SMP Opdracht

Inleiding

Als cyber security specialist moet je allerlei verschillende soorten gegevens kunnen doorzoeken. Dat kan zijn omdat je wordt gevraagd de veiligheid van de huidige infrastructuur te onderzoeken, te monitoren of wellicht om de impact van een gedetecteerde security breach te achterhalen. Er is geen beginnen aan om dit met de hand te doen, daarom gebruiken we hier een veelvoud aan tools voor waar jullie ondertussen al een klein beetje ervaring mee hebben opgedaan. Maar soms kan je met de bestaande tooling niet alles doen wat je zou willen doen, dus is het belangrijk dat in staat bent zelf een programma te maken dat deze taak kan doen.

Voor deze opdracht ga je met je gemaakte programma een dataset analyseren op potentieel interessante informatie. Het programma levert vervolgens een rapportage op basis van deze analyse.

De dataset bestaat uit opgevangen netwerk-traffic. Dit is traffic naar een webserver van een universiteit, en de dataset bevat de pakketjes die je daarvan mag verwachten. Sommige hosts zijn verbonden met het universiteitsnetwerk, en zitten daarom in hetzelfde subnet als de webserver, terwijl andere hosts vanaf een andere locatie de website proberen te bezoeken. Hoewel de meeste hosts op de juiste manier communiceren met de webserver, zijn er ook hosts in deze dataset die minder nobele doelen hebben: een aantal proberen op de webserver in te breken, deze te overspoelen met verzoeken, of op een andere manier de webserver uit de lucht te krijgen.

Zoals je hebt geleerd tijdens Infrastructure werkt een webserver met het client-server model: de webserver wacht af totdat een client een webpagina opvraagt. Bij het eerste contact legt de client verbinding via de TCP handshake. Vervolgens wordt er een verzoek gestuurd naar de webserver, en stuurt de webserver één of meerdere berichten terug met de opgevraagde webpagina. Aan het einde sluit de client de verbinding weer af. De hele communicatie tussen de webserver en de client, vanaf het moment dat er een TCP handshake plaatsvindt tot het moment dat de client de verbinding weer sluit, wordt gezien als één communicatie-sessie tussen de client en de server.

Omschrijving

Dataset Analyse

Begin de analyse met een verkennende blik op de dataset. Kijk naar de gegevens die je hebt, en denk na over de vragen die je daarover kan stellen.Eerst moet je zelf een aantal vragen bedenken die je aan deze dataset wilt gaan vragen. Vervolgens ga je bedenken welke gegevens je uit de dataset nodig hebt om deze

vragen te beantwoorden. Pas wanneer je weet wat je nodig hebt, en voor welke vraag, ga je de informatie uitlezen uit het JSON-bestand. Dit bestand bevat 3.000 opgevangen pakketjes tussen de webserver en clients. In deze pakketjes zit meer informatie dan je nodig hebt. Elk pakketje bestaat uit een eigen `dictionary`, met daarin informatie over het pakketje op de vijf lagen van het TCP/IP protocol stack. Omdat je niet alle informatie nodig hebt, zal je eerst de relevante gegevens moeten uitsorteren en organiseren. De analyses die jouw programma gaat maken documenteer je in de readme.

Zo zou je op basis van deze dataset kunnen onderzoeken:

- Hoe lang duurt de gemiddelde communicatie-sessie met een host;
- Welke protocollen worden er gebruikt;
- Hoeveel verschillende hosts communiceren er met de webserver;
- Welke hosts communiceren het meest/minst met de webserver;
- Wanneer de meeste/minste requests worden gedaan;
- Welke berichten zijn onderdeel van een aanval;
- Zitten er verschillen tussen hosts vanuit dit netwerk/van buiten dit netwerk in:
  - Welk protocol het meest wordt gebruikt;
  - Hoe laat er wordt gecommuniceerd;
  - Hoe vaak berichten opnieuw moeten worden verstuurd (TCP retransmit);

Het programma levert vervolgens een rapportage op die een duidelijk inzicht/antwoord geeft op de vooraf gestelde vragen.

Commandline Interface

Het gemaakte programma moet natuurlijk in ieder geval alle vragen kunnen beantwoorden. Maar het moet ook mogelijk zijn om specifiek aan te geven welke analyse uitgevoerd moet worden. De gemaakte tool moet ook makkelijk te herbruiken zijn door niet alleen andere personen, maar zou ook potentieel te integreren moeten kunnen zijn in een grotere automatisering.

Om dit te bereiken maak je een commandline interface voor jouw programma dat als argument het path naar het databestand meeneemt.

```
smp-analyzer.py dataset.json
```

Ook moeten als opties kunnen worden meegegeven welke analyses worden uitgevoerd. Dat zou bijvoorbeeld zo kunnen:

```
smp-analyzer.py --list-protocols dataset.json
```

Documenteer de verschillende mogelijkheden van jouw programma.

Hiervoor gebruik je in principe de argparse module uit de standard library. Alternatieven gebruik je alleen als je goed kan onderbouwen waarom.

Automated Testing

Voor programma's die we langere tijd willen gebruiken, en dus moeten onderhouden en wellicht willen uitbreiden, is het belangrijk dat we erop kunnen vertrouwen dat het programma werkt zoals we willen dat het werkt. Handmatig testen kan hierin in helpen, maar hoe vaker we moeten testen hoe handiger het is als dit automatisch kan.

Om software goed te kunnen test moet duidelijk zijn wat de specificaties zijn. Wat verwacht je dat het programma doet, wat verwacht je dat het programma juist niet doet. Hier heb je bij stap 1, het analyseren, als het goed is over nagedacht en dit gedocumenteerd. De tests die je gaat maken testen altijd zoveel mogelijk één aspect van de specificaties. Tests zijn dus klein en je hebt meerdere nodig om één functionaliteit goed te kunnen testen.

Gebruik voor het schrijven van de testsuite pytest.

Conventies

Code houdt zich aan de pep8 styling guide:

- https://peps.python.org/pep-0008/
- https://pep8.org/
- maximum line length: 99

Voeg code comments toe ter verduidelijking van programmeer keuzes en ter documentatie van classes en functies. Zie ook pep8.

Houdt de volgende folder structuur aan:

```
/
 - docs/
    - *.md
 - tool-naam/
    - *.py
 - tests/
    - test_*.py
 - readme.md
```

Zie ook: https://docs.python-guide.org/writing/structure/
note: rst (reStructuredText) is een alternatief voor md (markdown)

docs folder is optioneel en alleen nodig voor uitgebreidere documentatie dan wat al in de readme past.

Eisen

- Je hebt tenminste 3 vragen voor jouw programma bedacht en deze door je docent laten goedkeuren.

- De verschillende analyses die het programma maakt zijn duidelijk gedocumenteerd.
- Voor tenminste één feature zijn automatische tests aanwezig met zowel positieve als negatieve tests.
- Het programma is te gebruiken als cli tool. Hoe de tool te gebruiken is gedocumenteerd.
- De code voldoet aan de codeconventies.

Beoordeling

Als er twijfel over de authenticiteit van het werk bestaat, kan je worden uitgenodigd voor een gesprek met de docent ter verificatie. Een soort assessment dus.

cijfer = punten / 2

Criteria	Onvoldoende (1)	Matig* (2)	Voldoende (3)	Goed (4)	Uitstekend (5)
Probleemstelling Analyse & Specificaties	Afwezig / Slecht	Onvolledig	Simpel, maar volledig	Uitgebreid	Complex
Kwaliteit	Programma werkt niet	Slordig	Hanteert afgesproken codering standaarden	Goed te onderhouden code (leesbaar)	Uitstekend te onderhouden (modulair)
Complixiteit programma	Voldoet niet aan minimale eisen	nvt	Minimaal 1 class, cli opties aanwezig	Uitgebreide configuratie mogelijkheden en/of meerdere logische classes	Gebruikt geavanceerdere python concepten waar toepasselijk
Testsuite	Afwezig	Matige testset	Redelijke testset voor één functionaliteit	Testsets voor meerdere functionaliteiten of zeer goede testset voor één functionaliteit	Compleet

**maximaal 1 onderdeel mag matig zijn.*