBugs, PMD, and Android Lin.

Module 10: Incident Response

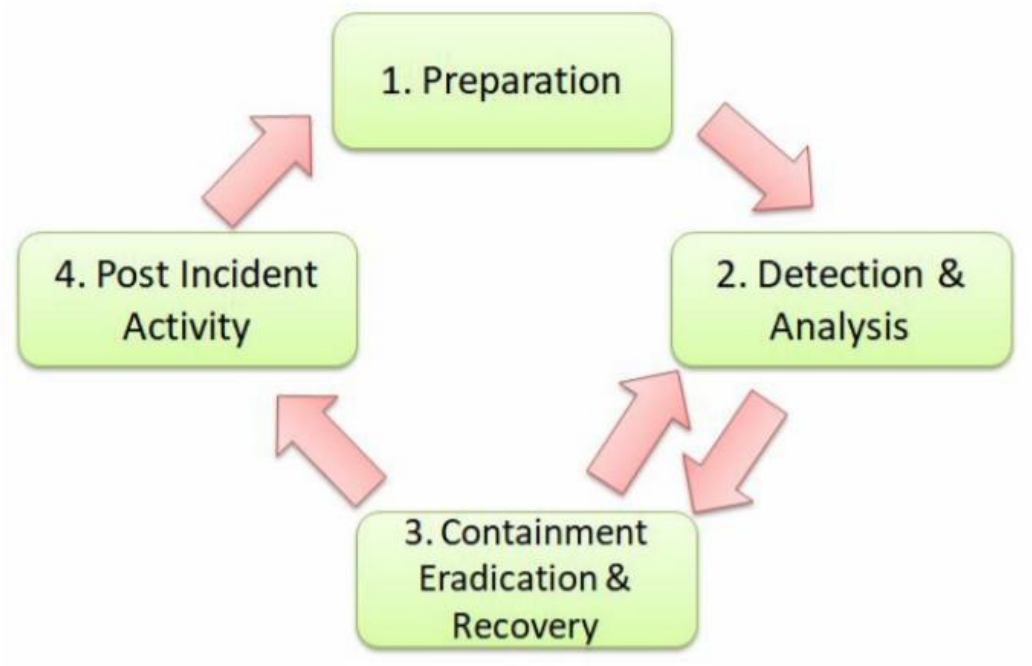
Security incident response process

Here are some of the recommended industry references related to security incident response:

- NIST SP 800-62 Computer Security Incident Handling Guide (https://cs rc.nist.gov/publications/detail/sp/800-61/rev-2/final)
- SANS Incident Handler Handbook (https://www.sans.org/reading-room/whitep apers/incident/incident-handlers-handbook-33901)
- ENISA Cloud Computing Benefits, risks, and recommendations for information security (https://resilience.enisa.europa.eu/cloud-security-and-res ilience/publications/cloud-computing-benefits-risks-and-recommendations-for-infor mation-security)
- MITRE Ten Strategies of a World-Class Cyber Security Operations Center (https://www.mitre.org/sites/default/files/publications/pr-13-1028-mitre10-strategies-cyber-ops-center.pdf)
- FIRST (https://www.first.org/education/FIRST_PSIRT_Service_Framework_v1.0)

NIST SP 800-62 defines the incident response life cycle as consisting of four phases: preparation, detection and analysis, containment eradication and recovery, and post-incident activity. We will introduce some practical tools

for each phase in the upcoming sections:



Preparation

Here are some suggested security practices to be performed in the incident response preparation phase:

- Incident handler communication plan
- Incident analysis hardware and software tools (refer to the section on incident forensics)
- Existing networking diagram and baselines
- Prevention controls, such as risk assessments, host security, network security, malware protection, user awareness, and training (refer to the CIS security controls)
- The blue and red team security exercise (refer to the following table)
- Bounty program for whitehat hackers or security researchers to submit security issues

The following open source tools can help to generate an internal attack simulation without compromising business operations. These tools don't generate real attack samples, but simulate the behaviors of hacking or advanced persistent threat (APT) behaviors:

Tools	**Simulation of APT**
DumpsterFire	The DumpsterFire tool includes various kinds of simulated attack scenarios, such as an account attack, file download, drop files, command execution, and web access in Python. It provides a user-friendly menu to customize the security incidents, even for those who don't understand Python.
METTA	The METTA tool allows the security team to customize the simulation of APT attacks based on MITRE ATT&CK. The simulated APT behaviors

	defined by YAML include credential access, evasion, discovery, execution, exfiltration, lateral movement, persistence, and privilege escalation.
Red Team Automation (RTA)	The Red Team Automation tool is a collection of Python and PowerShell scripts that can simulate over 50 malicious behaviors based on ATT&CK.
Atomic Red Team (ART)	The Atomic Red Team tool provides Windows, macOS, and Linux shell scripts to simulate the MITRE ATT&CK.
APT Simulator	The APT Simulator tool is a collection of Windows BAT scripts that simulate APT behaviors.
Network Flight Simulator	The Network Flight Simulator tool can be used to generate malicious network traffic, such as DNS tunneling, C2 communication, DGA traffic, and port scans.

Detection and analysis

Identifying the signs of a security incident requires the deployment of various security solutions and log sensors. The sources of infections include IDS/IPS, SIEM, antivirus, file-integrity monitoring, OS/network logs, and public and known vulnerabilities. The deployment of the whole enterprise's security controls may refer to the CIS Critical Security Controls for Effective Cyber Defense (you can find the information at https://www.cisecurity.org/controls/).

These consist of 20 security controls, as summarized in the following table. There are many commercial solutions in each security control, but only open source solutions are listed in the table:

Cybersecurity controls	**Examples of security techniques and open source tools**
CSC1: Inventory of Authorized and Unauthorized Devices	Endpoint security, Asset Management
CSC2: Inventory of Authorized and Unauthorized Software	Endpoint security, Asset Management
CS3: Secure Configurations for Hardware and Software on Mobile Devices, Laptops, Workstations, and Servers	CIS Security Benchmark, OpenSCAP.
CSC4: Continuous Vulnerability Assessment and Remediation	OpenVAS Nmap OWASP Dependency Check OWASP Dependency-Track vulscan
CSC 5: Controlled Use of Administrative Privileges	Strong password complexity

	Auditing logs for root and administrator activities
CSC 6: Maintenance, Monitoring, and Analysis of Audit Logs	Syslog, Event Logs, SIEM ELK GrayLog Security Onion Malicious Traffic Detection
CSC 7: Email and Web Browser Protections	Email Protection, AntiSpam, Web Application Firewall ModSecurity Email Encryption Scramble Linux Malware Detection
CSC 8: Malware Defenses	Endpoint Protection, Antivirus, HIDS/HIPS OSSEC ClamAV
CSC 9: Limitation and Control of Network Ports, Protocols, and Services	Nmap OpenSCAP
CSC 10: Data Recovery Capability	Bacula
CSC 11: Secure Configurations for Network Devices, such as Firewalls, Routers, and Switches	CIS Security Benchmark
CSC 12: Boundary Defense	Firewall, IPS, HoneyPot Security Onion
CSC 13: Data Protection	OSQuery Data Vault
CSC 14: Controlled Access Based on the Need to Know	Data Classification, Firewall, VLAN, Logging
CSC 15: Wireless Access Control	VPN, SSL Certificate, WAP2
CSC 16: Account Monitoring and Control	Log Analysis Tools Fail2ban
CSC 17: Security Skills Assessment and Appropriate Training to Fill Gaps	Security Training and Labs Resource CybraryIT
CSC 18: Application Software Security	OWASP
CSC 19: Incident Response and Management	NIST SP800-61 Computer Security Incident Handling Guide FIR (Fast Incident Response)

CSC 20: Penetration Tests and Red Team Exercises	Refer to some of the open source tools we suggested in the Preparation section

Containment and recovery

For the containment, there are typical network- or host-containment criteria established by network policy enforcement. Whenever one of the criteria is met, the containment actions can include blocking that specific host, redirecting the traffic to apply the latest security patches, and rejecting specific communication traffic or ports.

The following are common security policy enforcement criteria that will trigger the network or host containment:

- The host hasn't installed any antivirus products.
- The antivirus pattern/engine versions are not updated.
- There are known vulnerable components on the host.
- There is suspicious communication traffic on the specified ports.
- A known virus is detected on the hosts.
- There is outgoing communication to an external known malicious IP or domain. Refer to the following resources:
  http://iplists.firehol.org/
  https://www.spamhaus.org/drop/
  https://rules.emergingthreats.net/fwrules/emerging-Block-IPs.txt
  https://check.torproject.org/exit-addresses

Security incident response platforms (SIRP)

When handling a security incident, there will be lots of information that needs to be processed and analyzed. An ideal security incident response platform should be able to do the following:

- Receive alerts and security events from different sources (SIEM, IDS, email)
- The security incident case management should allow a security analyst to add related logs, IOCs, or findings during the incident case handling life cycle
- Compare its analysis with external threat information, such as VirusTotal, to identify the malicious behaviors of a file, hash, domain, or IP address

The open source tool TheHive can help you to provide a security incident response management platform. TheHive can also work with MISP, which is a threat intelligence platform for sharing and correlating indicators of compromise (which indicate that a targeted attack has taken place) and vulnerability information. Refer to the following documentation for more information:

- https://thehive-project.org/
- http://www.misp-project.org/index.html

For more information on how TheHive, CorTex, and MISP can integrate together for a threat incident response, go to https://blog.thehive-project.org/201 7/06/19/thehive-cortex-and-misp-how-they-all-fit-together/.

SOC team

The security operations center (SOC), also known as the computer incident response team (CIRT), is the security team that handles and monitors daily security events.

The organizational structure of SOC can include parts of the existing IT team, an outsourced team, or a dedicated security team. No matter what kind of structure it has, there are several key functions that the team will have:

1-Security incident analysis and forensics (call center)

This function team may include the Tier 1 case handling in the 24/7 security monitoring center. The Tier 1 team typically handles the case by following the predefined checklist or SOP to perform initial root-cause analysis or mitigation based on the incidents.

2-Security operations and administration

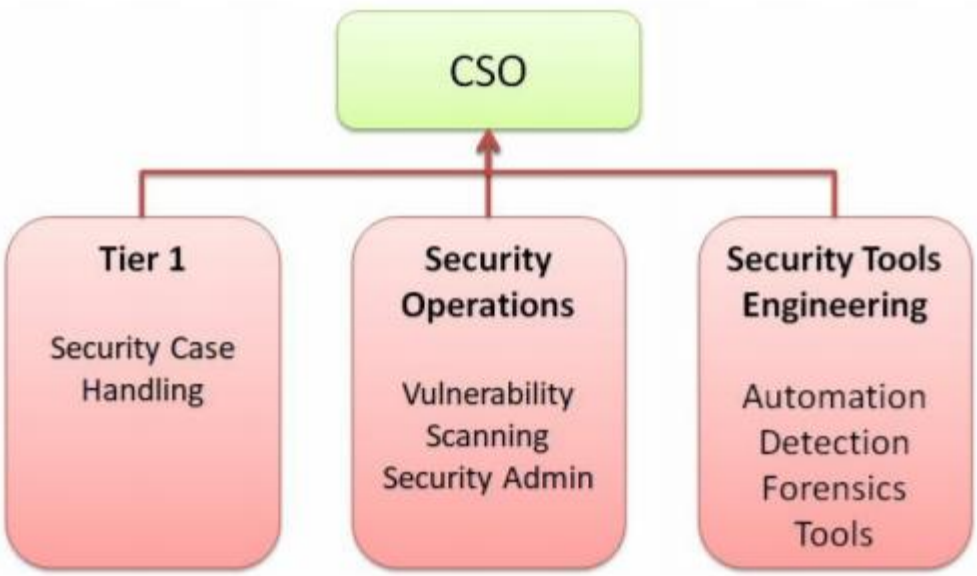
This functional team involves the following routine security activities. These are regular security checking activities for the production environments:

- Network scanning (weekly)
- Vulnerability scanning (weekly)
- Penetration testing (monthly)
- Security awareness training (bi-monthly)
- Security log trending analysis (monthly)
- Security administration and monitoring (daily)
- Patch or security signature update (daily/weekly)

3-Security tools engineering

The security engineering team implement security tools for the security call center or security operations team. The security tools can be security automation, suspicious behaviors detectors, forensic analysis tools, security configurations checker, threat intelligence integration, threat signatures creation, and so on.

The SOC team can consist of parts of an IT call center or a dedicated security team depending on the size of the whole organization. A typical dedicated SOC team structure is shown in the following diagram:



Incident forensics techniques

The NIST SP 800-86 Guide to Integrating Forensic Techniques into Incident Response defines four major phases to perform digital forensics on a compromised computer:

- Collection: Collect all the relevant logs of the compromised computer or networking activities logs
- Examination: Extract and correlate the information that may highly relate to suspicious behaviors
- Analysis: Analyze all the information for root causes of the malicious infection
- Reporting: Conclude the summary results

The forensics techniques require the capability of the incident response team to perform the analysis. In the following table, we have listed some quickwin solutions that can perform semi-automated forensics, including collection, examination, and analysis:

Category	**Tools**	**Purpose and usage scenario**
Log Collection	OSX Collector	Mac OS X Log Collector is an automated forensic evidence collection for macOS. The Python script, osxcollector.py, is the script that performs all the collection jobs. The tool will generate a JSON file as a summary of the collected information.
Log Collection	IR Rescue	IR Rescue is a Windows and Linux script for collecting host forensic data. For the Windows version, it integrates several utilities from the from Sysinternals and NirSoft.
Log Collection	FastIR Collector	FastIR Collector (for Linux) only requires one Python script to collect all related logs in Linux.

		For Windows systems, it will require additional modules and tools. Refer to https:/ /github.com/SekoiaLab/Fastir_Collector for more information.
Malware Detector	Linux Malware Scanner	Free malware scanners for Linux are available from the following links: CalmAV: It's an open source antivirus software for Windows. Linux Malware Detect (LMD): It's an open source antivirus software for Linux.
Suspicious Files Analysis	Cuckoo	Cuckoo is an automated malware analysis system. It can analyze the dynamic runtime and static behaviors of the unknown and suspicious files under Windows, Linux,macOS, and Android.
Client/Server log collector and analysis	GRR Rapid Response	You can use Google Remote Live forensics for incident response. It will require the installation of a Python agent on the target hosts to collect the logs and on the Python server to do the analysis.
Client/Server log collector and analysis	OSQuery	The OSQuery tool works in a similar way to GRR. The key difference is that OSQuery provides an SQL query to perform endpoint analysis. For more information, you can read the documentation at the following links: • https://osquery.io/ • https://osquery.readthedocs.io/en/stable/deployment/anomaly-detection/

Module 11: Security Monitoring

Logging policy

This monitoring can be done by various kinds of security tools, such as host IDS, network IDS/IPS, antivirus software, firewalls, and also security information and event management (SIEM).

The NIST SP 800-92 Guide to Computer Security Log Management suggests that the log collection configuration should be based on the security impact to the systems

Examples of logging configuration settings by NIST SP 800-92:

Category	**Low impact**	**Moderate impact**	**High impact**
How long to retain log data (Keep in mind that the cybersecurity law may also have explicitly requested the log	One to two weeks	One to	Three

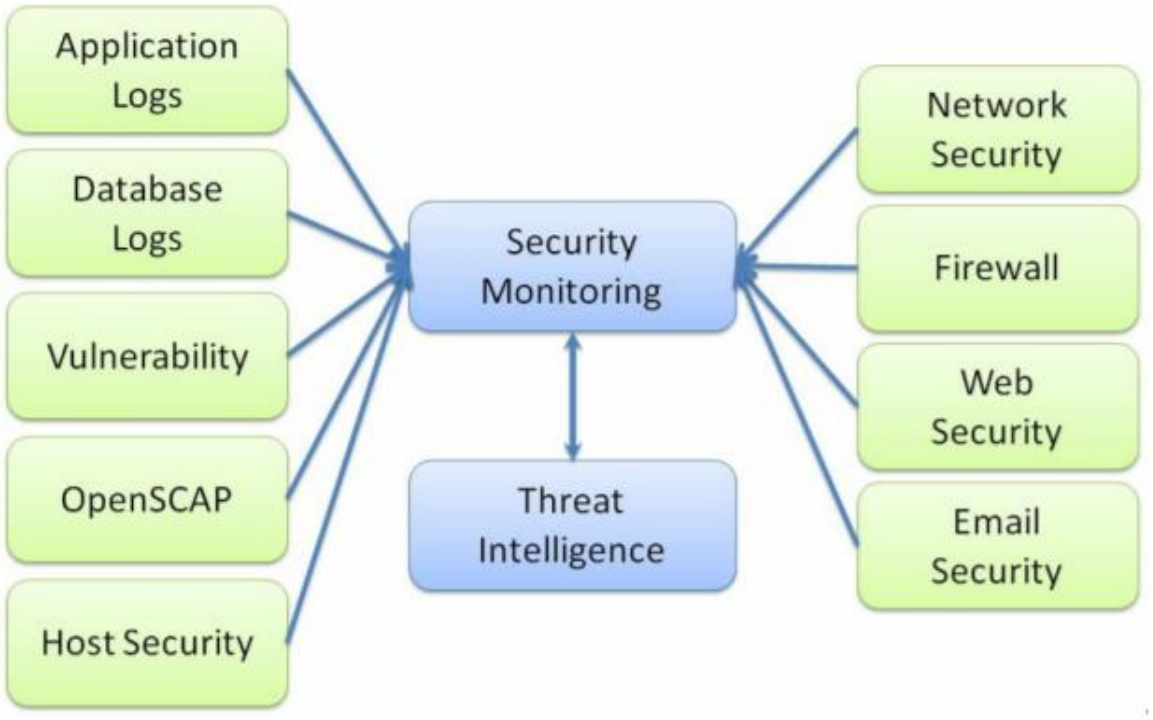
retention period. The number here is just an example.)			
How often to rotate logs	Optional (if performed, at least every week, or for every 25 MB)	Every six to 24 hours, or every 2 to 5 MB	Every 15 to 60 minutes or every 0.5 to 1.0 MB
How frequently the organization requires the system to transfer log data to the log management infrastructure, if it has this policy	Every 3 to 24 hours	Every 15 to 60 minutes	At least every five minutes
How often log data needs to be analyzed locally (through automated or manual means)	Every 1 to 7 days	Every 12 to 24 hours	At least six times a day
Whether log file integrity checking needs to be performed for rotated logs	Optional	Yes	Yes
Whether rotated logs need to be encrypted	Optional	Optional	Yes
Whether log data transfers to the log management infrastructure need to be encrypted or performed on a separate logging network	Optional	Yes, if feasible	Yes

Security monitoring framework

Building a complete security monitoring framework involves incorporating the following key components:

- **Log collector:** This is responsible for collecting and forwarding all the logs to the security monitoring team for further analysis. In the production environment, the concern of the log collection is the performance impact of the host and the number of logs needed to be forwarded. Syslog is the most common way to send the logs to security monitoring management
- **Security monitoring (SIEM):** This gives the security administrator a visualized security overview of the whole environment. An ideal SIEM can even do automated security correlation analysis based on predefined rules to identify abnormalities and potential risks.
- **Threat intelligence:** Threat intelligence is used to correlate the collected in-house security logs with external threat information, such as blacklisted IPs, Tor exit nodes, known malicious domains, user agents, file hashes, and the indicators of compromise (IOC).
- **Threat intelligence feeds:** These form the threat database that includes known current threat information provided by cybersecurity communities, security vendors, or customer submissions. An organization may use the external threat intelligence feeds to correct internal security events in order to identify whether there are any suspicious activities, such as internal hosts connected to a known cybercrime IP.

The following diagram shows the typical scope of security monitoring:



Source of information

The various log sources will help you to provide security events in different respects. Here are some of the general recommendations of the security monitoring focuses:

1-Application logs

These are the operational and error logs generated by the application. If the application is a web service, the logs may be included in Apache or nginx logs:

- Monitor the user activities, especially those activities that involve access to sensitive data
- Monitor the major changes of user profiles, such as login IPs, abnormal endpoint devices, non-browser connection clients, and concurrent connections from different IP sources
- Monitor the activities of administration and service accounts
- Monitor login failures and web errors, such as 401, 404, and 501

2-Host security, database logs

These mainly rely on the host-based IDS/IPS detection logs, OS, and database logs:

- Successful and failed authentication of users
- Administrative access and changes
- Unauthorized login failure
- Major configuration file changes, such as mysql.cnf

- Database accounts added
- Massive data transmission to specific hosts

3-Vulnerability

The OpenVAS or NMAP scanning results of CVE vulnerabilities

Insecure communication ports or protocols, such as Telnet, SSH v1, SSL, and FTP

4-OpenSCAP

The adoption of OpenSCAP scanning tools can help you to identify the insecure configuration of the applications, OS, database, and web services

5-Network security, firewalls

Rely on the network IDS/IPS detection logs, and also the logs from the load balancer, switches, and routers. For the updated firewall rules for IPtables, Snort, and Suricata, refer to the EmergingThreats website.

6-Web security

Rely on the web application firewall detection logs:

- Client IP is from a blacklisted IP
- User-agent associated with suspicious clients
- Too many errors in the weblogs, such as 401, 404, and 500
- Refer to the OWASP ModSecurity CRS which includes the web application firewall ruleset.

7-Email Security

Reply to the email security scanning and detection logs:

- Unusual mail receivers or senders
- Malicious Email Attachment
- Malicious URL in message body

Threat intelligence toolset

The whole threat intelligence process normally includes the following key components:

- The log collector: This is used to collect the internal system, applications, and security logs
- SIEM/visualization: This is used to visualize the security posture in one dashboard
- Threat intelligence platform: This is used to correlate the internal and external threat information
- Threat intelligence feeds: This is the external threat database, such as the blacklist IP, malicious hash, suspicious domain, and so on

Here are some of the open source tools that will help you to build the whole threat intelligence solution:

Category	Open source security tools
Log collector/sensor	Syslog-NG: Syslog-ng is an enhanced log daemon which can handle not only standard syslog message but also unstructured data. **Rsyslog:** Rsyslog stands for a rocket-fast system for log processing. **FileBeat:** Filebeat provides a backpressuresensitive protocol that controls the flow of sending data to Logstash or Elasticsearch **LogStash:** Logstash is a data processing pipeline that collects the data, transforms it, and then sends it to Elasticsearch.
SIEM/visualization	**Kibana:** Kibana provides the visualization of the Elasticsearch data. ElasticSearch: Search, index and analyze the data in real time. **AlienValut OSSIM:** It's an open source SIEM (Security Information and Event Management) solution provided by AlienValut. **Grafana:** It provides a quick solution for log query and visualization regardless of the data store. **GrayLog:** It's an open soure solution for enterprise log management.
Threat intelligence platforms	**MISP - open source threat intelligence platform:** The MISP is the threat sharing platform which can search and correlate IoC (Indicators of Compromise), threat intelligence feeds and vulnerability information.
Threat intelligence feeds	External threat feeds for blacklisted IP list and firewall rules suggestions: • https://rules.emergingthreats.net/fwrules/ • https://www.spamhaus.org/drop/ • https://rules.emergingthreats.net/fwrules/emerging-B lock-IPs.txt • https://check.torproject.org/exit-addresses http://iplists.firehol.org/

Security scanning toolset

Here are some open source tools that can perform security monitoring, scanning, and detection. Although your organization may have some commercial security solutions in place, these open source security detection rules can be a good reference when optimizing the existing security detection, such as the IDS/IPS, firewall, and web security.

You may find the following rules helpful to update or improve your existing firewall rules:

- **Wazuh host IDS rules:** Host-based intrusion defense rules.
- **OSSEC host IDS rules:** Host-based intrusion defense rules.
- **ModSecurity WAF rules:** Web Application Firewall rules.
- **Suricata network IDS/IPS rules:** Network-based intrusion prevention firewall rules.
- **Snort network IDS/IPS rules:** Network-based intrusion prevention firewall rules.

The table lists the security monitoring tools in each category.

Category	Open source security monitoring tools

All-in-one security scanning (host, network, visualization)	Security Onion: https://github.com/Security-Onion-Sol utions This includes several open source security tools, such as Elasticsearch, Logstash, Kibana, Snort, Suricata, Bro, OSSEC, Sguil, Squert, and NetworkMiner.
All-in-one hostbased IDS, secure configuration, and visualization	The Wazuh integrates the OSSEC (a host-based IDS), OpenSCAP (secure configuration scanner), and Elastic Stack (threat visualization).
Secure configuration	he OpenSCAP defines the secure configuration for OS, Web, database, and application.
Vulnerability	The OpenVAS and OWASP dependency are two of popular open source vulnerability scanners.
Antivirus	The CalmAV is the open source antivirus for Windows. The LMD (Linux Malware Detect) is the Linux version open source antivirus.
Host IDS/IPS	The OSSEC and Samhain are two of open source host IDS/IPS solutions to be considered.
Web application firewall (WAF)	The ModSecurity which is one of OWASP open source project is a light-weight web application firewall.
Network IDS/IPS	Snort and Suricata are two of the popular open source network IDS/IPS solutions. These two solutions also provide frequently updated rules

Malware behavior matching – YARA

YARA (https://virustotal.github.io/yara/) is a pattern-matching Swiss army knife for malware detection.

For example, say that one host identifies suspicious webshell activities, but the antivirus software does not detect any suspicious activities. The security administrator can use the YARA detector with predefined YARA rules to scan all the files on the host or to scan the collected logs. Here is one example of a YARA rule to detect the web shell:

```
rule  php_webshell : webshell
{
    meta:
        description = “This is a sample of a PHP webshell detection rule.”
    strings:
         $x1 = “eval(\\\x65\\x76\\x6C”
         $x2 = “Dim wshell, intReturn, strPresult”  fullword   ascii
    condition:
         filesize < 15KB and all of them
}
```

The YARA rules define two characteristics of a web shell. When the YARA rules are scanned with any binary files, and if the files match the conditions where the file size is less than 15 KB and the criteria stipulated under x1 and x2 are also met, then the YARA scanner will identify a match.

The YARA scanner can be executed as a standalone command-line tool or as a Python plugin. Refer to the YARA introductory guide Compiling and Installing YARA to get your YARA scanner on Windows, Linux, and macOS. You can find the guide at https://yara.readthedocs.io/.

The latest YARA rules—as well as the signatures and detection of malware, malicious emails, webshells, packers, documents, exploit kits, CVEs, and cryptography—can be found at the following links:

- https://github.com/Yara-Rules/rules
- https://github.com/Neo23x0/signature-base
- https://github.com/InQuest/awesome-yara

Module 12: Security Assessment for New Releases

The following table shows an example of the relationship between the application releases and the security assessment scope:

Application release objective	**Security assessment scope**
New or major application release	Full assessment
Third-party component update	Assessment based on the third party and the integration interfaces
Patch releases	Targeted assessment based on patch scope
Emergency releases	The security testing scope is limited to ensure that there are no major security issues

he following table shows an example of the security assessment activities' execution by the development, security, and DevOps teams:

Security review stage	**Example key security practices**	**Executed by**
Self assessment	• Review the OWASP ASVS checklist • Review the OWASP Top 10 checklist • Execute the defined automated security tools, such as ZAP, NMAP, and SQLmap	Product development team

	• Execute the defined automated security tools, such as ZAP, NMAP, and SQLmap	
Pre-release	• Submit the self-assessment testing results and the prerelease package to the security team • The security team focuses on the assessment with the highest risk modules • The security team performs the acceptance security testing, which includes not only the packages, but also the secure configurations of the whole system, such as Linux, MySQL, and NginX • Manual and automated application and network security testing will be performed by the security review team, and you will receive your review results (see the following results section for more details)	Security team
Production	Perform regular security scans for the following: • Known CVEs of software components • Secure configurations • Network communications, such as ports and insecure protocols • OWASP Top 10 security issues	Operation and security team

Security checklist and tools

the security testing approaches, and the suggested security testing tools:

Security category	**Security testing approaches**	**Suggested security testing tools**
Hidden communication ports or channels	• Ensure that there are no hidden communication ports or backdoors • Ensure that there are no hidden hardcoded secrets, passwords, or hard keys • Check for unnecessary system maintenance tools • Perform a source code review for networking communication, such as Java-related API connect(), getPort(), getLocalPort(), Socket(), bind(), accept(), ServerSocket() • Listening to 0.0.0.0 is forbidden	NMAP Graudit TruffleHog Snallygaster Hping masscan
Privacy information	• Search for the plaintext password and key in the source code • Search for the personal information for the GDPR compliance	TruffleHog Blueflower YARA PrivacyScore

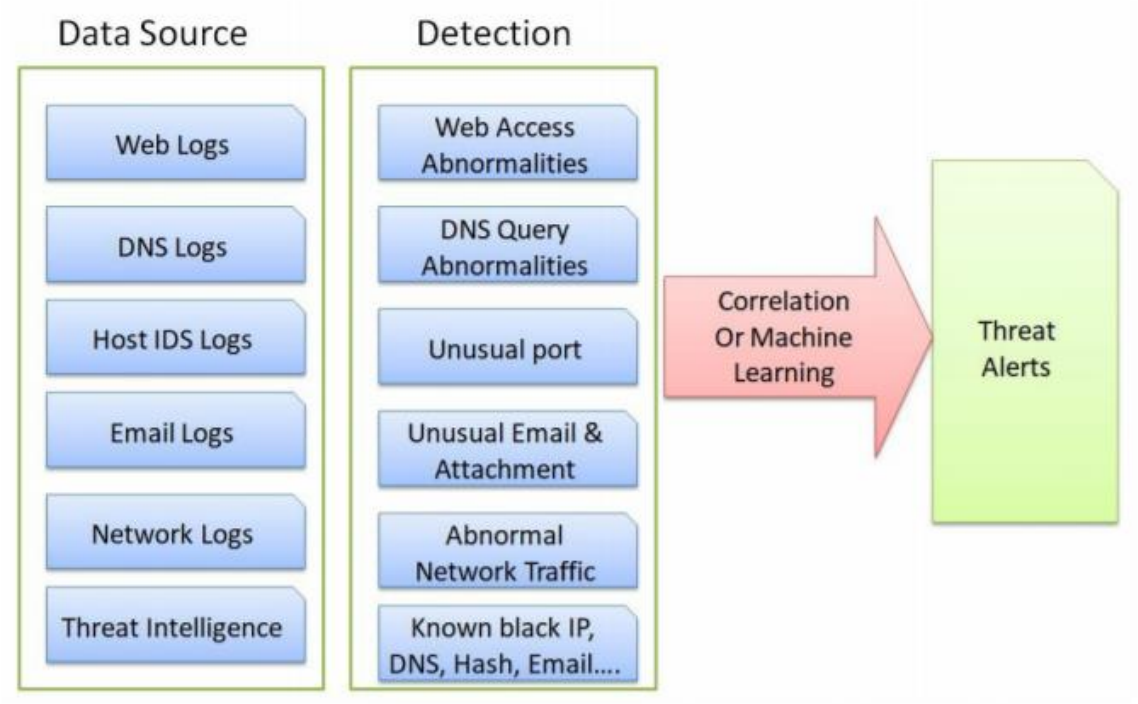
	• The personal information can be modified and removed by the end user • The personal information can be removed within a defined period	Snallygaster
Secure communication	• SSH v2 instead of Telnet • SFTP instead of FTP • TLS 1.2 instead of SSL TLS 1.1	NMAP WireShark SSLyze SSL/TLS tester
Third-party components	• CVE check • Known vulnerabilities check • Hidden malicious code or secrets	OWASP Dependency check LMD (Linux Malware Detection) OpenVAS NMAP CVEChecker
Cryptography	• Ensure that there is no weak encryption algorithm • Ensure that there are no secret files on the public web interfaces	Graudit SSLyze Snallygaster
Audit logging	Ensure that the operation and security teams can log the following scenarios: • Non-query operations, including success and failure actions • Non-query scheduled tasks • API access or tool connections that execute administration tasks	GREP
DoS attacks	The testing of the DoS is to ensure if the application failure is as expected. The DoS scenario may cover the following: • TCP Sync flooding • HTTP Slow • HTTP Post Flooding • NTP DoS • SSL DoS	Pwnloris Slowloris Synflood Thc-sll-DoS Wreckuests ntpDoS
Web security	To develop a policy concerning web security, you can refer to the OWASP Testing Guide and OWASP Top 10: • Injection • Authentication • Data exposure • XXE Broken • access control Security • misconfiguration • XSS • Insecure deserialization	OWASP ZAP BurpSuite Arachni Scanner SQLMap

	• Known vulnerabilities • Insufficient logging and monitoring	
Secure configuration	Ensure that the configurations of applications, web services, databases, and the OS are secure. The secure configurations are based on the CIS security benchmark and OpenSCAP.	OpenSCAP Docker Bench Security Clair
Fuzz testing	The purpose of fuzz testing is to generate dynamic testing data as input to check whether the application will fail unexpectedly.	API Fuzzer Radamsa American Fuzzy lop FuzzDB Wfuzz
Mobile app security	Refer to the OWASP Mobile App	Mobile Security Framework
Top common issue	Draw up a list of the most common security issues based on projected historical data.	CWE/SANS Top 25 Most Dangerous Software Errors
Security compliance	Security compliance that is based on business needs may also be included, such as GDPR or PCI DSS.	Refer to the specific security compliance requirements

Module 13: Threat Inspection and Intelligence

Unknown threat detection

The following diagram shows the concept of correlation or machine learning with different data sources:



The following are some typical abnormal network traffic examples:

Abnormal network traffic	**Potential threats**
Port/host scan	The port or host scan behaviors mean one of the hosts may have been infected by a malware program, and the malware program is looking for vulnerabilities, other services, or hosts on the network.
A high number of outbound DNS requests from the same host	This is a symptom of Command and Control (C&C) malware, establishing communication between the infected host and the C&C server using the DNS protocol.
A high number of outbound HTTP requests from the same host	This is a symptom of C&C, establishing communication between the infected host and the C&C server using the HTTP protocol.
Periodical outbound traffic with samesized requests or during the same period of time every day	This is a symptom of C&C malware, establishing communication between the infected host and the C&C server.
Outbound traffic to an external web or DNS listed as a known threat by threat intelligence feeds	The user may be tricked through social engineering to connect to an external known threat web or the C&C connection is successfully established.

To visualize the network threat status, there are two recommended open source tools: Malcom and Maltrail (Malicious Traffic detection system). Malcom can present a host communication relationship diagram. It helps us to understand whether there are any internal hosts connected to an external suspicious C&C server or known bad sites

https://github.com/tomchop/malcom#what-is-malcom

Indicators of compromises

An analysis of hosts for suspicious behaviors also poses a significant challenge due to the availability of logs. For example, dynamic runtime information may not be logged in files and the original process used to drop a suspicious file may not be recorded. Therefore, it is always recommended to install a host IDS/IPS such as OSSEC (Open Source HIDS SEcurity) or host antivirus software as the first line of defense against malware. Once the host IDS/IPS or antivirus software is in place, threat intelligence and big data analysis are supplementary, helping us to understand the overall host's security posture and any known **Indicators of Compromises (IoCs)** in existing host environments.

Based on the level of severity, the following are key behaviors that may indicate a compromised host:

Abnormal host behaviors	**Potential threats**
Multiple compromised hosts' data communication to external hosts	The compromised hosts are sending data to external C&C servers.
The host connects to an external known APT IP address or URL and/or downloads a known malicious file	The host shows an indication of compromise from APT or a malware attack
Several unsuccessful login attempts	One of the internal compromised hosts is trying to log in in order to access critical information.
An email message that includes a dangerous URL or malicious file	Attackers may use social engineering to send emails for target attacks. Include the email senders in the watch list.
Rare and unusual filenames in process/service/program start	The malware installs itself to start up so as to continue to act even after rebooting. One of the common ways in which malware can achieve persistence is as follows: In the case of Windows, using AutoRuns to check whether the host is compromised with suspicious malware is recommended. https://docs.microsoft.com/en-us/sysinternals/do
Unusual event and audit logs alert	The following system event or audit logs may need further analysis: • Account lockouts • Users added to the privileged group • A failed user account login

	• Application error(s) • Windows error reporting • BSOD • The event log was cleared • The audit log was cleared • A firewall rule change

An analysis of web access is also very critical, since the majority of internet connections are based on the HTTP protocol. There are two major scenarios regarding web access. One is the internal hosts that connect to external websites, and the other is the hosted web services connected by internal or external hosts. The following table lists some of the common techniques and tools for web access analysis:

Web access analysis	**Detection techniques**
External source client IP	The source of IP address analysis can help to identify the following: • A known bad IP or TOR exit node • Abnormal geolocation changes • Concurrent connections from different geolocations The MaxMind GeoIP2 database can be used to translate the IP address to a geolocation: https://dev.maxmind.com/geoip/geoip2/geolite2/#Downloads
Client fingerprint (OS, browser, user agent, devices, and so on)	The client fingerprint can be used to identify whether there are any unusual client or non-browser connections. The open source ClientJS is a pure JavaScript that can be used to collect client fingerprint information. The JA3 provided by Salesforce uses SSL/TLS connection profiling to identify malicious clients. ClientJS: https://clientjs.org/ JA3: https://github.com/salesforce/ja3
Web site reputation	When there is an outbound connection to an external website, we may check the threat reputation of that target website. This can be done by means of the web application firewall, or web gateway security solutions https://www.virustotal.com/
Random Domain Name by Domain Generation Algorithms (DGAs)	The domain name of the C&C server can be generated by DGAs. The key characteristics of the DGA domain are high entropy, high consonant count, and long length of a domain name. Based on these indicators, we may analyze whether the domain name is generated by DGAs and could be a potential C&C server. DGA Detector: https://github.com/exp0se/dga_detector/ In addition, in order to reduce false positives, we may also use Alexa's top one million sites as a website whitelist. Refer to https://s3.amazonaws.com/alexa-static/top-1m.csv.zip.

Suspicious file downloads	Cuckoo sandbox suspicious file analysis: https://cuckoosandbox.org/
DNS query	In the case of DNS query analysis, the following are the key indicators of compromises: • DNS query to unauthorized DNS servers. • Unmatched DNS replies can be an indicator of DNS spoofing. • Clients connect to multiple DNS servers. • A long DNS query, such as one in excess of 150 characters, which is an indicator of DNS tunneling. • A domain name with high entropy. This is an indicator of DNS tunneling or a C&C server.

Security analysis using big data frameworks

After discussing some of the common techniques for detecting unknown potential threats, we are going to introduce some open source frameworks to do security analysis with threat intelligence and big data technologies. You may consider applying these open source solutions as a basis if you are planning to build a security log analysis framework that can do the following:

- Machine learning and correlation with the IoCs
- Analysis involving external threat intelligence feeds
- Data enrichment such as GeoIP information
- Visualization and querying of the relationships of IoCs

Project	Key features
TheHive project	TheHive provides threat incident response case management that allows security analysts to flag IOCs. The Cortex can perform analysis with threat intelligence services such as VirtusTotal, MaxMind, and DomainTools. There are over 80 threat intelligence services supported. The Hippocampe provides a query interface through a REST API or a Web UI: https://thehive-project.org/
MISP	This is mainly a threat intelligence platform to share IoCs and indicators of malware. The correlation engine helps to identify the relationships between attributes and indicators of malware: https://www.misp-project.org/documentation/ The MISP provides over 40 threat intelligence feeds. Refer to https://www.misp-project.org/feeds/

Apache Metron	Apache Metron is a SIEM (threat intelligence, security data parsers, alerts, and a dashboard) and also a security analysis (anomaly detection and machine learning) framework based on Hadoop's big data framework: https://metron.apache.org/ Typical technology components used to build a big data framework include the following: • Apache Flume • Apache Kafka • Apache Storm or Spark Apache Hadoop • Apache Hive • Apache Hbase • Elasticsearch • MySQL

These open source solutions can work together with one another. For example, TheHive can be used as a security operation center to manage security incident cases with IoC information, and integrate TheHive with MISP to query external threat intelligence feeds. Moreover, Metron can perform log data enrichment and analysis with machine learning to identify abnormalities.

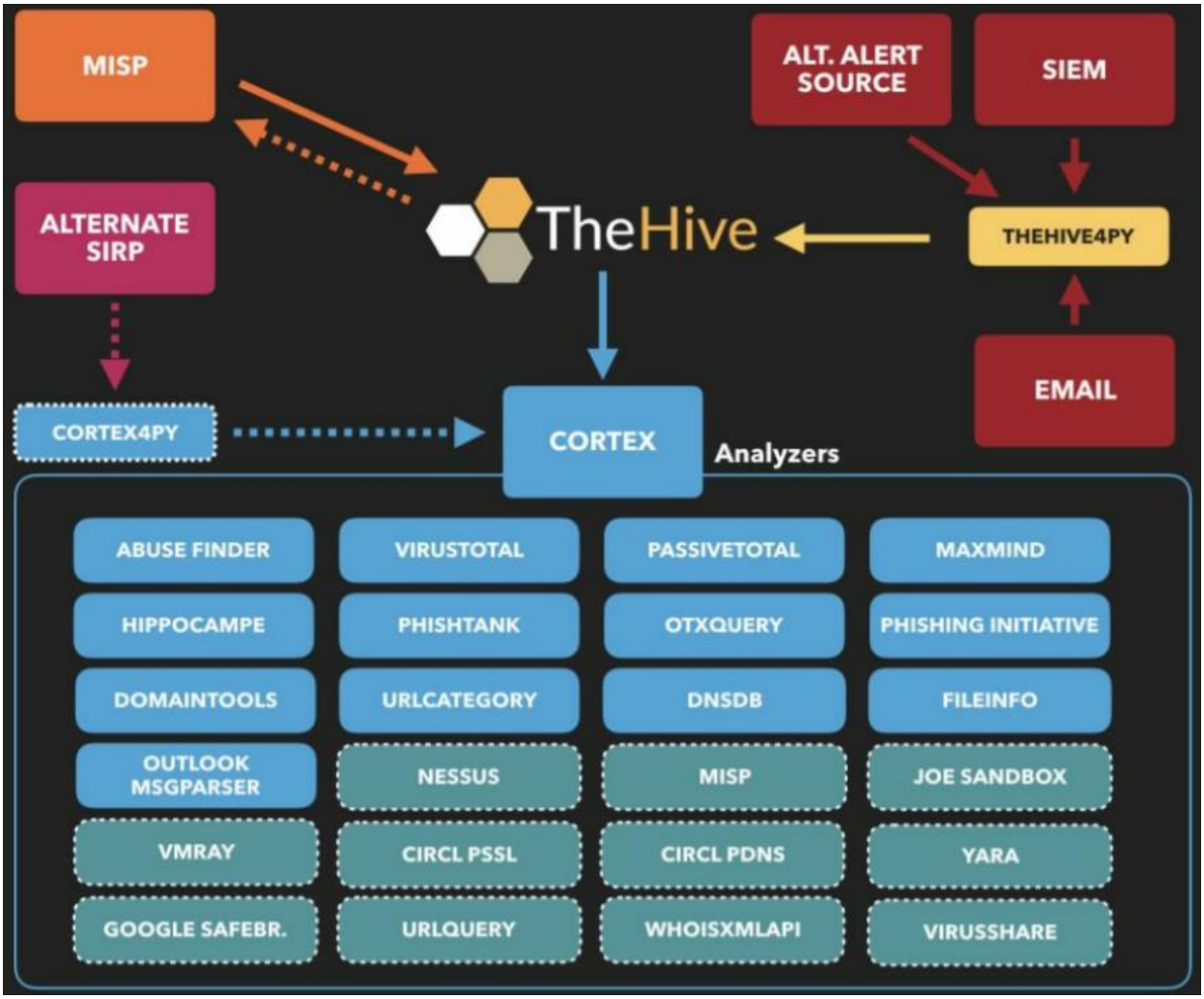
In addition, there are also some open source analysis frameworks based on the **Elasticsearch, Logstash, Kibana (ELK)**. Refer to the following list:

- Response Operation Collection Kit (ROCK) NSM: http://rocknsm.io/
- A Hunting Elasticsearch, Logstash, Kibana (ELK) with advanced analytical capabilities: https://github.com/Cyb3rWard0g/HELK
- Cyber Analytics Platform and Examination System (CAPES): http:// capesstack.io/

TheHive

TheHive is a security incident response platform that integrates **Malware Information Sharing Platform (MISP)**. The Cortex can help to analyze observables using external threat analysis services such as VirusTotal, DomainTools, and MaxMind. The Hippocampe provides the REST API or Web UI to enable users to carry out analysis reports and perform queries.

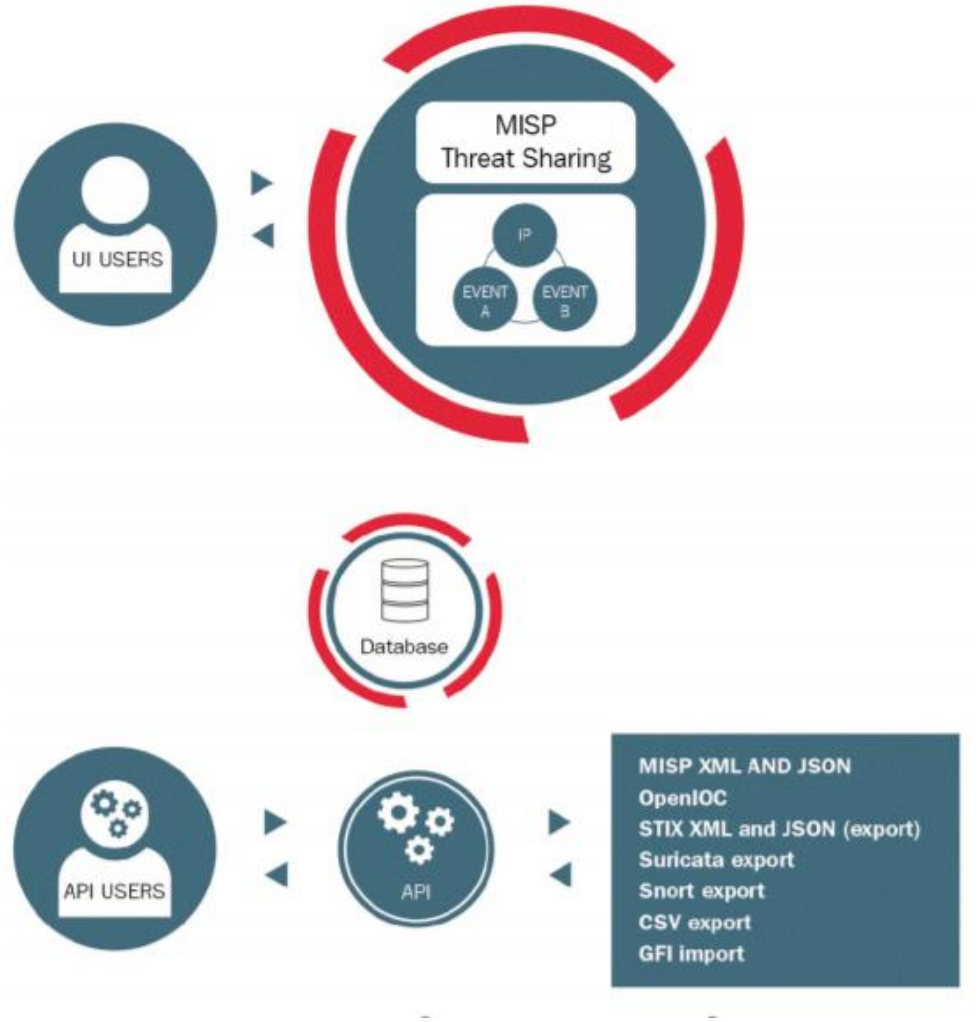
The following diagram shows the collaboration between TheHive, Cortex, SIEM, and also MISP:



MISP – an Open Source Threat Intelligence Platform

MISP is a Threat Intelligence Platform that can carry out correlations with threat attributes, IOCs, and indicators. MISP can also generate Snort/Suricata IDS rules, STIX, and OpenIOC detection rules based on the IOCs observed.

The following diagram refers to MISP (Malware Information Sharing Platform):



In addition to MISP, you may also refer to the open source **Your Everyday Threat Intelligence (YETI)** platform solution, which also provides a similar threat intelligence platform. Refer to https://yeti-platform.github.io/.

Apache Metron

Apache Metron is a cybersecurity application framework that can perform big data analysis to identify anomalies. The framework provides the following key characteristics:

- The processing, enrichment, and labeling of the data source for security analysis, search, and query.
- Anomaly detection using machine learning algorithms
- SIEM-like capabilities (alerting, threat intelligence framework, agents to ingest data sources)
- A pluggable framework for various kinds of data sources and that can add parsers for new data sources

Resources

1. Hands-On Security in DevOps by Tony Hsu