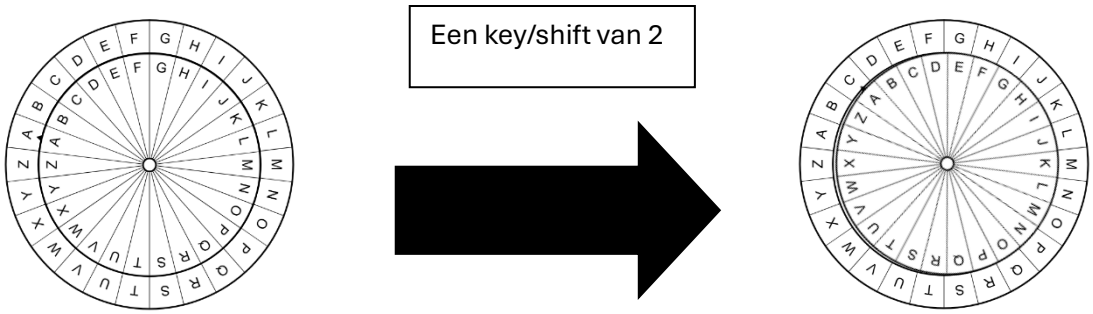
Voorwaardelijke opdrachten overzicht

Er zijn 5 à 6 voorwaardelijke opdrachten die gemaakt moeten worden voor het vak van Cyber security en wetgeving. Deze opdrachten zijn bedoeld om ervaring te geven op het gebied van programmeren met cyber veiligheid in gedachten. Een aantal opdrachten worden systemen voorbereid om gebruik van te maken, andere opdrachten kunnen van scratch gemaakt worden. De opdrachten krijgen een C# template maar mogen gemaakt worden in een andere gangbare programmeertaal voor backend development (Java, C++, Python, C#, Javascript, TypeScript en Node.js zijn voorbeelden van gangbare talen) Communiceer dit met de docent indien je een onbekendere taal gaat gebruiken.

Opdracht 1: Caesar cipher

Het caesar cipher is een oud romeinse encryptie methodiek die het toeliet om een bericht te vervangen met een onleesbare tekst die ontcijferd kan worden. Het coderen en decoderen wordt gedaan aan een Key. Bij een caesar cipher is deze key een cijfer tussen de 1 en 25.

De intentie is dat een het geencodeerde alfabet verschoven wordt met de key waarde en dat het bericht herschreven wordt met dat verschoven alfabet. Hier wordt vaak een set van 2 ringen voor gebruikt.



Voor enccoderen is de binnenste ring het originele bericht en de buitenste ring de encryptie en lees je dus van binnen naar buiten. Voor decoderen is lees je van buiten naar binnen.

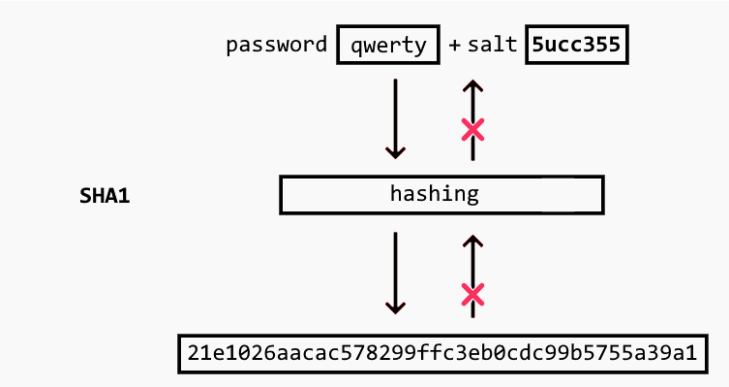
Het doel van de opdracht is om in een console applicatie te maken waarin je een bericht kan encoderen en decoderen aan de hand van een ingevoerde key.

Stappen:

1. Maak een console applicatie die vraagt of je wil encoderen of decoderen
2. Vraag om de key (hou rekening met valse input zoals letters, en getallen onder de 0 en boven de 26)
3. Vraag om het bericht of het geencodeerde bericht.
4. Maak het geencodeerde of gedecodeerde bericht
5. Print het nieuwe bericht uit.

Opdracht 2: Hashing

Het hashen van een wachtwoord is een industrie standaard waarbij je aan de hand van een keyword/salt en je wachtwoord een complex algoritme doorloopt om een random lijkende string terugkrijgt. Hashing is een 1 directionele encryptie, wat betekent dat ook al weet je de keyword/salt dan kan je nog steeds niet het terugdraaien. Dit komt door een complexe wiskundig algoritme dat er voor zorgt dat er meerdere inputs naar hetzelfde hashed wachtwoord zouden leiden.



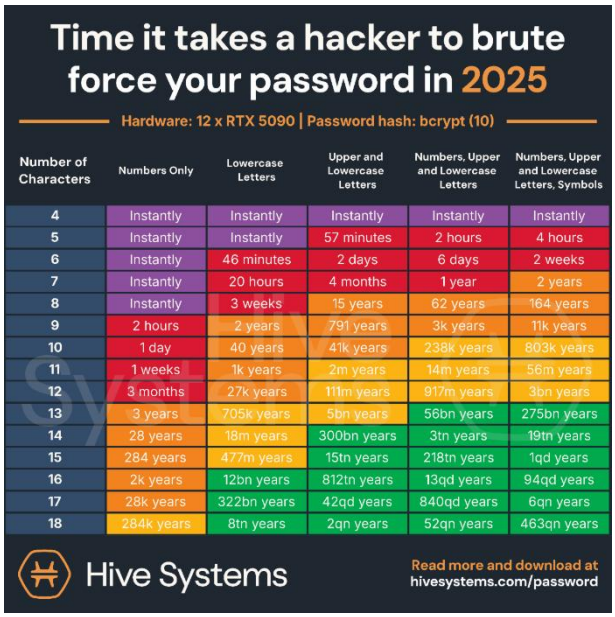
In deze opdracht ga je **NIET** een eigen hashing algoritme schrijven maar wel gebruik maken van een, namelijk SHA. Je gaat een console applicatie maken die in een loop werkt. De gebruiker moet kunnen inloggen en een nieuw account kunnen maken.

Hoewel het niet verstandig is voor bedrijfsapplicaties om de wachtwoorden lokaal op te slaan, laten we het nu toe voor deze opdracht.

Stappen:

1. Maak een console applicatie die met een loop de gebruiker vraagt of deze wil inloggen, aanmelden of stoppen.
2. Maak een string variabele salt die globaal gebruikt wordt
3. Indien de gebruiker wil aanmelden wordt er een naam en wachtwoord gevraagd.
    a. Maak een functie die deze twee variabelen mee neemt
    b. Encrypt het wachtwoord aan de hand van de salt en SHA
    c. Eindig de functie
    d. print de hashed wachtwoord uit als “Server check: ”+ hashedWachtwoord
    e. Sla de naam en het gehashed wachtwoord op in een setje lokale variabelen
    f. Einde van het programma, de loop gaat opnieuw
4. Indien de gebruiker wil inloggen:
    a. Controleer of er een gebruiker lokaal staat opgeslagen, zo niet geef dit aan en eindig deze ronde van de loop
    b. Vraag om de naam en het wachtwoord
    c. Hash het wachtwoord met een functie, maak gebruik van dezelfde salt en encryptie als 3.2
    d. Vergelijk de naam en wachtwoord met elkaar
    e. Als het klopt print “Inlog succes” zo niet “Verkeerd wachtwoord of gebruikersnaam”
5. Indien de gebruiker wil stoppen, dan break je de loop af en eindigt het gehele programma.

Opdracht 3: Password checker



Wachtwoord beveiliging is een belangrijk begrip binnen de ICT wereld. Daarom is het belangrijk dat je begrijpt hoe je een wachtwoord kan beoordelen zodat de kans klein is dat een gebruiker van je systeem gehackt wordt.

Hier zijn de eisen van de password checker:

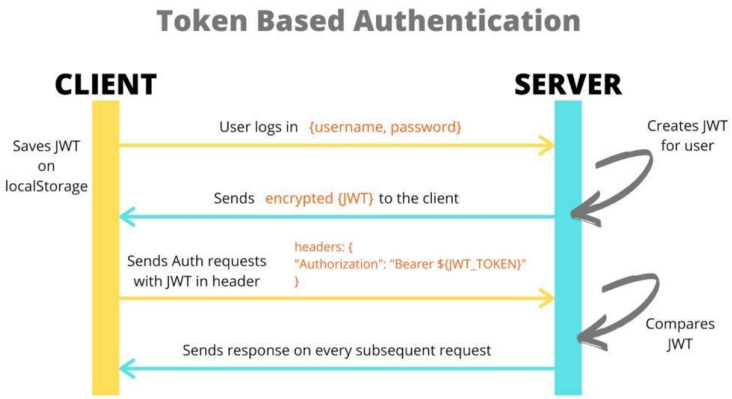
1. De console applicatie moet je vragen om een wachtwoord
2. Hierna controleert de applicatie of het aan jouw eisen voldoet
    - Controleer op de lengte van je wachtwoord, te kort is een probleem
    - Controleer op de gebruikte tekens, is het alleen cijfers of letters? Of meer?
    - Controleer op hoofdletters
    - Controleer op vaak gebruikte wachtwoorden (Deze kan je zelf schrijven of gebruik maken van een die op google/GitHub staat)
3. Is het wachtwoord falend, geef dit dan aan en waarom
4. Is het wachtwoord goed, geef dit aan in een bericht.

Entrophy of randomness wordt niet gevraagd want dit vereist complexe algoritmes. Data breaches wordt zeker niet gevraagd aangezien dit vaak betaalde datasets zijn om gebruik van te maken

Voor de extra opdracht kan je bezig gaan met een brute force hack, hierbij ga je een set wachtwoorden opstellen van oplopende complexiteit en lengte en ga je een brute force algoritme schrijven om de wachtwoorden te breken en daarvan de tijd bij te houden.

Opdracht 4: JWT

JWT of JSON web Token is een manier om bij te houden of een gebruiker nog ingelogd is. Na het inloggen stuurt de server een JWT naar de gebruiker. Dit zijn unieke tokens die lokaal opgeslagen worden in de browser of applicatie als een cookie. Deze wordt dan bij iedere vervolginteractie met de software gecontroleerd of meegestuurd zodat de server weet of de persoon is wie die zegt dat hij/zij is.



JWT zijn zowel persoonsgebonden als tijdgebonden. Sommige programma's laten een eindeloze JWT toe maar meeste beveiligde programma's laten een JWT toe van een dag of in sommige gevallen van 15 minuten.

In deze opdracht ga je nabootsen hoe dit zou kunnen werken. Je maakt een simpele inlog programma, gebruikt de voorbeeldcode die op de GitHub staat of gebruikt het resultaat van opdracht 2.

Stappen:

1. Laat de gebruiker inloggen
2. Bij een foutieve inlog print je uit dat het niet klopt en eindig je de loop/ronde van de loop/programma
3. Bij een correcte stuur je een aangemaakte JWT naar de gebruiker (Deze sla je lokaal op voor deze opdracht)
    a. De JWT moet 1 minuut geldig zijn.
4. Start een loop waarin de gebruiker een actie kan uitvoeren, dit mag een print command zijn met een timer/Wait functie erin.
    a. De JWT moet bij iedere interactie gecontroleerd worden of deze nog geldig is.
5. Zodra de JWT niet meer geldig is krijgt de gebruiker hierover bericht en stopt de applicatie.

Opdracht 5: Database injection

Database injection is een methode waardoor gegevens in en uit een database gehaald kunnen worden of zelfs gedeletet. Vele Websites en systemen zijn hier tegen beschermt en worden gewaarschuwd tegen pogingen maar sommige systemen zijn dat niet. In deze opdracht kunnen jullie gebruik maken van een mock systeem dat niet beveiligd is. het is een games library search applicatie met daarin de wat zwakheden. Speel er eerst mee rond en kijk wat je kan breken door SQLlite injections toe te passen.

Om de opdracht te voltooien moet je de injections kunnen opvangen. Dit kan je op meerdere manieren doen, welke dat is mag je zelf bepalen.

Het project is beschikbaar en te vinden op de GitHub.

Voor eventuele zelfstudie probeer deze course ook te doorlopen:

https://tryhackme.com/room/sqlilab

Opdracht 6: Wireshark, netwerken afluisteren (Nog in overleg, nog **NIET** maken!!!)

Wireshark is een netwerk afluister programma dat volledig legaal is. Wireshark laat je inzien welke berichten en de inhoud ervan gestuurd worden over een netwerk. De reden dat dit legaal is, is omdat het niks anders doet dan het visualiseren van data dat al rondgestuurd wordt over de router. Het grote verschil tussen een legaal en illegaal afluister programma is dat Wireshark geen decryptie uitvoert op de berichten die gestuurd worden.

Probeer aan de hand van Wireshark en een google search

- Zorg ervoor dat je interface op wifi staat anders pakt wireshark hem niet
- Vervolgens in de filter zet je: dns.qry.name contains “google”
  Dit zorgt ervoor dat je alleen google ziet en niet alle andere berichten.
- Google een onderwerp alvast zodat deze klaar staat
- Klik op het haaienvinnetje om te starten
- Druk snel op refresh op je browser
- Druk dan in wireshark op de stop knop.
- Als het goed is zie je twee berichten.
  De eerste is je verzoek, de tweede is je resultaat
- Onderin kan je zien hoe de berichten zijn opgesteld. De berichten zijn geencrypteerd maar je kan wel zien dat er berichten verstuurd worden.