

Cyber Security 2

Status: moet nog worden aangevuld.

1 is herhaling/uitbreiding van Cyber Security 1

2 en 2 zijn belangrijk voor SD

1 Cybersecurity, risico versus impact

Inleiding

In cybersecurity 1 hebben we gekeken naar een aantal soorten cyberaanvallen.

We gaan ons lijstje uitbreiden.

☐ Soorten Cyberaanvallen – Overzicht

1. **Phishing:** iemand probeert jou te misleiden om je wachtwoord af te geven (bv. via een nep-mail van je bank)
2. **Adware:** software dat ongevraagd advertenties toont.
3. **Virus:** speciaal soort malware dat zichzelf kan vermenigvuldigen (net als het Corona virus).
4. **SQL-injection:** via een formulier wordt je database gehackt
5. **Man-in-the-middle:** iemand onderschept je gegevens tussen jou en een website
6. **Trojan Horse:** Lijkt op legitieme software, maar voert op de achtergrond schadelijke acties uit.
7. **Ransomware:** Blokkeert je bestanden of systeem en eist losgeld om weer toegang te krijgen.

8. **Spyware:** Volgt je activiteiten op je apparaat zonder dat je het merkt, vaak om wachtwoorden of surfgedrag te stelen.
9. **Keylogger:** Registreert alles wat je typt, zoals wachtwoorden en privéberichten.
10. **Distributed Denial-of-Service (DDoS):** Een aanval waarbij een website of server overspoeld wordt met verkeer en daardoor onbruikbaar wordt.
11. **Brute Force Attack:** Het systematisch proberen van alle mogelijke wachtwoorden tot het juiste gevonden is.
12. **SQL Injection:** Aanval via een invoerveld waarin schadelijke databasecode wordt ingevoerd.
13. **Cross-Site Request Forgery (CSRF):** De gebruiker wordt misleid om zonder het te weten een actie uit te voeren, zoals een betaling.

Test je kennis - Cyberaanvallen

Opdracht: Lees elk voorbeeld en probeer te raden om welke cyberaanval het gaat. Klik op de vraag om jezelf te controleren.

1. Je krijgt een e-mail van “de bank” met een link waarin je gevraagd wordt opnieuw in te loggen, maar het is een nepwebsite.

Phishing

2. Zodra je een gratis app installeert, verschijnen er overal pop-ups met reclame, zelfs als je de app niet gebruikt.

Adware

3. Een programma op je computer verspreidt zich automatisch naar andere bestanden en computers, net als een griepvirus.

Virus

4. Een website is opeens onbereikbaar omdat duizenden bots tegelijk verzoeken sturen naar de server.

DDoS-aanval

5. Je klikt op een “update”-melding, maar je installeert ongemerkt een programma dat op de achtergrond je bestanden versleutelt en geld vraagt om ze terug te krijgen.

Ransomware

6. Je vult een formulier in op een website, en een hacker gebruikt die invoer om toegang te krijgen tot de database.

SQL-injection

7. Iemand op hetzelfde wifi-netwerk leest mee met de gegevens die jij naar een website stuurt, zonder dat jij het merkt.

Man-in-the-middle

8. Je downloadt een handig ogend programma, maar zodra je het opent blijkt het kwaadaardige acties uit te voeren op je computer.

Trojan Horse

9. Je toetsenbordactiviteit wordt in het geheim vastgelegd zodat iemand je wachtwoord kan achterhalen.

Keylogger

10. Een hacker probeert tienduizenden verschillende wachtwoorden totdat hij de juiste vindt en toegang krijgt tot een account.

Brute Force Attack

11. Een kwaadaardige link zorgt ervoor dat jij zonder het te weten geld overmaakt via een formulier dat je niet hebt ingevuld.

Cross-Site Request Forgery (CSRF)

12. Je surft naar een website en zonder te klikken wordt automatisch schadelijke software op je computer gezet.

Drive-by Download

13. Je bezoekt een vertrouwde website, maar deze is ongemerkt gehackt en stuurt jou door naar een foute pagina.

Watering Hole Attack

14. Een hacker onderschept je login-sessie en gebruikt jouw sessie-ID om toegang te krijgen tot je account.

Session Hijacking

15. Je krijgt een sms van “de pakketdienst” met een link naar een track-and-trace pagina, maar die blijkt nep te zijn.

Smishing

16. Iemand installeert een app op jouw telefoon die in het geheim foto's, locatie en berichten doorspeelt.

Spyware

17. Je gebruikt overal hetzelfde wachtwoord en een hacker probeert deze combinatie op andere websites.

Credential Stuffing

Quizvraag: Wat is het verschil tussen risico en impact?

Risico = Hoe groot is de kans dat het gebeurt?

Impact = Wat zijn de gevolgen als het gebeurt?

Voorbeelden

- **Fiets wordt gestolen bij school**

- *Risico*: Hoog → je laat hem vaak ongeopend buiten staan.
- *Impact*: Middel → je baalt, moet lopen, misschien nieuwe kopen.

- **Telefoon valt in water**

- *Risico*: Laag → je past goed op.
- *Impact*: Hoog → je bent alles kwijt: foto's, apps, schoolwerk.

- **Je vergeet een wachtwoord**

- *Risico*: Middel → gebeurt wel eens.
- *Impact*: Laag → je kunt meestal resetten.

Opdracht

Lees de 6 scenarios en bepaal vervolgens van elke scenario het **risico** en de **impact** en licht je keuze toe.

Scenario 1 Phishing

Stel je voor: je krijgt een e-mail van “de Rabobank” met als onderwerp “**Dringende actie vereist - je account wordt geblokkeerd**”. In het bericht staat dat je onmiddellijk moet inloggen om je rekening te verifiëren. Er staat een link bij die lijkt op de echte site, maar eigenlijk leidt naar een nepwebsite. Zodra je daar je gegevens invult, krijgt een oplichter direct toegang tot je bankaccount.

Scenario 2 Ransomware

Je opent een bijlage in een e-mail die lijkt te komen van je school, bijvoorbeeld “roosterwijziging.pdf”. Maar zodra je het opent, wordt je scherm zwart en verschijnt er een melding: “**Je bestanden zijn versleuteld. Betaal €300 in bitcoins om ze terug te krijgen.**”

Alle documenten, foto's en schoolopdrachten op je laptop zijn geblokkeerd. Dit is een **ransomware-aanval**: je bestanden zijn gegijzeld en je moet losgeld betalen om ze terug te krijgen.

Scenario 3 Adware

Je downloadt een gratis spelletje van een onbekende website. Na de installatie verschijnen er ineens overal pop-ups en reclames, zelfs als je de browser niet open hebt. Je wordt telkens doorgestuurd naar webwinkels of vage aanbiedingen. Je computer wordt trager, en sommige advertenties proberen je te laten klikken op gevaarlijke links. Dit is een typische **adware-infectie**: ongewenste software die reclame toont en soms zelfs je surfgedrag volgt.

Scenario 4 Brute Force

Je hebt een simpel wachtwoord zoals “welkom123” voor je schoolaccount. Een hacker probeert via een script duizenden combinaties van veelgebruikte wachtwoorden, één voor één, totdat jouw wachtwoord wordt geraden. Dit heet een **brute force-aanval**: de aanvaller probeert systematisch allerlei wachtwoorden uit tot hij toegang krijgt tot je account. Als je een zwak of kort wachtwoord gebruikt, ben je hier extra kwetsbaar voor.

Scenario 5 DDos aanval

Je wilt je favoriete webshop bezoeken, maar de site laadt niet of geeft een foutmelding. Op dat moment voeren hackers een **DDoS-aanval** uit: duizenden computers sturen tegelijk nepverzoeken naar de server van de webshop, waardoor die overbelast raakt en voor echte bezoekers onbereikbaar wordt. De website stort tijdelijk in, terwijl er eigenlijk niets “stuk” is — hij wordt gewoon overspoeld met verkeer.

Scenario 6 Man in the Middle

Je zit op een openbaar wifi-netwerk in de trein en logt in op een webshop. Wat je niet weet, is dat een hacker op hetzelfde netwerk zit en al het internetverkeer onderschept. Hij leest ongemerkt mee met de gegevens die jij verstuurt, zoals je inlognaam en wachtwoord. Dit is een **man-in-the-middle-aanval**: iemand zit letterlijk “tussen jou en de website in” en vangt alles op wat je verstuurt, zonder dat je het doorhebt.

Cyberaanval	Risico	Impact	Uitleg
1. Phishing	Laag	Hoog	Risico laag omdat we meestal de nep-email herkennen, maar als we er toch "intrappen" dan is de impact groot want dan kan je (veel) geld verliezen.
2. Ransomware			
3. Adware			

Cyberaanval	Risico	Impact	Uitleg
4. Brute Force			
5. DDoS			
6. Man in the Middle			

☐☐ Inleveren

Bepaal van scenario 2 t/m 6 het risico en de impact en leg uit waarom

2 SQL Injection in PHP

☐☐ Leerdoelen

- Je begrijpt wat een SQL Injection is en waarom het gevaarlijk is.
- Je kunt zelf een simpele SQL Injection uitvoeren op een onveilige PHP-pagina.
- Je kunt uitleggen hoe je dit soort aanvallen voorkomt.

☐☐ Uitleg

Bij een **SQL Injection** wordt misbruik gemaakt van een fout in je code, waarbij de invoer van een gebruiker direct in een SQL-query wordt gezet. Als je niet oppast, kan een hacker zo gegevens uit je database stelen of zelfs verwijderen.

Een klassiek voorbeeld is een inlogformulier waarin de gebruikersinvoer zonder controle in de query komt te staan:

```
<?php
$conn = new mysqli("localhost", "root", "", "testdb");

$username = $_POST['username'];
$password = $_POST['password'];

$sql = "SELECT * FROM users WHERE username = '$username' AND password = '$password'";
```

```
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    echo "Welkom!";
} else {
    echo "Ongeldige login.";
}

?>
```

Als een aanvaller het volgende invoert:

Gebruikersnaam: ' OR '1'='1
Wachtwoord: watdanook

dan wordt de SQL-query dit:

```
SELECT * FROM users WHERE username = '' OR '1'='1' AND password = 'watdanook'
```

Omdat '1'='1' altijd waar is, kan de aanvaller inloggen zonder wachtwoord te kennen.

☐☐ Opdracht – Voer een SQL Injection uit

1. Maak een database `testdb` met een tabel `users`:

```
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(255),
    password VARCHAR(255)
);
```

2. Voeg een gebruiker toe:

```
INSERT INTO users (username, password) VALUES ('admin', 'admin123');
```

3. Maak een PHP-bestand `login.php` met bovenstaande onveilige code.
4. Maak een formulier met twee velden: `username` en `password`.
5. Voer een SQL-injection uit zoals hierboven uitgelegd. Wat gebeurt er?

☐☐ Vervolgopdracht – Beveilig je formulier

1. Pas de code aan zodat je gebruik maakt van **prepared statements**:

```
<?php
$conn = new mysqli("localhost", "root", "", "testdb");

$username = $_POST['username'];
$password = $_POST['password'];

$stmt = $conn->prepare("SELECT * FROM users WHERE username = ? AND password = ?");
$stmt->bind_param("ss", $username, $password);
$stmt->execute();

$result = $stmt->get_result();

if ($result->num_rows > 0) {
    echo "Welkom!";
} else {
    echo "Ongeldige login.";
}
?>
```

2. Test of de SQL Injection nu nog werkt. Leg uit waarom het niet meer lukt.

☐☐ Reflectie

- Waarom is SQL-injection zo gevaarlijk voor een website?
- Wat had de originele programmeur moeten doen om dit te voorkomen?
- Welke techniek moet je gebruiken in PDO om SQL injection te voorkomen.
- Wat gebeurt er als je bij de aangepaste (veilige) code toch SQL-injection probeert?
- Wat zou een reden kunnen zijn dat een software developer geen gebruik maakt van technieken om SQL injection te voorkomen?

☐☐ Inleveren

- Lever je aangepaste code in waarbij SQL injection niet meer mogelijk is.

- Maak de reflectie en lever die in (PDF).

3 Cross-Site Scripting (XSS)

Leerdoelen

- Je weet wat Cross-Site Scripting (XSS) is en hoe het werkt.
- Je kunt zelf een XSS-aanval uitvoeren in een onveilige webpagina.
- Je kunt uitleggen hoe je je website kunt beschermen tegen XSS.

Uitleg

Wat is XSS?

Cross-Site Scripting (XSS) is een aanval waarbij een aanvaller kwaadaardige scripts (zoals JavaScript) invoert in een formulier of URL. Deze scripts worden dan uitgevoerd in de browser van een andere bezoeker.

Stel: je maakt een gastenboek waar mensen een berichtje kunnen achterlaten. Als je hun invoer niet goed filtert, kan iemand dit invoeren:

```
<script>alert('Ik ben gehackt!')</script>
```

Iedereen die daarna het gastenboek bezoekt, krijgt dan deze melding te zien – het script wordt uitgevoerd alsof het van jouw site komt.

Opdracht – Voer een XSS-aanval uit

1. Maak een PHP-bestand `gastenboek.php` met het volgende formulier en afhandelingscode:

```
<form method="post">  
  Naam: <input name="naam"><br>  
  Bericht: <textarea name="bericht"></textarea><br>  
  <input type="submit" value="Verstuur">  
</form>
```

```
<?php
if ($_POST) {
    echo "<h3>Bericht ontvangen:</h3>";
    echo "<p>Van: " . $_POST['naam'] . "</p>";
    echo "<p>" . $_POST['bericht'] . "</p>";
}
?>
```

2. Voer nu als bericht het volgende in:

```
<script>alert('XSS test')</script>
```

3. Wat gebeurt er?

☐☐ Vervolgopdracht – Beveilig je formulier

1. Pas de PHP-code aan zodat gebruikersinvoer wordt ge-escaped:

```
echo "<p>Van: " . htmlspecialchars($_POST['naam']) . "</p>";
echo "<p>" . htmlspecialchars($_POST['bericht']) . "</p>";
```

2. Voer nogmaals het script in. Wat gebeurt er nu?

☐☐ Reflectie

- Wat maakt XSS zo gevaarlijk voor je bezoekers?
- Wat is het verschil tussen `htmlspecialchars()` en `strip_tags()` in PHP?
- Hoe zou jij in een groter project XSS voorkomen?

☐☐ Inleveren

- Screenshot van de popup bij onveilige invoer
- Screenshot van veilige invoer na beveiliging
- Toelichting op je reflectie

- Bestandsnamen:

- `xss-aanval-.jpg`
- `xss-veilige-versie-.jpg`
- `xss-reflectie-.pdf`

4 Cross-Site Request Forgery (CSRF)

□□ Leerdoelen

- Je begrijpt wat Cross-Site Request Forgery (CSRF) is en waarom het gevaarlijk is.
- Je kunt uitleggen hoe een CSRF-aanval werkt met een voorbeeld.
- Je leert hoe je webapplicaties kunt beschermen tegen CSRF.

□□ Uitleg

Wat is CSRF?

Cross-Site Request Forgery (CSRF) is een aanval waarbij een gebruiker wordt misleid om een actie uit te voeren op een website waarop hij is ingelogd, zonder dat hij het weet.

Bijvoorbeeld: je bent ingelogd op je bankwebsite in tabblad 1. In tabblad 2 bezoek je een foute website die automatisch een formulier verstuurt waarmee geld wordt overgemaakt vanaf jouw bankaccount.

Voorbeeld

```

```

Als jij op dat moment bent ingelogd bij de bank, wordt dit verzoek uitgevoerd zonder dat je het merkt!

□□ Opdracht – Begrijp een CSRF-aanval

1. Maak een simpel PHP-bestand `bank.php` dat zogenaamd een betaling uitvoert:

```
<?php
if ($_GET['naar'] && $_GET['bedrag']) {
    echo "Overgemaakt naar: " . $_GET['naar'] . " voor €" . $_GET['bedrag'];
} else {
    echo "Geen betaling uitgevoerd.";
}
?>
```

2. Maak nu een tweede HTML-bestand `aanval.html` dat de aanval uitvoert:

```

```

3. Open `bank.php` in een tabblad en daarna `aanval.html` in een ander tabblad. Wat zie je?

☐ Vervolgopdracht – Bescherm je formulie

1. Gebruik een **CSRF-token** in je formulier:

```
session_start();
if (!isset($_SESSION['csrf_token'])) {
    $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
}
$token = $_SESSION['csrf_token'];
?>
<form method="post" action="verwerk.php">
    <input type="hidden" name="csrf_token" value="<?php echo $token; ?>">
    <input name="naar">
    <input name="bedrag">
    <input type="submit">
</form>
```

1. Controleer in `verwerk.php` of het token klopt:

```
session_start();
if ($_POST['csrf_token'] !== $_SESSION['csrf_token']) {
    die("CSRF aanval gedetecteerd!");
}
```

☐☐ Reflectie

- Waarom is CSRF gevaarlijker dan je op het eerste gezicht denkt?
- Wat is het verschil tussen een CSRF-aanval en een XSS-aanval?
- Waarom werkt een CSRF-aanval alleen als je al bent ingelogd?
- Hoe kun je in een groot project zorgen dat alle formulieren automatisch CSRF-bescherming krijgen?

☐☐ Inleveren

- Screenshot van de onveilige aanval (aanval.html).
- Screenshot van de beschermde versie met token.
- Toelichting bij je reflectie in PDF.
- Bestandsnamen:
 - csrf-aanval.jpg
 - csrf-beschermd.jpg
 - csrf-reflectie.pdf

Revision #13

Created 22 June 2025 08:11:57 by Max

Updated 3 July 2025 12:36:24 by Max