



Akademia
Techniczno-Humanistyczna
w Bielsku-Białej

PROCES TWORZENIA BEZPIECZNEGO OPROGRAMOWANIA

Cyberbezpieczeństwo

1

dr inż. Ruslan Shevchuk

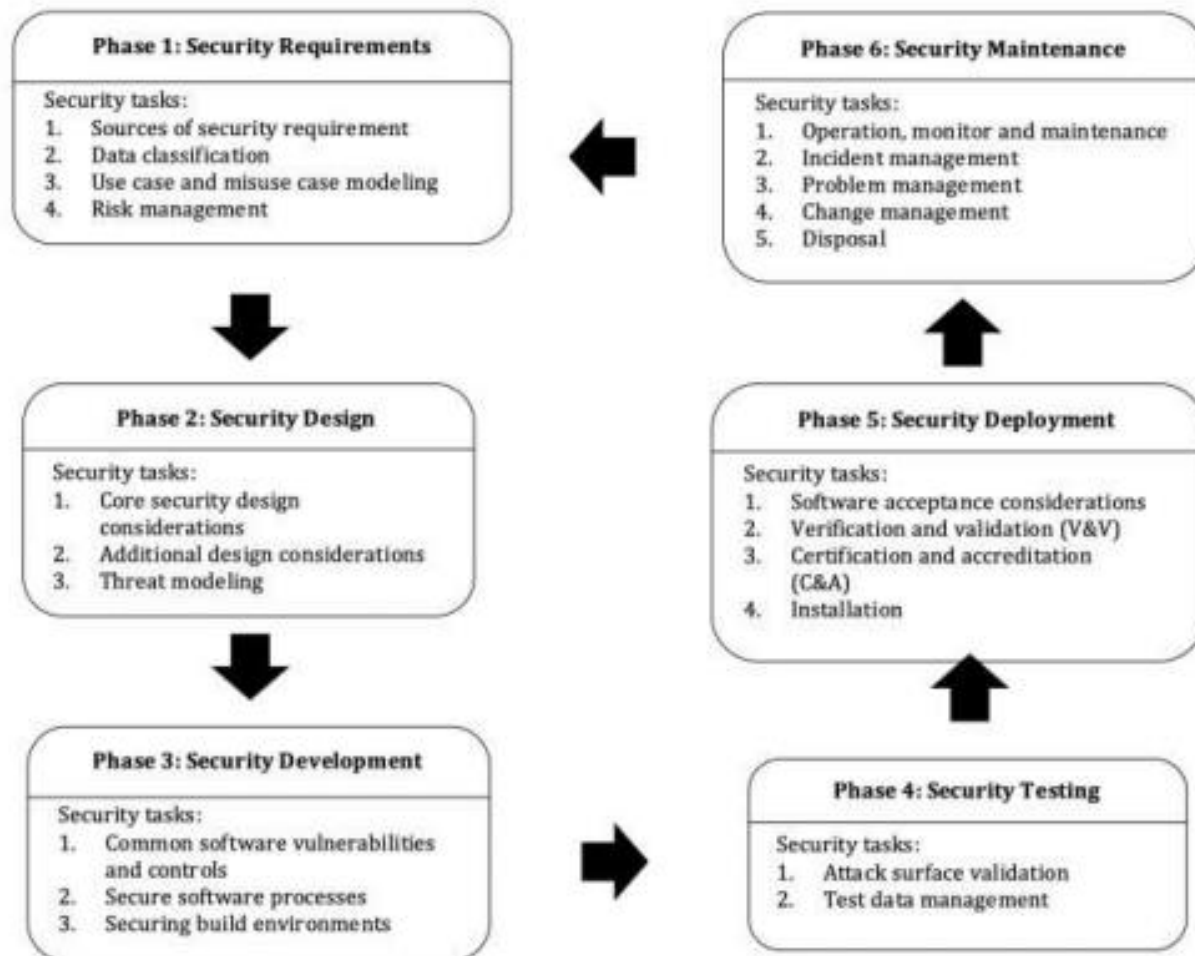
PLAN WYKŁADU:

1. Faza wymagań bezpieczeństwa
2. Faza projektowania bezpiecznego
3. Faza bezpiecznego kodowania
4. Faza testowania bezpieczeństwa
5. Faza wdrażanie bezpieczeństwa
6. Faza utrzymania bezpieczeństwa

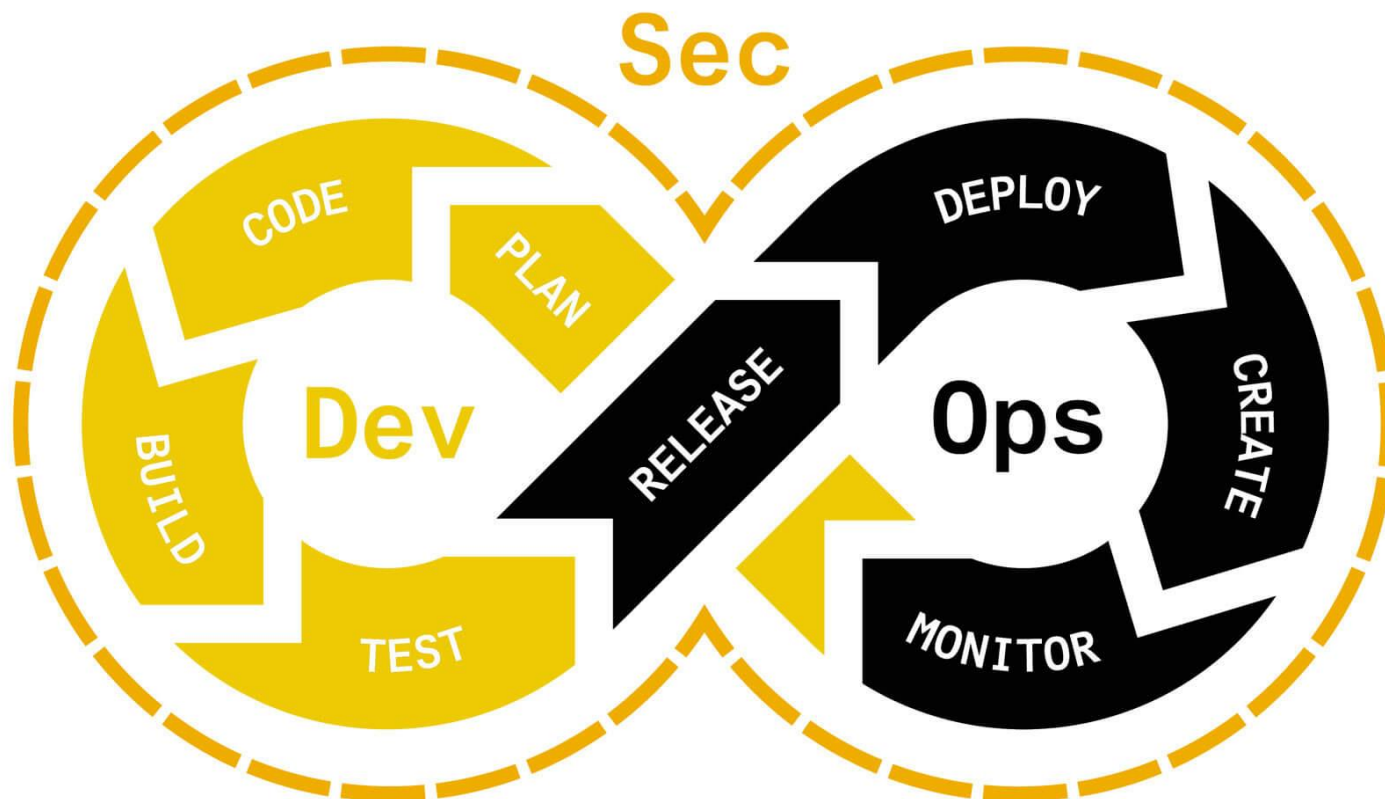


SECURE SOFTWARE DEVELOPMENT LIFECYCLE

Secure Software Development Lifecycle to podejście mentalne i techniczne uwzględniające kwestie bezpieczeństwa na każdym etapie wytwarzania oprogramowania. Znacząco zwiększa efektywność przy niższych kosztach wynikających z wcześniej wykrytych błędów.



DEVSECOPS = DEVOPS + SEC



Celem podejścia DevSecOps jest zakotwiczenie bezpieczeństwa IT w dzisiejszych typowo szybkich cyklach rozwojowych.





1. FAZA WYMAGAŃ BEZPIECZEŃSTWA

5

PHASE 1: SECURITY REQUIREMENTS

CELE

- Zidentyfikować i zdefiniować odpowiednie wymagania bezpieczeństwa
- Przygotować bezpieczeństwo tożsamości do tworzenia oprogramowania.

ZADANIA BEZPIECZEŃSTWA

- 1. Identyfikacja źródeł wymagań bezpieczeństwa** (1.1. Identify core security requirements, 1.2. Identify general requirements, 1.3. Identify operational requirements, 1.4. Identify other requirements)
- 2. Klasyfikacja danych** (2.1. Types of data, 2.2. Labeling the data, 2.3. Data ownership and roles, 2.4. Data lifecycle management (DLM), 2.5. Privacy requirements)
- 3. Modelowanie przypadków użycia** (3.1 Analyze the use case scenarios, 3.2 Analyze the misuse case scenarios, 3.3 Creating attack model, 3.4 Select mitigation control)
- 4. Zarządzanie ryzykiem** (4.1 Risk assessment, 4.2 Risk mitigation, 4.3 Evaluation and assessment)



1.1. IDENTIFY CORE SECURITY REQUIREMENTS

a) Wymagania dotyczące poufności.

Mechanizmy zachowania poufności: **Kryptografia** (szyfrowanie, haszowanie, steganografia i cyfrowe znaki wodne) i **Maskowanie** (oryginał informacji są oznaczone gwiazdką lub X)

b) Wymagania dotyczące uczciwości.

Mechanizmy zachowania poufności: kontrola dostępu użytkowników, kontrola wersji oprogramowania, dodawanie sum kontrolnych, kryptograficznych sum kontrolnych

c) Wymagania dotyczące dostępności

d) Wymagania dotyczące uwierzytelniania

Formy uwierzytelniania: Anonymous authentication, Basic authentication, Digest authentication, Integrated authentication, Client certificate-based authentication, Forms authentication, Token-based authentication, Smart card-based authentication, Biometric authentication

e) Wymagania dotyczące autoryzacji

Modele kontroli dostępu: Discretionary Access Control (DAC), Non-Discretionary Access Control (NDAC), Mandatory Access Control (MAC), Role-Based Access Control (RBAC), Resource-Based Access Control

EXAMPLE OF SECURITY REQUIREMENTS (1)

Table B-1 Core Security Requirements (Security Goals) and Associated Security Measures

Core security requirements	Associated security measures
Confidentiality	Access control; Physical protection; Security policy
Integrity	Access control; Non-repudiation; Physical protection; Attack detection
Availability	System recovery; Physical protection; Attack detection
Accountability	Non-repudiation; Attack detection

Table B-2 Security Measures and Associated Security Mechanisms

Security measures	Associated security mechanisms
Access control	Biometrics; Certificates; Multilevel security; Passwords and keys; Reference monitor; Registration; Time limits; User permissions; VPN
Security policy	Administrative privileges; Malware detection; Multilevel security; Reference monitor; Secure channels; Security session; Single access point; Time limits; User permissions; VPN
Non-repudiation	Administrative privileges; Logging and auditing; Reference monitor
Physical protection	Access cards; Alarms; Equipment tagging; Locks; Off-site storage; Secured rooms; Security personnel
System recovery	Backup and restoration; Configuration management; Connection service agreement; Disaster recovery; Off-site storage; Redundancy
Attack detection	Administrative privileges; Alarms; Incident response; Intrusion detection systems; Logging and auditing; Malware detection; Reference monitor
Boundary protection	DMZ (Demilitarized Zone); Firewalls; Proxies; Single access point; VPN



EXAMPLE OF SECURITY REQUIREMENTS (2)

Table B-3 Security Requirements with Priority Level

Security type	Requirement description	Comments	Priority
Authentication	The system should have authentication measures at all the entry points, front panel, or inbound network connection.	To avoid unauthorized access	1
	The system should support Windows domain authentication, and when authenticate to AD (Active Directory), should support NTLMv2, as well as Kerberos protocol, and should avoid transmitting username/password on the wire when authentication to the AD.	Currently many systems use Single Sign-On (SSO) using Kerberos and Windows domain.	2
	The system should support Smartcard, USB token (two factors) authentication.	Improving the security using smartcard technologies and	1
Availability	The backup system should store the recover sources in a network system.	To help in case of failure or intruder action	1
	The system should do mirroring to allow data and software to be available in physically separated locals (separate site when the application is on the Web)	Help minimize the risk of one single point of failure.	3
	The system should apply high availability solution such as clustering.	Help minimizing the impact of potential system failures.	3
Integrity	The system should ensure all data provided by software has consistency (either create a new and valid state of data, or, return all data to its state before a transaction was started).	To avoid an authorized person or system alter data inadvertently or intentionally.	1
Auditing	The system should keep historical records (logging) of events and processes executed in or by an application.	Define more specific security loggings to allow recreating a clear picture of security events.	1
Non-Repudiation	The system should implement cryptographic methods such as generating digital signatures or digital fingerprinting.	Help the application and system avoiding repudiation.	1



1.2. IDENTIFY GENERAL REQUIREMENTS

- a) Wymagania dotyczące zarządzania sesjami
- b) Wymagania dotyczące zarządzania błędami i wyjątkami
- c) Wymagania dotyczące zarządzania parametrami konfiguracji

1.3. IDENTIFY OPERATIONAL REQUIREMENTS

- a) Wymagania dotyczące środowiska wdrażania
- b) Wymagania dotyczące archiwizacji
- c) Wymagania antypirackie

1.4. IDENTIFY OTHER REQUIREMENTS

- a) Wymagania czasowe
- b) Wymagania międzynarodowe
- c) Wymagania dotyczące zamówień



2.1. TYPES OF DATA

Klasyfikacja danych według typu (dane ustrukturyzowane, dane nieustrukturyzowane)

2.2. LABELING THE DATA

Oznaczenie danych to próba przypisania etykiet (poziomów wrażliwości) do danych informacyjnych w oparciu o potencjalny wpływ na poufność, integralność i dostępność (CIA) po ujawnieniu, zmianie lub zniszczeniu.

2.3. DATA OWNERSHIP AND ROLES

Decyzje dotyczące klasyfikacji danych, kto ma dostęp i jaki poziom dostępu itp. to są decyzje, które trzeba podjąć przez firmę/właściciela danych.

2.4. DATA LIFECYCLE MANAGEMENT

Podejście oparte na zasadach, obejmujące procedury i praktyki mające na celu ochronę danych przez cały cykl życia informacji: od momentu ich utworzenia do momentu usunięcia.

2.5. PRIVACY REQUIREMENTS (1)

Klasyfikacja danych może pomóc w identyfikacji danych, które będą wymagały zastosowania wymogów ochrony prywatności.

Najlepsze praktyki w zakresie prywatności danych, które należy uwzględnić w analizie wymagań oprogramowania:

- a) Jeśli dane nie potrzebujesz, nie odbieraj ich.
- b) Jeśli potrzebujesz dane zebrać tylko do przetwarzania, zbieraj ich dopiero po poinformowaniu użytkownika, że zbierasz jego dane, a on wyraził na to zgodę, ale ich nie przechowuj.
- c) Jeśli masz potrzebę zebrania dane w celu przetwarzania i przechowywania, zbieraj za zgodą użytkownika i przechowuj tylko przez wyraźny okres przechowywania zgodny z polityką organizacyjną i/lub wymogi regulacyjne.
- d) Jeśli masz potrzebę gromadzenia i przechowywania danych, nie archiwizuj ich, jeśli dane przeżyły swoją przydatność i nie ma wymogu ich przechowywania.



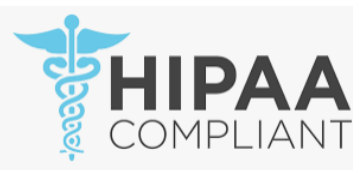
2.5. PRIVACY REQUIREMENTS (2)

Anonimizacja danych – czynność polegająca na przekształceniu danych osobowych w sposób uniemożliwiający przyporządkowanie poszczególnych informacji do określonej lub możliwej do zidentyfikowania osoby fizycznej albo, jeżeli przyporządkowanie takie wymagałoby niewspółmiernych kosztów, czasu lub działań

Dyspozycja - oprogramowanie jest podatne na ataki, dopóki nie zostanie ono bezpiecznie usunięte. Większość przepisów dotyczących prywatności wymaga wdrożenia zasad i procedur dotyczących ostatecznego dysponowania danych i informacji.

Modele bezpieczeństwa – formalna abstrakcja polityki bezpieczeństwa, która składa się z zestawu wymagań bezpieczeństwa, które są odporne na ataki, mogą tolerować ataki, którym nie można się oprzeć, i mogą szybko wyjść z niepożądanego stanu, jeśli zostaną naruszone.

Pseudonimizacja - przetwarzanie danych osobowych w taki sposób, aby nie było możliwe zidentyfikowanie, do kogo one należą, bez dostępu do innych informacji, przechowywanych bezpiecznie w innym miejscu.



3.1. ANALYZE THE USE CASE SCENARIOS

Diagramy przypadków użycia pozwalają na graficzne zaprezentowanie własności systemu tak, jak są one widziane po stronie użytkownika.

3.2. ANALYZE THE MISUSE CASE SCENARIOS

Nieprawidłowe scenariuszy to niezamierzone zachowanie oprogramowania, które nie jest pożądane przez właściciela oprogramowania w kontekście przypadku użycia.

3.3. CREATING ATTACK MODEL

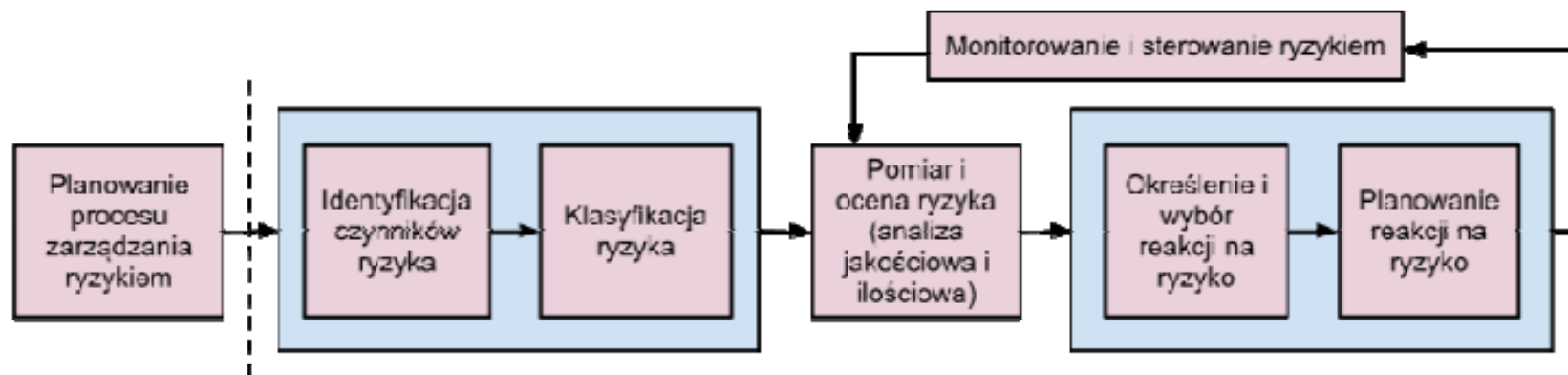
- a) Wybierzcie wzorce ataków odpowiednie dla oprogramowania. Zbudujcie przypadki nadużyć wokół tych wzorców ataków.
- b) Uwzględnicie każdego, kto może uzyskać dostęp do oprogramowania, ponieważ zagrożenia powinny obejmować wszystkie potencjalne źródła zagrożeń dla oprogramowania

3.4. SELECT MITIGATION CONTROL

Zaproponujcie kontrole na podstawie ataków zidentyfikowanych w poprzednim kroku. Podkreślicie tę kwestię, oferując kontrolę bezpieczeństwa.

4.1 RISK ASSESSMENT

Zarządzanie ryzykiem to takie podejmowanie decyzji, które pozwoli na uniknięcie zidentyfikowanych zagrożeń oraz potencjalnych szkód, które mogłyby by mieć miejsce w przypadku ich wystąpienia.



Model zarządzania ryzykiem

Klasyfikacja ryzyka wymaga zbadania jego źródła, w celu przyporządkowania go do jednej z pięciu płaszczyzn: technicznej, programowej, obsługowej, kosztowej oraz harmonogramowej.

4.2. RISK MITIGATION

Ograniczania ryzyka - ustalanie priorytetów, ocena i wdrażanie odpowiednich zasobów kontroli zmniejszających ryzyko zalecanych przez proces oceny ryzyka.



4.3. EVALUATION AND ASSESSMENT

Częsta zmiana ryzyka w czasie oraz brak pełnej informacji o możliwych skutkach ryzyka wymaga od kierownika regularnej aktualizacji planu zarządzania ryzykiem.

RISK ASSESSMENT: PART OF RAPORT

Potential loss	Business functions/ financial state	Compliance with laws	Reputation
1 – Low	Minor issues (the loss is up to \$5,000).	Minor violations (administrative responsibility, amount of penalty up to \$1,000).	Little, insignificant problem
3 – Middle	Financial problems (the amount of loss is up to \$50,000).	Breach of legislation (administrative responsibility, amount of penalty up to \$50,000).	Negative customers', partners', or investors' response
5 – High	Serious financial problems / unprofitability of a company (the amount of loss is up to \$500,000).	Serious breach of legislation (administrative responsibility – amount of penalty up to \$500,000).	Extremely negative customers', partners', or investors' response
7 – Very high	Severe financial problems (the amount of loss is more than \$500,000).	Severe breach of legislation (criminal responsibility, administrative responsibility – amount of penalty more than \$500,000).	Company existence is under threat

	Risk level	Level Description
Green	[1-9] – low	A minor risk that does not require any preventive actions. It does not influence company performance or its impact is very low.
Yellow	[15-25] – middle	An acceptable risk that has little impact on company performance. Preventive measures can be developed for risk mitigation, but are not obligatory.
Red	[27-63] – high	An unacceptable risk with serious negative impact. All necessary risk treatment measures must be undertaken.

Asset	Asset owner	Risk	Risk owner	Likelihood	Loss	Risk level	Treatment option	Control
Corporate laptop	Employee	Loss of data due to insecure information storage	Head of Corporate Security	5 (once a year)	5 (Serious financial problems)	25	Mitigate	Conduct training for laptop users
Exchange server	System administrator	Equipment failure due to the service end-of-life	Head of IT	1 (nearly impossible)	5 (Serious financial problems)	5	Accept	The risk is considered as acceptable due to the small probability of the event occurrence





2. FAZA PROJEKTOWANIA BEZPIECZNEGO

18

PHASE 2: SECURITY DESIGN

CELE

- Zaprojektować bezpieczną architekturę z uwzględnieniem elementów bezpieczeństwa
- Zabezpieczyć architekturę oprogramowania poprzez zrozumienie i analizę zagrożeń dla oprogramowania

ZADANIA BEZPIECZEŃSTWA

- podstawowe zagadnienia dotyczące projektowania bezpiecznego (1.1 Confidentiality design, 1.2 Integrity design, 1.3 Availability design, 1.4 Authentication design, 1.5 Authorization design, 1.6 Accountability design)
- dodatkowe zagadnienia dotyczące projektowania bezpiecznego (2.1 Programming languages, 2.2 Data type, format, range, and length, 2.3 Database security, 2.4 Interface design, 2.5 Interconnectivity)
- modelowanie zagrożeń (3.1 Decompose the software, 3.2 Determine and rank threats, 3.3 Determine countermeasures and mitigation)



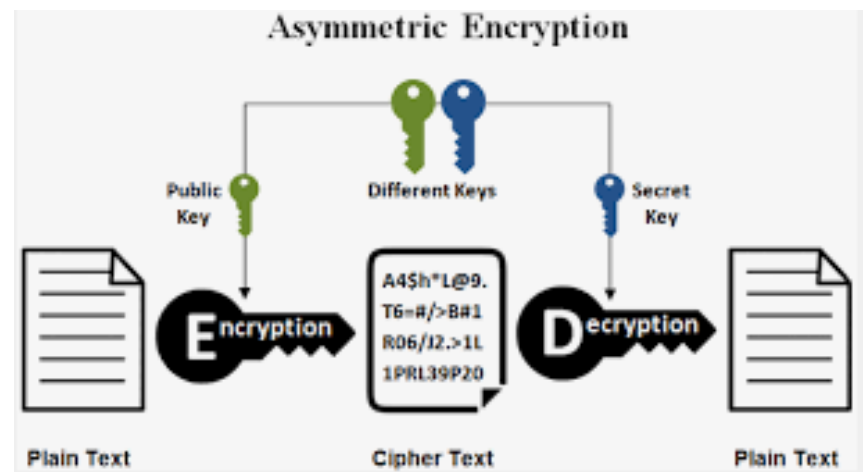
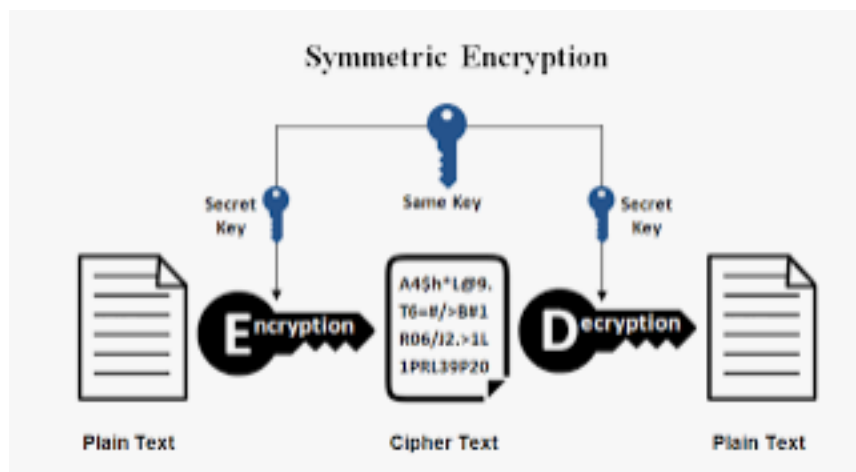
1.1 CONFIDENTIALITY DESIGN

Utajnianie informacji można zrealizować w dwojaki sposób przez: **maskowanie informacji** i **szyfrowanie informacji** (Algorytmy symetryczne, Algorytmy asymetryczne).

last_name	first_name	ssn	gender	state
Smith	Bob	123-45-6789	M	CA
Doe	Jane	098-76-5432	F	PA
King	Stephen	888-67-5309	M	WI
Savage	Randal;	135-24-6789	M	FL
Downer	Debbie	918-55-4680	F	NC

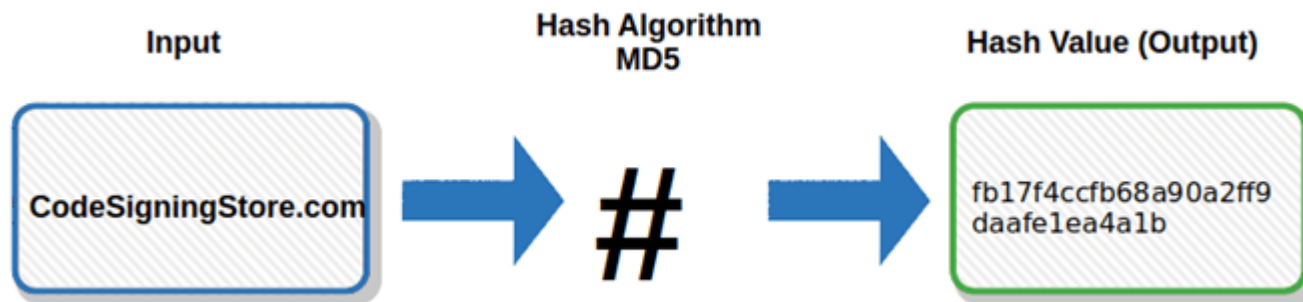


last_name	first_name	ssn	gender	state
Smith	Bob	xxx-xx-xxxx	M	CA
Doe	Jane	xxx-xx-xxxx	F	PA
King	Stephen	xxx-xx-xxxx	M	WI
Savage	Randy	xxx-xx-xxxx	M	FL
Downer	Debbie	xxx-xx-xxxx	F	NC



1.2 INTEGRITY DESIGN

Ochrona **integralności** zapobiega przypadkowemu zniekształceniu danych podczas odczytu, zapisu, transmisji lub magazynowania.



Hashing (Hash functions)

Primary Table

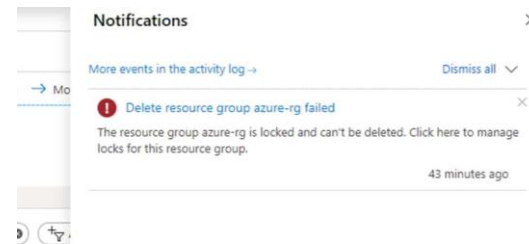
CompanyId	CompanyName
1	Apple
2	Samsung

Related Table

CompanyId	ProductId	ProductName
1	1	iPhone
15	2	Mustang

Associated Record ✓
Orphaned Record ✗

Referential Integrity

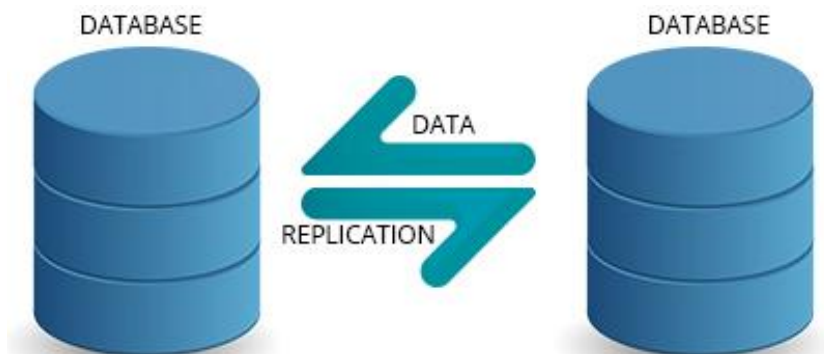


AZURE LOCKS

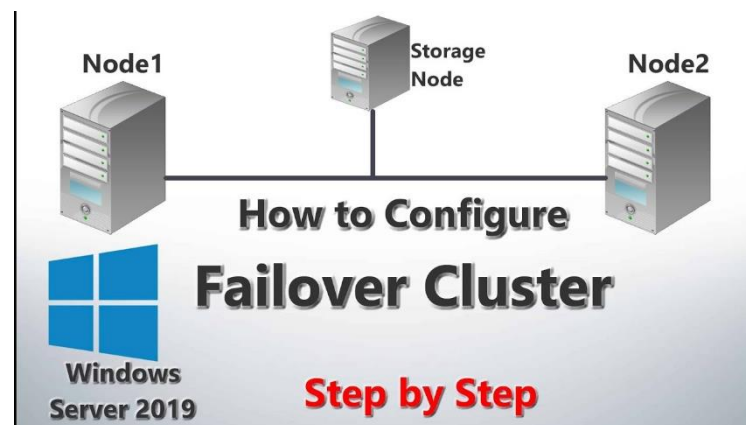
Resource Locking

1.3 AVAILABILITY DESIGN

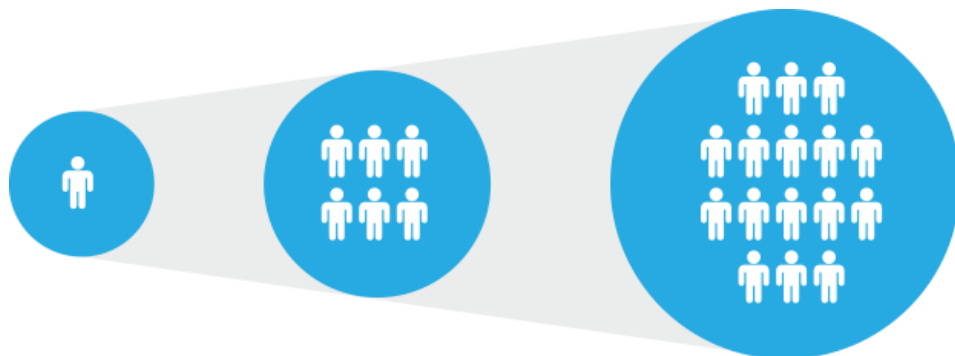
Aby chronić się przed atakami DoS/DDoS, musicie poprawnie zakodować oprogramowanie. Ponieważ jest to faza projektowania, można rozważyć wymagania konfiguracyjne. Poniżej przedstawiono metody używane do projektowania oprogramowania pod kątem dostępności:



Replication



Failover



Scalability



1.4 AUTHENTICATION DESIGN

Rozważcie użycie **uwierzytelniania wieloskładnikowego i jednokrotnego logowania (SSO)**.

1.5 AUTORIZATION DESIGN

Należy określić typ autoryzacji, który ma zostać zaimplementowany zgodnie z wymaganiami. Projektowanie pod kątem autoryzacji można zrealizować za pomocą zarządzania uprawnieniami, które polega na szczegółowej kontroli dostępu.

Mechanizmy kontroli dostępu:

- Directory
- Access Control List (ACL)
- Access control matrix
- Capability
- Procedure-oriented access control

1.6 ACCOUNTABILITY DESIGN

Audyt oprogramowania, zwłaszcza w przypadku naruszenia, przede wszystkim do celów kryminalistycznych.



3. FAZA BEZPIECZNEGO KODOWANIA

24

PHASE 3: SECURITY DEVELOPMENT

CELE

- Zastosować aspekty technologiczne i podstawowe zasady bezpiecznego tworzenia kodu.
- Weryfikować, że kontrola bezpieczeństwa określona w wymaganiach bezpieczeństwa wdrożona w kodzie
- Weryfikować, że uniknięto zagrożeń zidentyfikowanych na podstawie modelowania zagrożeń.

ZADANIA BEZPIECZEŃSTWA

- typowe luki w oprogramowaniu oraz ich zarządzanie (1.1 Vulnerability databases, 1.2 Defensive coding practices)
- bezpieczne procesy oprogramowania (2.1 Source code versioning, 2.2 Code analysis, 2.3 Code review, 2.4 Developer testing)
- zabezpieczenie środowisk kompilacji, w celu zapewnienia bezpieczeństwa (3.1 Physically securing access to the software's that building code, 3.2 Using access control lists (ACLs), 3.3 Using the version control software, 3.4 Build automation, 3.5 Code signing routine)

1.1 VULNERABILITY DATABASES

Bazy danych podatności: **CVE (Common Vulnerabilities and Exposures)**, **NVD (National Vulnerability Database)** oraz **CVSS (Common Vulnerability Scoring System)**.



<https://cve.mitre.org/>



<https://nvd.nist.gov/vuln/full-listing>



<https://www.first.org/cvss/>



1.2 DEFENSIVE CODING PRACTICES

Programowanie defensywne jest to zbiór technik programowania mających na celu zapewnienie poprawnego działania kodu i minimalizacji możliwości jego niepoprawnego wykorzystania. Techniki te skupiają się głównie na poprawie ogólnej jakości kodu, łatwości jego zrozumienia i zapewnienia przewidywalnego działania

Techniki programowania defensywnego:

- Upraszczanie kodu źródłowego.
- Zewnętrzne audyty kodu źródłowego.
- Wyjątki i asercje.
- Testowanie oprogramowania.
- Bezpieczna obsługa wejścia i wyjścia.
- Zapis danych w postaci kanonicznej.
- Zasada najmniejszego uprzywilejowania.



2.1 SOURCE CODE VERSIONING

System kontroli wersji służy między innymi do śledzenia informacji o zmianach na plikach. Dostarcza takie możliwości jak przywrócenie zmian, dodanie nowych modyfikacji, czy sprawdzenie jakie zmiany zostały ostatnio wykonane. Najpopularniejsze systemy to **GIT** oraz **Mercurial**

GIT	
+ Pros	- Cons
<ul style="list-style-type: none">+ Interactive rebase: Rewrite history, - rebase / modify a commit+ Git doesn't track renames+ Git is faster than Mercurial for Network Operations+ Git's approach to branches more powerful+ Addons can be written in many languages+ Index, a powerful staging tool+ Can convert Git repository to Mercurial and back again without data loss	<ul style="list-style-type: none">- Overly Complicated / complex modes of the checkout command also revert + reset- Git doesn't track renames- Git is slower on Windows than Mercurial- Large commands – tedious to input- Usability, not user friendly at all- Addons have limitations- Index, adds an extra step to everything- Rebase can be destructive, accidents do happen, and when they do it can affect the entire team.

MERCURIAL	
+ Pros	- Cons
<ul style="list-style-type: none">+ Easier to learn than Git+ Bookmarks in Mercurial share a single namespace+ Mercurial is better at merging than Git+ GUI, Usability is better than Git+ Help via HG Help is better than GIT help+ Revision numbers rather than GIT SHAS+ Branches visually easier to understand+ Optional Revsets to emulate Index when needed+ Rebase alternative extensions, such as collapse, histedit or MQ are additional addons reducing the chance of accidental use, also permanent backups are made.+ Cannot rewrite history – read only history (unlike Git)+ Webserver built in	<ul style="list-style-type: none">- Addons must be written in Python- Combining features and functionality is problematic when using different extensions- Can't roundtrip convert Mercurial Repository to a GIT one then back again without data loss- Rollback command will only undo the last commit, additional extensions must be used for more, – GIT provides greater control out of the box and overall better control over history- No partial checkouts – big limitation for large projects (need to create all the files before hiding them with Mercurial)

2.2 CODE ANALYSIS

Scanners	Java	.NET	C, C++	Python	Golang	Ruby/Rails	PHP
Bandit	X	X	X	√	X	X	X
Brakeman	X	X	X	X	X	√	X
Codesake Dawn	X	X	X	X	X	√	X
FindBugs	√	X	X	X	X	X	X
FindSecBugs	√	X	X	X	X	X	X
Flawfinder Flawfinder	X	X	√	X	X	X	X
Graudit	√	√	X	√	X	√	√
LGTM	√	X	√	√	X	X	X
Progpilot	X	X	X	X	X	X	√
PreFast (Microsoft)	X	X	√	X	X	X	X
Puma Scan	X	√	X	X	X	X	X
.NET Security Guard	X	√	X	X	X	X	X
RIPS	X	X	X	X	X	X	√
phpcs-security-audit	X	X	X	X	X	X	√
SonarQube	√	√	√	√	√	√	√
VisualCodeGrepper (VCG) -	X	X	√	X	X	X	√

Źródło <https://www.qualitynoc.com/source-code-analysis-tools/>



2.3 CODE REVIEW

Code Review (w skrócie **CR**, po polsku *inspekcja kodu*, *przegląd kodu* lub *czytanie kodu*) — to praktyka polegająca na przeglądaniu kodu napisanego przez programistę przez drugą osobę (najczęściej kolegę z zespołu) w celu zmniejszenia ryzyka wystąpienia błędów oraz dostosowania go do przyjętych w danej firmie standardów i założeń związanych z architekturą.

Dobre praktyki Code Review

- Zasada nr 1: Mniej znaczy więcej
- Zasada nr 2: Checklista Code Review
- Zasada nr 3: Narzędzia do Code Review
- Zasada nr 4: Spraw, by miało to znaczenie

CODE REVIEW



Downloaded by User Name Ltd
from Source Project

```
function register()
{
    if (!empty($_POST)) {
        $msg = '';
        if ($_POST['user_name']) {
            if ($_POST['user_password_new']) {
                if ($_POST['user_password_new'] == $_POST['user_password_repeat']) {
                    if (strlen($_POST['user_password_new']) > 5) {
                        if (strlen($_POST['user_name']) < 65 && strlen($_POST['user_name']) > 1) {
                            if (preg_match('/^[a-z\d]{2,64}$/i', $_POST['user_name'])) {
                                $user = read_user($_POST['user_name']);
                                if (!isset($user['user_name'])) {
                                    if (!isset($_POST['user_email'])) {
                                        if (strlen($_POST['user_email']) < 65) {
                                            if (filter_var($_POST['user_email'], FILTER_VALIDATE_EMAIL)) {
                                                create_user();
                                                $_SESSION['msg'] = 'You are now registered so please login';
                                                header('Location: ' . $_SERVER['PHP_SELF']);
                                                exit();
                                            } else $msg = 'You must provide a valid email address';
                                        } else $msg = 'Email must be less than 64 characters';
                                    } else $msg = 'Email cannot be empty';
                                } else $msg = 'Username already exists';
                            } else $msg = 'Username must be only a-z, A-Z, 0-9';
                        } else $msg = 'Username must be between 2 and 64 characters';
                    } else $msg = 'Password must be at least 6 characters';
                } else $msg = 'Passwords do not match';
            } else $msg = 'Empty Password';
        } else $msg = 'Empty Username';
        $_SESSION['msg'] = $msg;
    }
    return register_form();
}
```



2.4 DEVELOPER TESTING

Testowanie programistów jest czynnością polegającą na testowaniu regresji kodu źródłowego przez programistów. Jest to czasami nazywane „testowaniem regresji jednostkowej”, ale wiele testów programistycznych wykracza poza testy jednostkowe, aby zająć się również testami integracyjnymi.

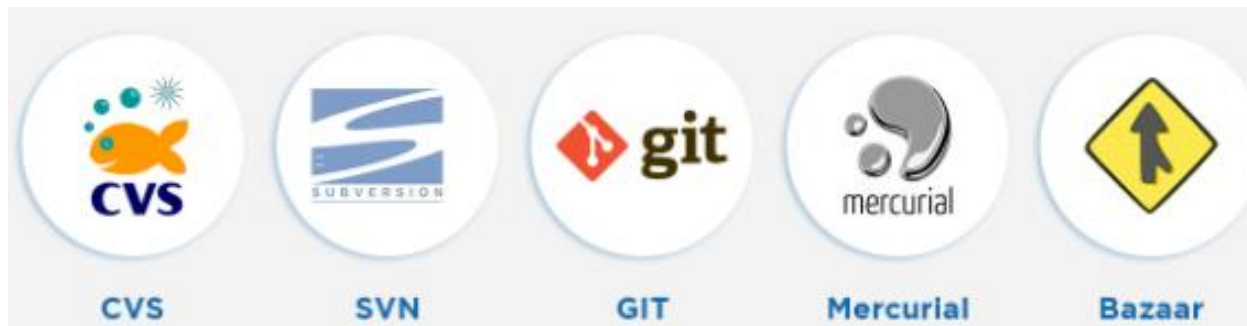


3.1 PHYSICALLY SECURING ACCESS TO THE SOFTWARE'S THAT BUILDING CODE

3.2 USING ACCESS CONTROL LISTS

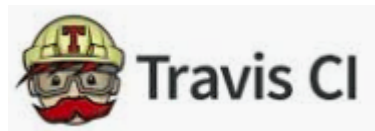
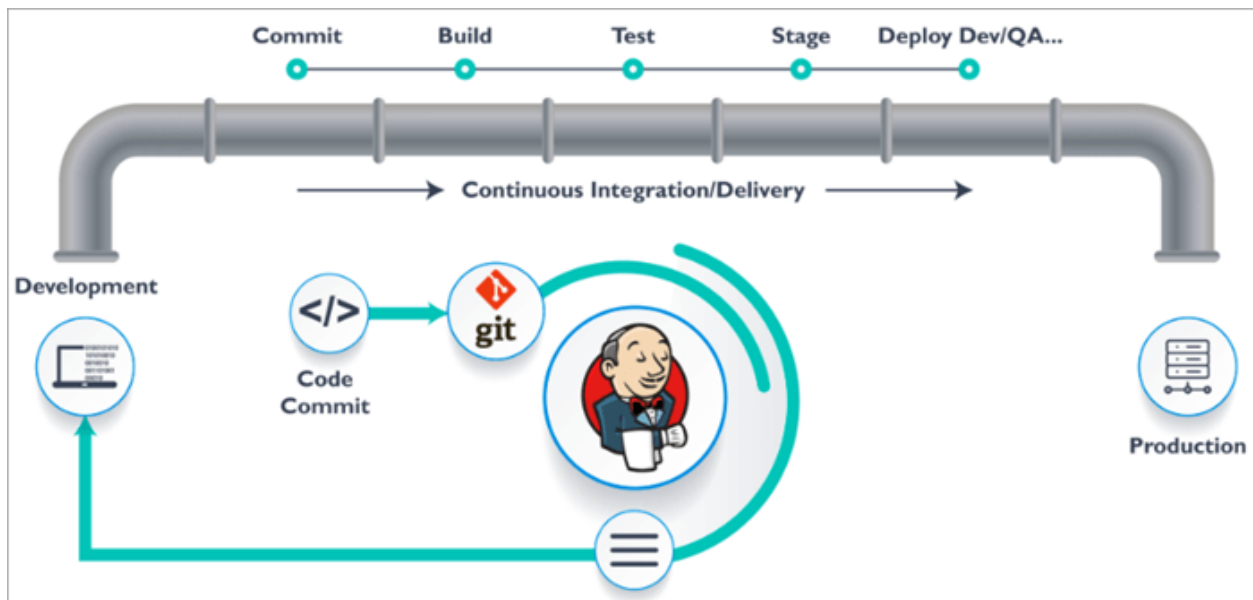
Access-control list, ACL (ang., lista kontroli dostępu) – lista uprawnień skojarzonych z obiektem komputerowego systemu plików. Określa, którzy użytkownicy lub procesy systemowe mają dostęp do obiektów, a także jakie operacje są dozwolone na danych obiektach. Każdy wpis na typowej liście ACL określa obiekt i operację.

3.3 USING THE VERSION CONTROL SOFTWARE



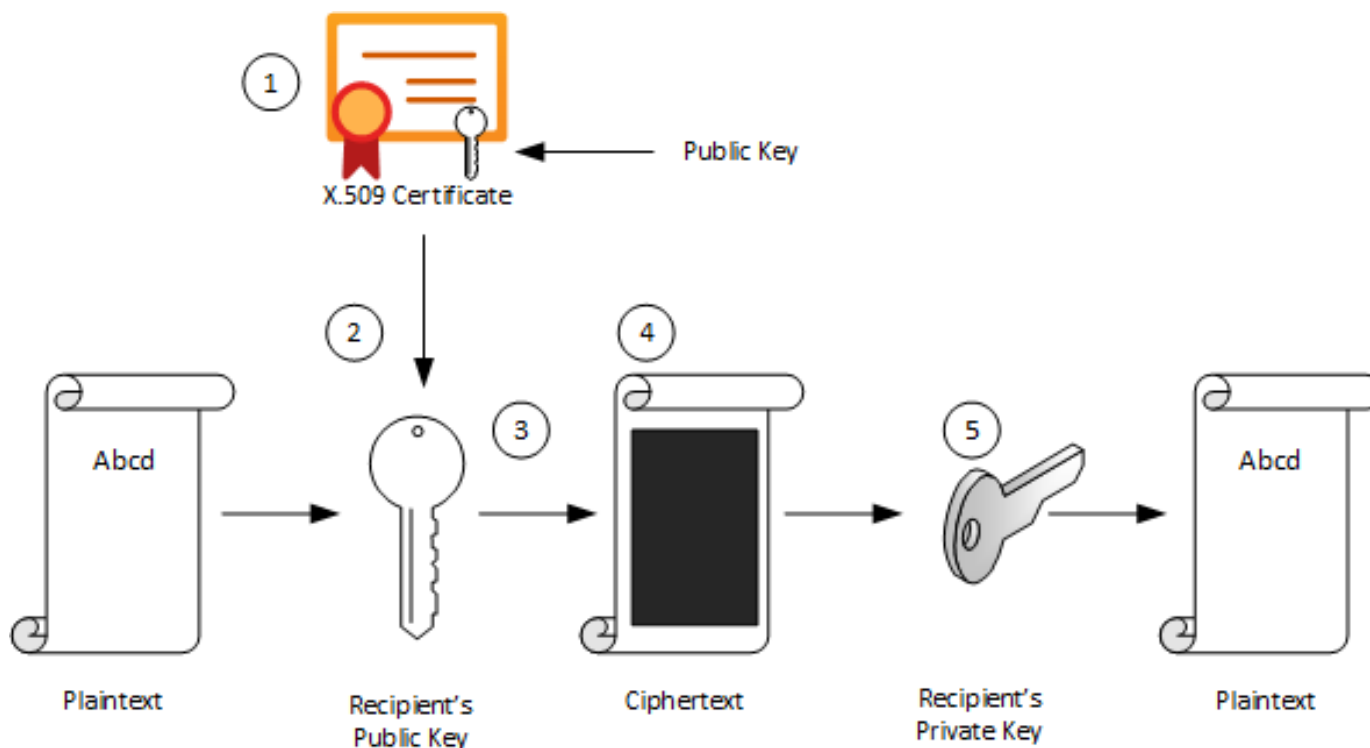
3.4 BUILD AUTOMATION

Automatyzacja kompilacji to technika stosowana w cyklu życia oprogramowania, w której kod źródłowy oprogramowania jest zgodny z kodem języka maszynowego komputera za pomocą skryptu kompilacji automatyzacji .



3.5 CODE SIGNING ROUTINE

Podpisywanie kodu to metoda umieszczania podpisu cyfrowego na pliku, programie lub aktualizacji oprogramowania, która gwarantuje, że plik wykonywalny nie został naruszony. Jest to sposób na zapewnienie użytkownikowi dodatkowego poziomu pewności, że przedmiot jest autentyczny i bezpieczny w użyciu.




4. FAZA TESTOWANIA BEZPIECZEŃSTWA

PHASE 4: SECURITY TESTING

CELE

- Walidacja i weryfikacja funkcjonalności i bezpieczeństwa oprogramowania za pomocą testów zapewnienia jakości
- Zapewnić, że opracowany kod działa zgodnie z przeznaczeniem

ZADANIA BEZPIECZEŃSTWA

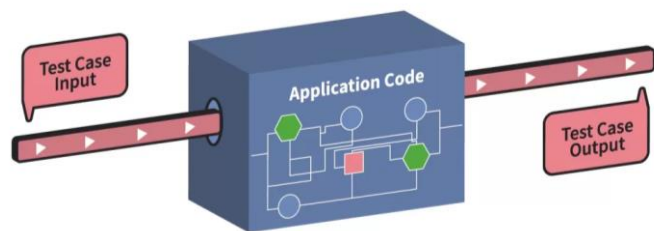
- walidacja powierzchni ataku (1.1 Post-development testing, 1.2 Perform security testing using security testing methods, 1.3 Perform software security testing for quality assurance)
 - zarządzanie danymi testowymi (2.1 Identify output test data to confirm software requirements, 2.2 Apply testing with synthetic transactions, 2.3 Test data management solutions, 2.4 Defect reporting and tracking)
- 

1.1 POST-DEVELOPMENT TESTING

W tym kroku wykonuje się dynamiczną analizę kodu, czyli kontroli kodu podczas jego wykonywania.

1.2 PERFORM SECURITY TESTING USING SECURITY TESTING METHODS

Unit Testing Integration Testing System Testing



White box testing



Black box testing



Cryptographic validation testing



1.3 PERFORM SOFTWARE SECURITY TESTING FOR QUALITY ASSURANCE

Weryfikacja bezpieczeństwa jest ważnym i bardzo szerokim pojęciem w procesie sprawdzania jakości oprogramowania.

Testowanie bezpieczeństwa różni się od innych form testowania, w dwóch istotnych obszarach:

- Standardowe techniki wyboru wejściowych danych testowych mogą nie uwzględniać ważnych kwestii bezpieczeństwa.
- Objawy problemów związanych z bezpieczeństwem bardzo się różnią od symptomów, wykrywanych w innych rodzajach testowania.

Testy bezpieczeństwa są następujące:

- a) Testing for input validation
- b) Testing for injection flaws controls
- c) Testing for scripting attacks controls
- d) Testing for non-repudiation controls
- e) Testing for spoofing controls
- f) Testing for error and exception handling controls (failure testing)
- g) Testing for privileges escalations controls
- h) Anti-reversing protection testing
- i) Stress testing




2.1 IDENTIFY OUTPUT TEST DATA TO CONFIRM SOFTWARE REQUIREMENTS

Identyfikacja oczekiwanych danych wyjściowych pomaga potwierdzić, czy oprogramowanie spełnia wymagania. Jakość danych testowych jest bezpośrednio związana z jakością samego testu, dlatego danymi testowymi należy zarządzać. Ze względu na przepisy (o ochronie prywatności) **aplikacji nie mogą wykorzystywać rzeczywistych danych produkcyjnych jako danych testowych.**

2.2 APPLY TESTING WITH SYNTHETIC TRANSACTIONS

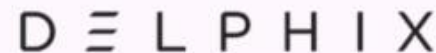
Wykonywania transakcji, które nie mają wartości biznesowej, która jest związana z danymi syntetycznymi. **Dane syntetyczne to sztucznie wygenerowane informacje, które można wykorzystać zamiast prawdziwych danych.**



2.3 TEST DATA MANAGEMENT SOLUTIONS

Zapewnia zintegrowane i poufne mechanizmy wyszukiwania danych, klasyfikację biznesową i maskowanie danych oparte na regułach w celu deidentyfikacji i bezpiecznego wykorzystania danych produkcyjnych, wykorzystywanych w środowiskach testowych oraz programistycznych. TDM może być wdrażany lokalnie lub w chmurze.

BEST TEST DATA MANAGEMENT TOOLS



2.4 DEFECT REPORTING AND TRACKING

Narzędzie do śledzenia błędów może pomóc w rejestrowaniu defektów, raportowaniu, przypisywaniu i śledzeniu w projekcie tworzenia oprogramowania. Jeśli znajdziemy problemy na wczesnym etapie rozwoju, a jeśli to możliwe, zanim napotka je użytkownik końcowy, mamy większą szansę na wprowadzenie prostych poprawek przy minimalnym wpływie na terminy i budżety projektów.



Kanboard



TAIGA



zendesk



Jira



WhiteSource



Wrike

zipBoard

OverOps



flexible project management

Trello



CODE CLIMATE



Azure DevOps

servicenow



Bugzilla



trac

Integrated SCM & Project Management

5. FAZA WDRAŻANIE BEZPIECZEŃSTWA

PHASE 5: SECURITY DEPLOYMENT

CELE

- Zapewnić wykonania planu operacyjnego i dokumentacji aplikacyjnej
- Zapewnić zatwierdzenie przez kierownictwo i akceptację ryzyka dla wdrożenia
- Zapewnić, że aplikacja spełnia swoją funkcjonalność i jest zabezpieczona
- Zapewnić bezpieczne środowisko i konfigurację do wdrożenia

ZADANIA BEZPIECZEŃSTWA

- Kwestie akceptacji oprogramowania (1.1 Completion criteria, 1.2 Change management, 1.3 Approval to deploy or release, 1.4 Risk acceptance and exception policy, 1.5 Documentation of software)
- Weryfikacja i walidacja (2.1 Reviews, 2.2 Testing)
- Certyfikacja i akredytacja (3.1 Obtain certification, 3.2 Obtain accreditation)
- Instalacja (4.1 Hardening, 4.2 Environment configuration, 4.3 Release management, 4.4 Bootstrapping and secure startup)

1.1 COMPLETION CRITERIA

Weryfikacja wszystkich wymagań funkcjonalnych i wymagań bezpieczeństwa.

Przykłady kroków związanych z bezpieczeństwem obejmują:

- tworzenie wymagań bezpieczeństwa, oprócz wymagań funkcjonalnych, na etapie wymagań;
- uzupełnienie modelu zagrożenia na etapie projektowania;
- przegląd i podpisanie architektury bezpieczeństwa pod koniec fazy projektowania;
- sprawdzenie kodu pod kątem luk w zabezpieczeniach po fazie rozwoju;
- zakończenie testów bezpieczeństwa pod koniec fazy testowania programu; oraz
- skompletowanie dokumentacji przed rozpoczęciem fazy rozmieszczania.

1.2 CHANGE MANAGEMENT

Zarządzanie zmianą to podzbiór zarządzania konfiguracją. Zmiany w środowisku komputerowym i przeprojektowanie architektury zabezpieczeń mogą potencjalnie stworzyć nowe luki w zabezpieczeniach, zwiększając tym samym ryzyko. W celu wdrożenia oprogramowania należy ustanowić niezbędne kolejki wsparcia i procesy.

1.3 APPROVAL TO DEPLOY OR RELEASE

Przed instalacją nowego oprogramowania należy przeprowadzić analizę ryzyka i określić ryzyko szczątkowe. Wyniki analizy ryzyka wraz z działaniami eliminującymi (łagodzącymi lub akceptującymi) należy przekazać właścicielowi firmy. Osoba upoważniona musi zostać poinformowana o ryzyku szczątkowym. Zatwierdzenie lub odrzucenie wdrożenia powinno obejmować wskazówki i wsparcie ze strony zespołu ds. bezpieczeństwa. Docelowo za zatwierdzanie zmian odpowiedzialny jest authorizing official (AO)

1.4 RISK ACCEPTANCE AND EXCEPTION POLICY

W pierwszej kolejności konieczne jest określenie ryzyka, który pozostaje po wdrożeniu bezpieczeństwa (ryzyko szczątkowe). Najlepszym rozwiązaniem ogólnego ryzyka jest jego ograniczenie, tak aby ryzyko szczątkowe spadł poniżej progu zdefiniowanego przez biznes (w tym przypadku ryzyko szczątkowe może zostać zaakceptowany). Ryzyko powinien podejmować właściciel firmy, a nie urzędnicy z działu IT

1.5 DOCUMENTATION OF SOFTWARE

Konieczne jest zapewnienie dostępności całej niezbędnej dokumentacji.



2.1 REVIEWS

Opinie należy przeprowadzić na końcu każdej fazy, aby upewnić się, że oprogramowanie działa prawidłowo i spełnia specyfikacje biznesowe.

2.2 TESTING

Na tym etapie należy wykazać i zidentyfikować wszelkie odchylenia od oczekiwanych wyników testowania.

Rodzaje testów, które są przeprowadzane na tym etapie:

- a) Testy wykrywania błędów
- b) Testy akceptacyjne
- c) Niezależne testy (strony trzeciej)

3.1 OBTAIN CERTIFICATION (1)

Usługa certyfikacji oprogramowania polega na gruntownym i całościowym zweryfikowaniu jakości oprogramowania w obszarze zdefiniowanym przez klienta lub opisanym przez normy i standardy.

Certyfikacja oprogramowania w oparciu o uznane metody i wiarygodnego partnera to obiektywna ocena jakości aplikacji czy systemu, potwierdzona certyfikatem jakości.

Audyt bezpieczeństwa produktu uwzględnia bezpieczeństwo danych i funkcjonalności przed zewnętrznymi i wewnętrznymi atakami. Certyfikacja bezpieczeństwa dotyczy oprogramowania w środowisku pracy.

Certyfikacja obejmuje ocenę wiarygodności następujących elementów:

- a) **Uprawnienia użytkowników, uprawnienia i zarządzanie profilami**
- b) **Wrażliwość danych i zastosowanie odpowiednich kontrole**
- c) **Konfiguracje oprogramowania, sprzętu i lokalizacji**
- d) **Relacje i zależności**
- e) **Tryb bezpieczeństwa pracy**

3.1 OBTAIN CERTIFICATION (2)

Kryteria oceny:

- CC Common Criteria (Wspólne kryteria do oceny zabezpieczeń informatycznych)
- PN-EN ISO/IEC 15408:2020-09 Techniki zabezpieczeń – Kryteria oceny zabezpieczeń;
- PN-EN ISO/IEC 19790:2020-09 - Techniki Bezpieczeństwa - Wymagania zabezpieczeń dla modułów kryptograficznych.

Metodyki oceny:

- CEM (Wspólna metodyka oceny zabezpieczeń informatycznych)
- PN-EN ISO/IEC 18045:2020-09 - Techniki bezpieczeństwa - Metodyka oceny zabezpieczeń informatycznych;
- ISO/IEC 24759:2017 Test requirements for cryptographic module.

3.2 OBTAIN ACCREDITATION

Jest to oficjalna decyzja kierownictwa o użytkowaniu oprogramowania w trybie bezpieczeństwa operacyjnego przez określony czas i jest formalną akceptacją zidentyfikowanych ryzyka związanych z obsługą oprogramowania.

4.1 HARDENING

Utwardzanie (ang. *Hardening*) to proces z zakresu bezpieczeństwa, którego celem jest ograniczenie płaszczyzny ataku. Stopień skomplikowania procesu zależy od rozmiaru utwardzanego systemu. Zgodnie z zasadą, iż łatwiej jest zabezpieczyć system wykonujący jedną funkcję niż wielofunkcyjny wykorzystywany w różnych celach.

Utwardzanie systemów ma na celu takie ich skonfigurowanie, by zminimalizować ryzyko ataku z zewnątrz i wpływu złośliwego oprogramowania na użytkownika. Eksperci, w ramach podejmowanych działań, zwracają szczególną uwagę na stan zabezpieczeń newralgicznych elementów – zarówno systemów operacyjnych, jak i urządzeń.

Servers	Operating systems	Applications	Databases
<ul style="list-style-type: none">■ Install all security patches■ Two different network interfaces■ Keep backups of all data and files	<ul style="list-style-type: none">■ Install latest service packs■ File and file system encryption■ Run services with least privileged accounts	<ul style="list-style-type: none">■ Install all security patches■ Implement an SSL architecture■ Install a web application firewall	<ul style="list-style-type: none">■ Locking and expiring unused accounts■ Role-based access control privileges■ Perform periodic data-base security audits

4.2 ENVIRONMENT CONFIGURATION

Bardzo ważne jest, aby środowisko programistyczne i testowe było zgodne z konfiguracją środowiska produkcyjnego oraz aby próbne testy identycznie emulowały konfigurację środowiska, w którym oprogramowanie zostanie wdrożone.

Sl. No.	Pre-installation Activity
1	<u>Install</u> all the prerequisite hardware and software given in the Tech Stack.
2	<u>Verify</u> the System Environment using the Environment Check Utility.
3	<u>Configure</u> the Database Instance settings.
4	<u>Install</u> and configure the web application server.
5	<u>Configure</u> the HTTP settings on the webserver.
6	<u>Create</u> the Installation, Download, and Metadata Repository Directories: <ul style="list-style-type: none">• Installation directory• Temporary directory• Staging Area/Metadata Repository• Download directory

4.3 RELEASE MANAGEMENT

Zarządzanie wersjami to proces nadzorowania i kontrolowania wydań oprogramowania w celu odpowiedniego ich zaplanowania i utrzymania możliwie płynnego wdrażania nowych wydań.

Proces zarządzania wersjami rozpoczyna się od wniosków przesłanych do zespołów programistycznych. Mogą to być żądania użytkowników od osób, które chcą większej funkcjonalności, a także sugestie lub prośby urzędników firmy, którzy chcą myśleć przyszłościowo i projektować potrzeby użytkowników. Zespół dokonuje przeglądu tych wniosków, dyskutuje o trudnościach związanych z ich wdrażaniem i zapewnia porady dotyczące dalszego postępowania, czy też odkładania wniosków na przyszłość. W końcu opracują listę zmian, które należy wprowadzić w oprogramowaniu i mogą wejść w fazę rozwoju.

4.4 BOOTSTRAPPING AND SECURE STARTUP

W komputerach, Bootstrap loader jest pierwszym fragmentem kodu, który działa, gdy maszyna uruchamia się i jest odpowiedzialny za załadowanie reszty systemu operacyjnego. W nowoczesnych komputerach jest przechowywany w pamięci ROM, ale przypominam sobie proces bootstrap na PDP-11, w którym za pomocą przełączników na panelu przednim wczytywano bity w celu załadowania określonego segmentu dysku do pamięci, a potem go uruchom.

Na tym etapie należy zweryfikować, czy procesy uruchamiania oprogramowania nie wpływają na poufność, integralność lub dostępność oprogramowania podczas instalacji oprogramowania. Autotest po włączeniu zasilania jest pierwszym etapem ładowania programu początkowego i jest zdarzeniem, które musi być zabezpieczone przed manipulacją.

5. FAZA UTRZYMANIA BEZPIECZEŃSTWA

PHASE 6: SECURITY MAINTENANCE

CELE

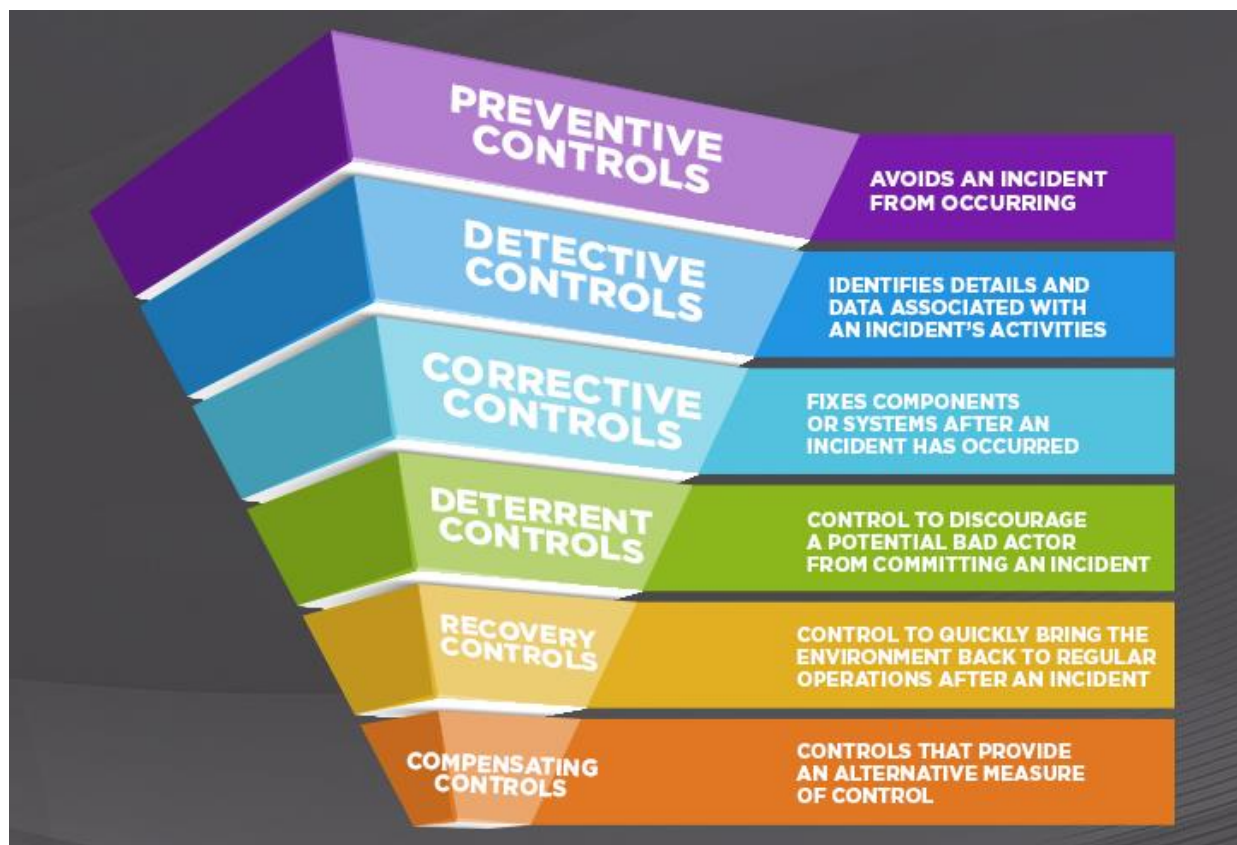
- Monitorować i zagwarantować, że oprogramowanie będzie nadal działać w sposób niezawodny i odporny.
- Zidentyfikować warunki, w których oprogramowanie musi zostać usunięte lub wymienione.

ZADANIA BEZPIECZEŃSTWA

- operacje monitorowania i utrzymania (1.1 Carry out the operations security, 1.2 Continuous monitoring, 1.3 Audit for monitoring)
- zarządzanie incydentami (2.1 Determine events, alerts, and incidents, 2.2 Identify types of incidents, 2.3 Incident response process)
- zarządzanie problemem (3.1 Incident notification, 3.2 Root cause analysis, 3.3 Solution determination, 3.4 Request for change, 3.5 Implement solution, 3.6 Monitor and report)
- zarządzanie zmianami (4.1 Patch and vulnerability management, 4.2 Backups, recovery and archiving)
- Utylizacja (5.1 End-of-Life policies, 5.2 Information disposal and media sanitization)

1.1 CARRY OUT THE OPERATIONS SECURITY

Zapewnienie bezpieczeństwa operacji polega na utrzymaniu bezpieczeństwa lub utrzymywaniu poziomów odporności oprogramowania powyżej akceptowalnych poziomów ryzyka.



Rodzaje kontroli bezpieczeństwa operacji

1.2 CONTINUOUS MONITORING

Przed wdrożeniem rozwiązania do monitorowania ważne jest, aby najpierw zdefiniować wymagania dotyczące monitorowania. Ciągłe audyty bezpieczeństwa powinny być przeprowadzane w zaplanowanych odstępach czasu lub jako zmiany w zależności od potrzeb lub wymagań.

1.3 AUDIT FOR MONITORING

Audyt jest przeprowadzany przez audytora, którego obowiązki obejmują wybór zdarzeń do audytu w oprogramowaniu, ustawienie flag audytu, które umożliwiają rejestrację i analizę tych zdarzeń audytowych. Audyty powinny być przeprowadzane okresowo.

2.1 DETERMINE EVENTS, ALERTS, AND INCIDENTS

EVENT - Jest to zaobserwowana zmiana w normalnym **zachowaniu urządzenia/systemu/procesu/ użytkownika**. W zasadzie wszystkie logi to eventy.

INCYDENT - To zdarzenie (event), które negatywnie wpływa na działalność biznesową firmy. Najczęściej incydent rozpoczyna się od eventu wywołującego alert, który analityk lub system kwalifikuje do rangi incydentu. Możliwe jest jeszcze podniesienie rangi bez wywołania alertu – na podstawie decyzji administratora. **Incydent może być złożony z kilku eventów.**

ALERT - to powiadomienie o wystąpieniu określonego zdarzenia (lub serii zdarzeń), które jest wysyłane do odpowiedzialnych stron w celu rozpoczęcia akcji. Alert to coś o czym ma Cię powiadomić urządzenie (SIEM, firewall, DLP) na podstawie reguł bezpieczeństwa, które zaprogramowałeś. **Alert może dotyczyć kilku eventów.** Alerty zazwyczaj pojawiają się w formie *pop-up*, są podświetlane lub wyszczególniane w inny sposób.

2.2 IDENTIFY TYPES OF INCIDENTS

Jest kilka różnych typów incydentów bezpieczeństwa:

- **Nadużycia konta na komputerze**
- **Wtargnięcie do sieci bądź komputera**
- **Kradzież bądź wyciek informacji**
- **Malware**
- **DOS/DDOS**
- **Zaszyfrowanie bądź zniszczenie krytycznych informacji**
-
- **Ujawnienie poufnych**
- **Kradzież systemu**
- **Zniszczenie systemu**
- **Naruszenie integralności danych**
- **Sabotaż**

2.3 INCIDENT RESPONSE PROCESS

Właściwe podejście do reakcji na atak, powinno znaleźć swoje odzwierciedlenie w planach postępowania na wypadek jego wystąpienia. By być przygotowanym na atak i podjąć właściwe kroki, organizacja musi mieć opracowany całościowy plan postępowania na wypadek wystąpienia incydentu (Incident Response Plan - IRP).

Dobrze przygotowana strategia przeciwdziałania incydentom i zwalczaniu tych, które już wystąpiły powinna być oparta na 6 krokach:

1. Przygotowanie (ang. Preparation)
2. Identyfikacja (ang. Identification)
3. Ograniczanie (ang. Containment)
4. Środki naprawcze (ang. Remediation)
5. Odzyskiwanie - powrót do stanu sprzed ataku (ang. Recovery)
6. Wnioski (ang. Aftermath)

3.1 INCIDENT NOTIFICATION

Identyfikowanie i zgłoszenie nieznanego incydentu lub problemu z oprogramowaniem.

Major Incident Notification Template



swapnil@techno-pm.com
To swapnil@techno-pm.com

Reply Reply All Forward ...

Thu 17/12/2020 2:42 PM

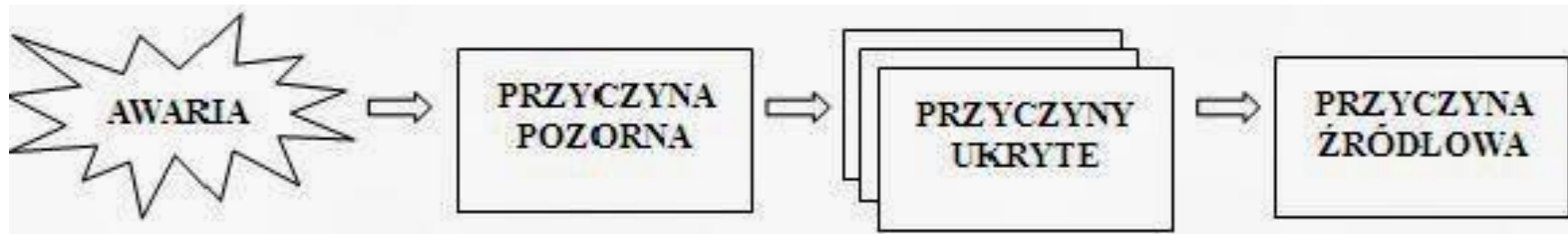
Major Incident Management Notification					
Incident Number	CRT-3445	Incident Status	OPEN	Start Time	Thu 17 th Dec, 8:30 AM EST
Incident Manager	Swapnil Wale	Incident Priority	Priority 1 (P1)	Estimated End Time	Not Available
Bridge Details	Phone Number : +61 2 8003 4979 / Passcode : 2312321				

Incident Details	Business Impact	Incident Timeline
<p>Around 8:30 AM on Friday 13th May 2019 few users reported that they cannot access the system once they log off and some users reported seeing error on screen which are not normal. The user tickets were logged using Service Now, phone calls and, we had some people approach the service desk. A critical incident CRT-3445 was opened at 8:56 AM.</p> <p>The issue was referred to the applications service team. The services team started their investigation around 9:15 AM. During the initial investigation it was identified by reviewing the system logs that database was not accessible. The incident was then referred to the DBAs for further investigation.</p> <p>Root Cause : Unknown</p>	<p>Currently, all the users have been logged out. We have received around 50 calls this morning and have more than 150 tickets related to this incident. This incident has caused customer requests to be backed up and estimated time lost so far is around 4 hours.</p>	<p>8:30 AM : Users reported issue 8:40 AM : Incident manager engaged 9:00 AM : Resource manager engaged 9:15 AM : Incident triage started 9:20 AM : First comms sent out</p>

PS : Please contact service desk on ext. 0078 if you need more details.

3.2 ROOT CAUSE ANALYSIS

Analiza przyczyn źródłowych RCA (**Root Cause Analysis**) jest kompletną i niezależną metodą analizy ryzyka, służącą do identyfikacji, badania oraz klasyfikowania przyczyn źródłowych zdarzeń zagrażających (inicjujących ryzyko) przedsiębiorstwu. RCA jest zestawem narzędzi, podejść i procesów do przeprowadzania analizy różnego rodzaju awarii a w rezultacie znajdowania przyczyn źródłowych ich powstawania.



3.3 SOLUTION DETERMINATION

Podejmowana jest decyzja, czy w oprogramowaniu należy zastosować tymczasowe czy stałe poprawki.

3.4 REQUEST FOR CHANGE

Na tym etapie tworzony jest wniosek o zmianę dla znanych błędów i aktualizowane są informacje o problemach z bezpieczeństwem.

3.5 IMPLEMENT SOLUTION

Wdrożenie zidentyfikowanego rozwiązania po zainicjowaniu wniosku o zmianę.

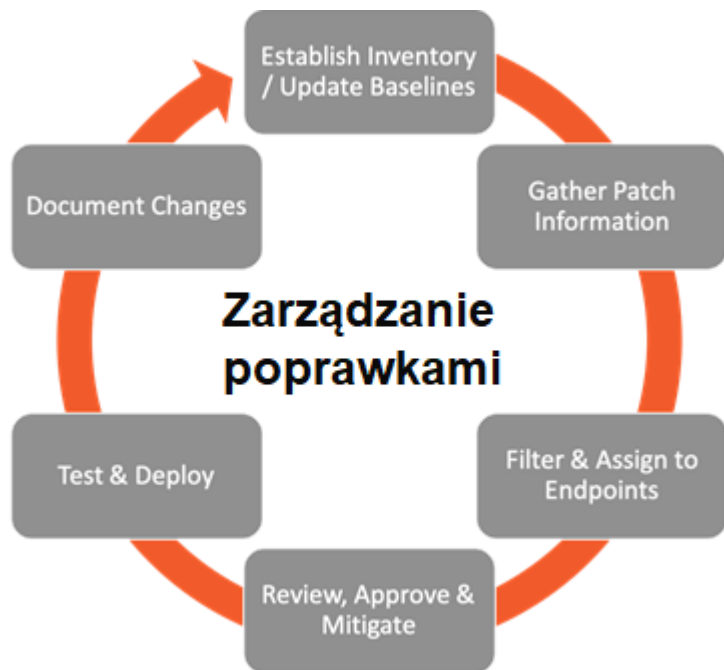
3.6 MONITOR AND REPORT

Tworzony jest raport monitorowania i rozwiązywania problemów.

4.1 PATCH AND VULNERABILITY MANAGEMENT

Zarządzanie poprawkami to proces zarządzania oprogramowaniem poprzez regularne wdrażanie wszystkich brakujących poprawek w celu zapewnienia aktualności komputerów

Zarządzanie podatnościami jest definiowane jako „cykliczna praktyka identyfikowania, klasyfikowania, ustalania priorytetów, naprawiania i prób załagodzenia skutków wystąpienia” podatności w zabezpieczeniach oprogramowania.



4.2 BACKUPS, RECOVERY AND ARCHIVING

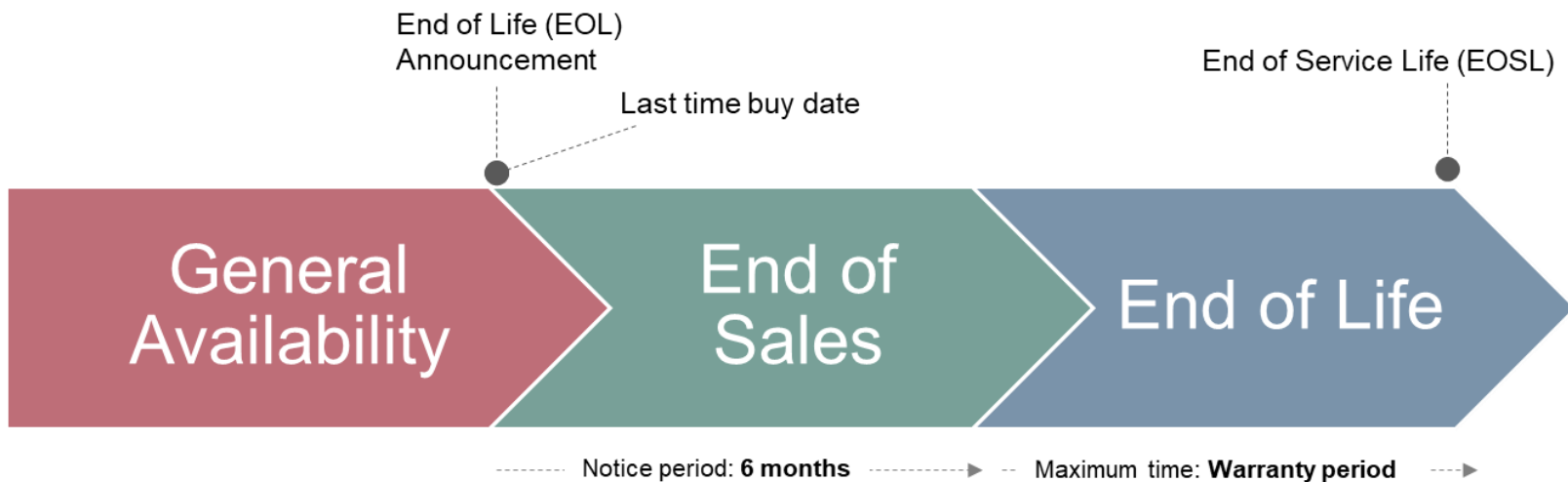
Tworzenie kopii zapasowej danych, inaczej **backup** to nic innego jak dodatkowe zabezpieczenie naszych plików. Służy ono do odtworzenia oryginalnych danych w przypadku ich utraty bądź uszkodzenia.

Odzyskiwanie danych (ang. Data recovery) - to proces **odzyskiwania** niedostępnych, utraconych, zepsutych, uszkodzonych lub sformatowanych **danych** z pamięci dodatkowej, nośników lub plików, gdy nie można uzyskać dostępu w normalny sposób.

Archiwizacja danych (ang. data archiving) **polega na przeniesieniu ich w inne miejsce w pamięci masowej, w celu ich długotrwałego przechowywania**. Często mylone z kopią bezpieczeństwa. W procesie archiwizacji danych dane starsze, mniej używane przenoszone są na wolniejsze, tańsze nośniki danych.

5.1 END-OF-LIFE POLICIES

Ogłoszenie o zakończeniu życia ("End of Life" - EOL). Jest to publikacja dat zakończenia sprzedaży i zakończenia obsługi oprogramowania.



5.2 DATA SANITIZATION

„Sanityzacja danych” oznacza proces certyfikowanego usuwania danych, który jest przeprowadzany na działającym i prawidłowo osadzonym dysku twardym w systemie

Why	<ul style="list-style-type: none">• No longer Need – EoL/Sale/donate• Declass & reuse• Object Resue - Data/ Remenance• Policy/ Classification (high/low)• NIST SP 800-88 (Media Sanitization)• Destruction/ Purging (Verify)	3 Degauss	<ul style="list-style-type: none">• Magnetic Field (Realign/ Rewrite)• Magnetic - HDD/Floppy/Tape• Destroy Electronics (access)• No assure -platter on different Drive• Not Optical (CD/ DVDs) or SSDs (IC)
1 Erasing	<ul style="list-style-type: none">• Deleting (Links), Unallocate Space• Until overwrite (size/ free), Free Tools• Formatting – Replace Address Table• SSD - Build in Erase Cmd but Vendor	4 Purging	<ul style="list-style-type: none">• Intense Clearing (reuse in less class)• Multiple overwrites + Degaussing• Not Top Secret -Effort/ Unknown Tech• Damaged Media-Destroy & Buy New
2 Clearing	<ul style="list-style-type: none">• Overwrite / Electronic Shred/ Wipe• 3 Pass (Char + compliment +bits)• Not Traditional Tools but forensics• Ghost Image (Flex), Spare/Bad, SSDs	5 Destroy	<ul style="list-style-type: none">• Top Secret/ Final Stage/ Disposed off• incinerate, pulverize, crush, shred• disintegration (2mm), acid + Encrypt• CDs/ DVDs/Non-volatile/ Platters