Mini Project 3

Posted: *2019-03-29* | ***Due:*** *2019-04-29*

Applications of Factor Graphs in Security

Changelog

This page will be continuously updated with more information.
Read the change log below carefully and revise your answers accordingly. A list of changes so far include:

- 04/27/2019 - Fixed Typo in number of events in Task4.0 from 5 to 6.
- 04/25/2019 - Added Report Template for final submission
- 04/18/2019 - Updated Task 3 DNS Tunneling factor funtions (f_7, f_8, f_9) q values.
- 04/18/2019 - Clarified Task 3.5 description
- 04/17/2019 - Task 3 and Task 4 released.
- 04/16/2019 - Made notation for events and states consistent with factor graph image in Task 3
- 04/15/2019 - Updated Task 3 description for factor function r. Updated project deliverables for checkpoint 2.
- 04/10/2019 - Task 2 and Task 3.0 released. Updated project deliverables for checkpoint 2.
- 04/04/2019 - Added instructions to access course Jupyter Notebook Server
- 04/01/2019 - Clarify questions 2 and 3 in Task 1: Host Logs Analysis (Attack Stages 3-5)
- 03/29/2019 - Initial release. Tasks 0 and 1 released.

Quick Links

Introduction

The goal of this project is to understand the progression of a multi-stage attack that aims to leak secret keys from target system. In this project, you will perform analysis of a multi-stage attack recreated from publicly available information on the Equifax breach. You will explore the use of signature-based, anomaly-based, and factor graph-based techniques.

As a defender, you are given monitoring logs to perform the following tasks:

- Compare the attacker's behavior vs. legitimate users' behavior (Task 1)
- Build a simple factor graph for a single event and a single attack stage (Task 2)
- Extend the factor graph to capture the evolution of a series of events (Task 3)
- Run inference on attack stages using the extended factor graph (Task 3)

Please read this entire document and plan ahead before starting with the project.

Project Deliverables

Checkpoint	Deadline	Deliverables
CP1	April 10,2019 (23:59 hrs)	Tasks 0 & 1 as a .ipynb notebook

Checkpoint	Deadline	Deliverables
CP2	April 17,2019 (23:59 hrs)	1. Task 2 .ipynb notebook along with code for Tasks 0 & 1 as .ipynb notebook 2. Task 3.0 Image of Constructed Factor Graph
CP3	April 29,2019 (23:59 hrs)	1. ipynb notebook including all tasks 2. PDF of Powerpoint presentation (template will be attached below)

Project Materials

File	Monitor	Format	Download (Right click, and Save as...)
Datasets	tcpdump	pcap	HTTP pcap, HTTP2 pcap
	tcpdump	pcap	DNS pcap
	osquery	json entries	osqueryd log
Pyshark Tutorial			Intro to pyshark **(Updated Mar 31, 2019)**
Task 2 Template			IPyNB
			Python
Task 3 Template			IPyNB
Report Template			PPTX

Environment Setup for Personal Device

1. Install wireshark from here. You may also install wireshark using your operating systems package manager.
   - For Windows, you should also install 'Build Tools for Visual Studio 2017' from here
2. In a jupyter notebook, execute the following code

```
!pip install pyshark
```

3. Verify that pyshark has been installed using

```
import pyshark
```

Access to Jupyter Notebook Server

To help you focus on solving the mini-project and avoid wasting time trying to get the necessary packages and software working for the project, we have set up a Jupyter Notebook Server that you can use. **Since this is a shared resource, please use it responsibly.** Please note that use of dvorak to complete task 0 & 1 is not a must if you are able to carry out the tasks using your personal devices. All submissions are still on Compass2G.

To get access to the notebook, follow these steps:

1. Make sure you are either on the university network or have a VPN connection to the university network.
2. Visit http://dvorak.csl.illinois.edu:8080
3. Log in using your university email and password. If your university email is not connected to a google account, please request for access for your personal gmail account here: https://forms.gle/6q6KEZ1ku9A1R7kUA. **Do this as soon as possible as granting access may take some time.**
4. For task 0 & 1, click "New" at the upper-right corner, then select "Python 3"

In Python 3, the pre-installed packages are pandas, pyshark, seaborn, and matplotlib.pyplot, as they provide all the necessary functionalities to complete task 0 & 1. All data released on the mp3 webpage is available in /data folder, if you are curious and would like to confirm that they are indeed there, running "!ls /data" in a cell, so if you want to read the data, e.g. http.cap, simply use the following path "/data/http.cap"

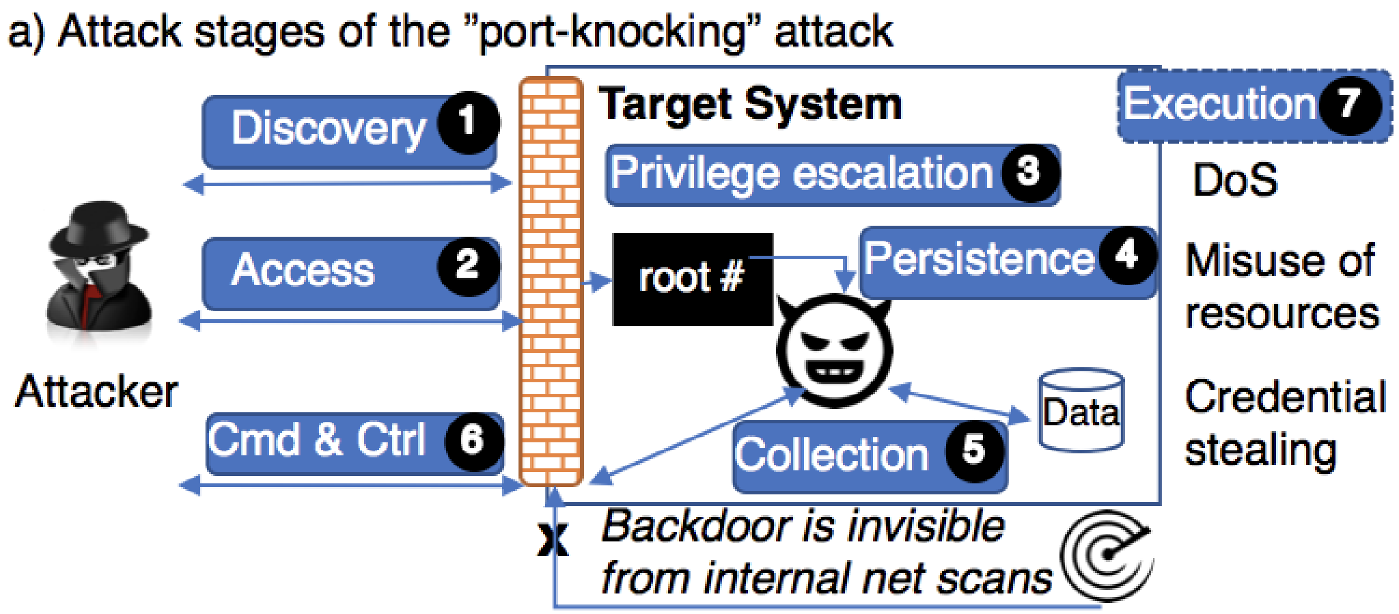
Note that all traffic on this server is heavily monitored. Any attempts to execute malicious code will be logged and will result in a failing grade for the assignment. Please review the Policy on Appropriate Use of Computers and Network Systems at the University of Illinois at Urbana-Champaign for guidelines concerning proper use of information technology at Illinois, as well as the Student Code (especially 1-302 Rules of Conduct, 1-402 Academic Integrity Infractions). As members of the university, you are required to abide by these policies.

Please download and backup your notebook after each session.
Course staff will not be responsible for any data loss if the server goes down.

Attack Description

The attack consists of three main attack vectors:

1. Remote code execution vulnerability in Apache Struts (CVE-2017-5638),
2. Local privilege escalation exploit (CVE-2016-5195), and
3. A kernel module backdoor for stealthy communications



Attack stages, security events, and logs observed by security monitors on a target systems. The actual logs in the provided dataset might be slightly different.

Stage No.	Stage Name	Attacker Action
1	Discovery	Scan for a vulnerable web server
2	Access	Login to the vulnerable server
3	Privilege Escalation	Escalate privilege to the super user
4	Persistence	Install a new system service
5	Collection	Collect a secret key
6	Cmd & Ctrl	Run additional commands
7	Exfiltration	Extract the secret key

Attack stages and attacker's actions performed at each stage.

Attacker Actions

Stages 1-2 The attack starts in the discovery stage (1), where an attacker repetitively scans for vulnerable Apache Struts portals (CVE-2017-5638). These scans do not require immediate action from security operators because the scans have not affected the system. The attack then progresses to the access stage (2) where the attacker obtains a remote shell terminal by exploiting the vulnerability (CVE-2017-5638).

Stages 3-4 After establishing a foothold in the system, the attacker executes commands with superuser permissions (i.e., the privilege escalation stage (3)) as the web server runs with as root. The attacker proceeds to the persistence stage (4), where he/she downloads a file with a sensitive extension, e.g., .c, which is commonly seen in both legitimate users and malicious activities. The source file is used to compile and to install a new kernel module. At this persistence stage (4), the ongoing attack needs an immediate defensive response since the attacker can maintain persistent access to the compromised system using a backdoor which provides a secret communication channel, i.e., DNS tunneling.

Stages 5-7 The attacker then collects secret keys, i.e., Secure Shell keys such as RSA keys in the collection stage (5). Using the backdoor, the attacker sends additional commands in the command & control stage (6) to establish a DNS tunnel. Then, the secret keys are extracted in the exfiltration stage (7).

Project Tasks

Task 0 - Introduction & Data Preprocessing

In this task, you will familiarize yourself with networking and operating system concepts.

Additionally, given the following set of raw input logs across 3 files, you will convert these input datasets to well structured CSV files.
(Please refer to the pyshark tutorial attached above if you are unfamilliar with data processing with pyshark)

- HTTP network packet captures over the period of the recreated attack (~5 min), and
- DNS network packet captures over the period of the recreated attack (~1 min), and
- OSQuery Logs containing linux kernel (insmod/rmmod module functions) and filesystem (open/close/read/write) activity obtained from monitoring select system calls on the server being attacked over the period of the recreated attack.

1. Familiarize yourself with the following concepts:

- Networking protocols - TCP/IP, UDP, DNS, HTTP, HTTPS, SSH
- Network Paths, HTTP GET, POST requests, HTTPS SSL/TLS
- Operating Systems - Filesystem layout, system logging mechanisms
- Security - Priveledge escalation, root-kits, remote code execution, Intrusion Detection System (IDS)

2. Import the DNS and HTTP pcap files into your notebook using pyshark.FileCapture.
   - Create new pandas dataframes for DNS and HTTP pcaps
   - For each pcap, extract relevant information from network packets into their dataframe.
   - At minimum, you should store sniff time, length, and highest layer for each packet.
   - What else? Take a look at Task 1 to determine what fields you need to perform your analysis.
3. The provided *osqueryd.results.log* file contains a JSON dictionary log entry on each line.
   - Convert the information contained in this file into a dataframe.
   - You may flatten nested dictionaries by prepending the parent key to each child key.
   For example:

```
dict1 = {"p": {"a": 1, "b": 2, "c": 3}}
```

   may be represented as columns “p_a”, “p_b”, “p_c” in your dataframe. *Hint: eval() will convert a string to a dictionary HintHint: You may want to look at the json_normalize method in the pandas.io.json package*
4. Export your created pandas dataframes as CSV files.
5. Take a look at http.pcap and http2.pcap, one is legitimate network activity, the other is attacker network activity. According to the description of CVE-2017-5638, the attacker network activity contains a “#cmd” string in Content-Type header (Ref: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-5638). Use this clue, identify which http pcap file represents legitimate activity, and which represents attacker activity. What is the Content-Type in legitimate activity pcap file?
   Use the attacker http pcap file for the following tasks.

Task 1 - Data Exploration/Forensics Prior To Building a Machine Learning Model

Security analysts and attackers both frequently study network traffic to search for vulnerabilities and to characterize network behaviors. In this section, you will examine network traces from a sample network we set up for this assignment to compare the attacker’s behavior vs. legitimate users’ behavior. While the analyses in this task may not be directly relevant in building the factor graph in Tasks 2 and 3, it provides the necessary background and insights in building factor graphs and interpreting inference results. Remember that this is an end-to-end project which includes data analytics of a security application and building a detection model.

HTTP Traffic Analysis (Attack Stages 1-2)

1. CVE-2017-5638 (related to this attack) discloses that a scan is initiated by querying the */showcase.action* page. A vulnerable server responds to this scan with a response code (HTTP/1.1 200 OK) and following HTTP header:

```
Server: Apache-Coyote/1.1
<some payload>
```

   - **SubTask 1.a** Report the UNIX timestamp of the first attempted scan on the vulnerable server
   - **SubTask 1.b** Identify the IP address of the vulnerable server
   - **SubTask 1.c** Identify the port of the vulnerable server

2. The attacker crafts malicious HTTP requests to access and exploit the Apache Struts web server. Such requests have abnormally long values in their *Content-Type* headers because the attacker embeds malicious commands in such headers. Such commands allow attacker to complete later stages of the attack.

   - **SubTask 2.a** Extract a list of the *Content-Type* headers sent to the vulnerable server from the provided HTTP packet capture. For each *Content-Type* header, provide its length as well.
   - **SubTask 2.b** Fill in the blanks in the table below. You may want to refer to UNIX man pages or this website. Refer to an example answer for the whoami command. Identify in the table below the UNIX commands that are present in the extracted *Content-Type* headers above. Not all commands in the table are present in the attacks. Interpret the utility of the identified commands, i.e., how such commands help an attacker to achieve their objective. Only provide the interpretation of the command if the command is present in the attack.

Command Name	Present in the attack?	Interpretaion of the command
whoami	*Yes*	*Displays the name of the current user*
wget		
ls		
cat		
cd		
insmod		

Command Name	Present in the attack?	Interpretaion of the command
ssh		
lsmod		

Host Logs Analysis (Attack Stages 3-5)

Please download osqueryd.results.log from the Project materials above

Since the server software runs as root, any commands embedded in GET requests by an attacker will execute with superuser/root privileges *(Stage 3: Privilege Escalation)*. After exploiting the Apache Struts server, the attacker installs a rootkit as a new kernel module *(Stage 4)*, and accesses the folder (.ssh) containing the secret key (id_rsa) and the list of internal hosts on the vulnerable server) *(Stage 5)*.

Using the osquery dataframe that you created in Task 0, perform the following subtasks:

1. Analyze kernel-related activities *(Stage 4: Persistence)*
   - Provide a list of kernel modules added or removed from the system
   - Identify the attacker-controlled kernel module. *Hint: The kernel module (*.ko) was downloaded via a GET request that you extracted in Task 1.2*
   - Verify that the kernel module that the attacker obtained in Task 1.2 has been loaded into the vulnerable server. How did you verify that the module was loaded onto the server?
2. The server contains a list of internal hostnames that the attacker can use for lateral movement. *(Stage 5: Collection)* What is the file name that contains the internal hostnames?
3. A naive attacker could extract the list of internal hostnames via HTTP to an attacker-controlled server. Do you observe any evidence that the attacker extracted the internal host names via HTTP in the logs?

DNS Traffic Analysis (Attack Stages 6-7)

After reading the secret key, the attacker establishes a DNS tunnel to a remote server *(Stage 6: Command and Control)*, and uploads this secret key to the remote server to avoid detection by traditional Intrusion Detection Systems *(Stage 7: Ex-filtration)*.

Using your DNS dataframe, perform the following tasks:

1. There are two DNS servers accepting DNS queries in the provided network trace.
   - Identify the attacker-controlled DNS server

- Identify the legitimate DNS server

  Hint: The attacker-controlled DNS server is behind the same network that hosts the kernel module in the HTTP task (Task 1.2).

2. Plot a **histogram** of the length of DNS queries for the two DNS servers on the same plot. Clearly label the two servers using different colors/styles. **Use log scale for the count on the Y axis.**

Task 2 Setup

This task requires the *opengm* library that is only available on Python 2 and Linux. We have set up a restricted notebook server on http://dvorak.csl.illinois.edu:8080 to execute Python 2 + OpenGM code.

Jupyter notebook server instructions:

- Upload and execute the Task 2 template notebook provided in the Project Materials table above.

Note that all traffic on this server is heavily monitored. Any attempts to execute malicious code will be logged and will result in a failing grade for the assignment.

Please review the Policy on Appropriate Use of Computers and Network Systems at the University of Illinois at Urbana-Champaign for guidelines concerning proper use of information technology at Illinois, as well as the Student Code (especially 1-302 Rules of Conduct, 1-402 Academic Integrity Infractions). As members of the university, you are required to abide by these policies.

Please download and backup your notebook after each session. Course staff will not be responsible for any data loss if the server goes down.

Personal Computer Setup Instructions:

If you are running a version of Linux that ships with Python 2, you may set up the required libraries using the instructions below. Note that you should not execute *opengm* code using your Jupyter setup, and use standard command line Python instead.

- Verify that you can run Python 2 using `python --version` or `python2 --version`. Running the command should output `Python 2.7.X`, and not `Python 3.6.X :: Anaconda, Inc.`.
- Install the following packages using your package manager:
  `libopengm-bin, libopengm-bin-dbgsym, libopengm-dev, libopengm-doc, python-opengm, python-opengm-doc`

On Ubuntu, you may do so by running:

```
sudo apt-get update
sudo apt-get install libopengm-bin libopengm-bin-dbgsym libopengm-dev libopengm-doc python-opengm
```

- Install other dependencies using pip:

```
pip install numpy scipy seaborn -U
```

  Depending on your default Python version, you may have to replace `pip` with `pip2` in the command above.

- Run Python on the command line. In the Python interpreter, execute:

```
import opengm
```

If the command succeeds without an `ImportError` or `ModuleNotFoundError`, you have successfully set up *opengm* on your machine. You may now execute the Python 2 version of the Task 2 template provided in the project materials.

Task 2 - Introduction to Programming Factor Graphs

In this task, you will construct a simple factor graph using one event (E_1 - *Scan*) observed during Stage 1 in the attack scenario above. Your task is to infer the hidden attack stage S_1 using the provided factor functions.

Before beginning with this task, familiarize yourself with the following concepts:

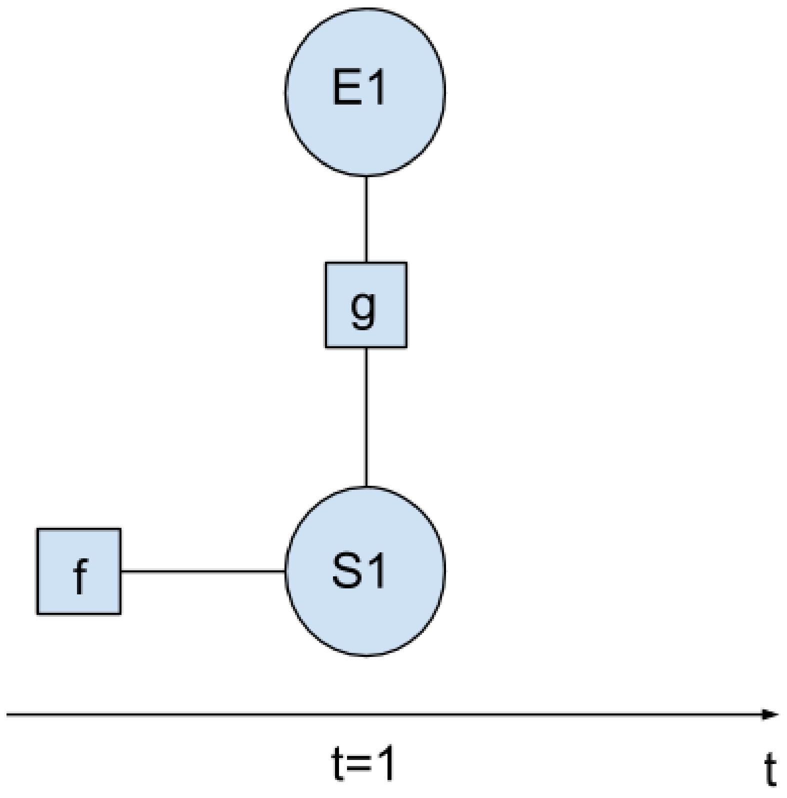
- Probabilistic Graphical Models - Factor Graphs, Factor Functions
- Inference on Graphical Models - Brute-force, Message Passing

At time step *t=1*, a security event E_1 (Scan) is observed and the corresponding stage S_1 is unknown to be a real attack or benign activity. In this task, both the event and the attack stages are indicated by binary variables. The event has a value 1 that indicates a scan for vulnerabilities and 0 otherwise. The attack stage has a value 1 that indicates an attack and 0 otherwise.

A simple factor graph of the event E_1 and the attack stage S_1 is shown below. Two factor functions: a prior on the stage $f(S_1)$ and a factor function $g(E_1, S_1)$ are provided as well. The factor graph represents the joint probability distribution:

$$P(E_1, S_1) = \frac{1}{Z} f(S_1) g(E_1, S_1)$$

where $Z = \sum_{E_1 \in 0,1; S_1 \in 0,1} f(S_1)g(E_1, S_1)$ is the normalization factor.



Task 2 Factor Graph

Event	Value	Description
E_1	0 or 1	0: no scan
		1: scanning activity is observed

Stage	Value	Description
S_1	0 or 1	0: no attack
		1: an attack is in progress at the discovery stage

Priors	$f(S_1)$
$S_1 = 0$	0.85

Priors	$f(S_1)$
$S_1 = 1$	0.15

Factor Function	$g(S_1, E_1)$
$S_1 = 0, E_1 = 0$	0
$S_1 = 0, E_1 = 1$	0.2
$S_1 = 1, E_1 = 0$	0
$S_1 = 1, E_1 = 1$	0.5

Using the provided template code:

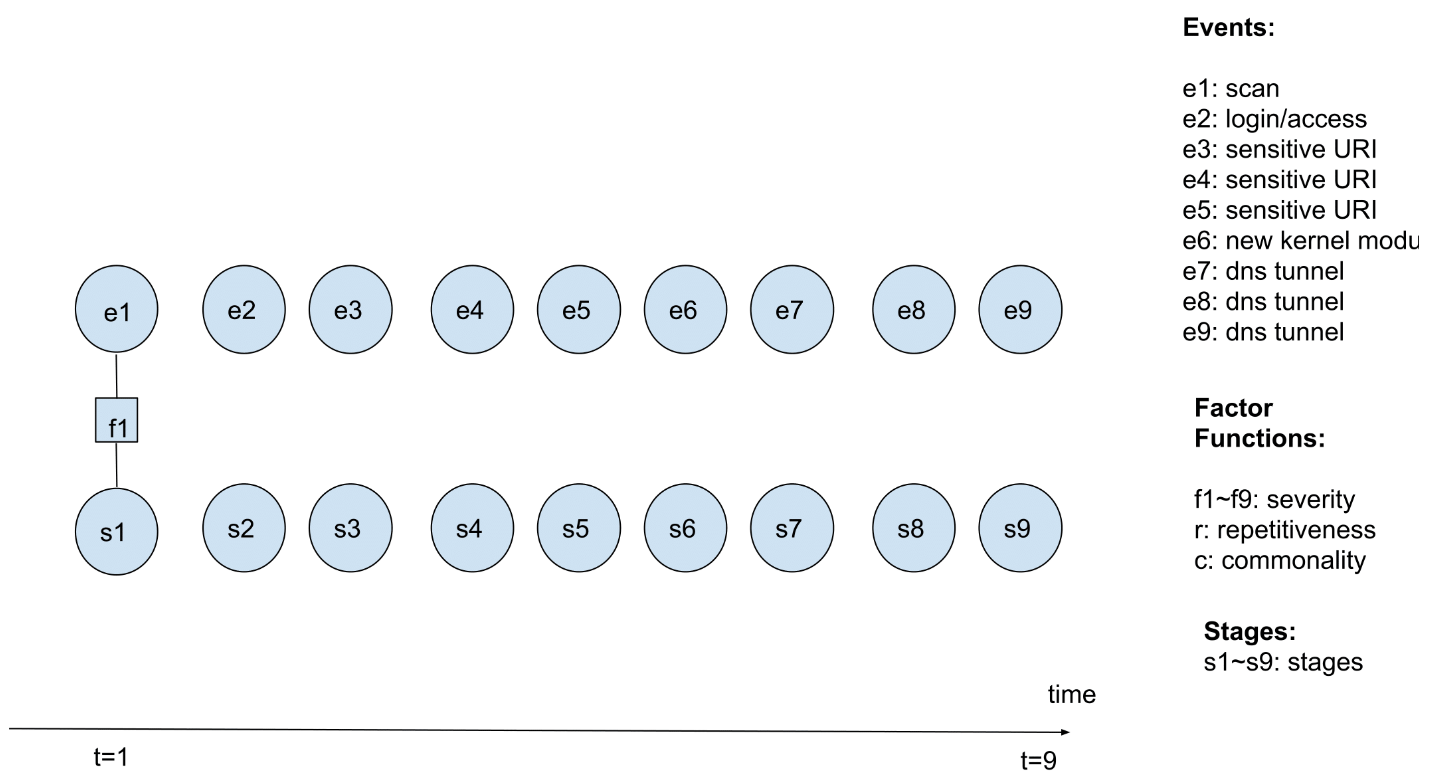
- Build the simple factor graph by completing the code
- Compute the marginal probability $P(S_1)$
- What value of S_1 maximizes the marginal probability $P(S_1)$? Explain.
- Verify your results with hand calculations. Show all steps of your work.

Task 3 - Inference for a factor graph of the Equifax attack scenario

In this task, you will construct a complete factor graph that models the Equifax attack scenario using the events observed in Task 1. The goal is to infer the hidden attack stages using provided factor functions.

The data provided in Task 1 indicates the following 5 unique security events: {Scan, Login, Sensitive URI, New Kernel Module, DNS Tunneling}

While each event by itself may not be an indicator for an attack, these events occurring in a certain sequence over a short period of time may indicate malicious intentions as the attack progresses over time.



Task 3 Factor Graph

In our factor graph model, we assume that the above events take place over 9 time steps, $t = 1$ to $t = 9$ in the following order: Each event at time $t = k$ is represented by event e_k. Each event e_k may take on any of the following values:

- ϵ_1 - Scan
- ϵ_2 - Login
- ϵ_3 - Sensitive URI
- ϵ_4 - New Kernel Module
- ϵ_5 - DNS Tunneling

At each time step from $t = 1$ to $t = 9$, there is an unknown stage (hidden state), i.e. s_1 - s_9. Each one of the hidden states s_i could assume any of the following 11 states at each time step:

- σ_1 - Benign

- σ_2 - Discovery
- σ_3 - Access
- σ_4 - Lateral Movement
- σ_5 - Privilege Escalation
- σ_6 - Persistence
- σ_7 - Defense Evasion
- σ_8 - Collection
- σ_9 - Exfiltration
- σ_{10} - Command and Control
- σ_{11} - Vulnerable Code Execution

We will model our factor graph as follows:

- f_1 to f_9 are severity factor functions corresponding to each event e_1 to e_9.
- c is a commonality factor function which connects events e_1, e_3 and e_6 and infers s_6.
- r is a repetitiveness factor function which connects events e_3, e_4 and e_5, and infers stage s_5.

The goal is to infer the hidden attack stages s_1 to s_9 using your factor graph model.

Probability of past attacks for each event and their significance are given in the following tables:

Event	Factor Functions	Attack States	Significance (p)	Probability in past attacks (q)
Scan	f_1	{ discovery, benign }	{ 0.03, 0.52 }	{ 0.5, 0.5 }
Login	f_2	{ benign }	{ 0.01 }	{ 0.5 }
Sensitive URI	f_3, f_4, f_5	{ privilege escalation, benign }	{ 0.015, 0.015 }	{ 0.09, 0.11 }
New Kernel Module	f_6	{ persistence, benign }	{ 0.005, 0.03 }	{ 0.07, 0.09 }

Event	Factor Functions	Attack States	Significance (p)	Probability in past attacks (q)
DNS Tunneling	f_7, f_8, f_9	{ benign, exfiltration }	{ 0.08, 0.004 }	{ 0.16, 0.18 }

Table of events, stages, q and p values for factor functions $f_1 - f_9$ (severity)

Event Sequence	Factor Function	Attack States	Significance (p)	Probability in past attacks (q)
{Scan, Sensitive URI, New Kernel Module}	c	persistence	0.006	0.15

Table of events, stages, q and p values for factor function c (commonality)

Event Sequence	Factor Function	Attack States	Significance (p)	Probability in past attacks (q)
{Sensitive URI, Sensitive URI, Sensitive URI}	r	privilege escalation	0.05	0.15

Table of events, stages, q and p values for factor function r (repetitiveness)

Probability values for each factor function is specified as an 11 dimensional vector, with each cell representing its corresponding attack stage. The probability of a stage is given by $q \times (1 - p)$. For stages not present in an event, its corresponding probability is 0.

For example, the probability vector for factor function f_1 is

$$[0.5 \times (1 - 0.52), 0.5 \times (1 - 0.03), 0, 0, 0, 0, 0, 0, 0, 0, 0]$$

since only attack states σ_1 (benign) at index 0 and σ_2 (discovery) at index 1 are defined for S_1 (Scan) .

A table containing action probabilities for each stage is given below. Your Factor Graph model should pick the appropriate action for each stage of the attack that it encounters. While you could build complex models using these probability values, simply picking the *argmax* for each stage should suffice for this project.

Stage Name	No-Op Action	Monitor Action	Stop Attack Action
Benign	1.00	0.00	0.00
Discovery	0.61	0.39	0.00
Access	0.69	0.31	0.00
Lateral Movement	0.09	0.84	0.07
Privilege Escalation	0.20	0.63	0.17
Persistence	0.00	0.70	0.30
Defense Evasion	0.00	0.07	0.93
Collection	0.00	0.10	0.90
Exfiltration	0.00	0.00	1.00
Command and Control	0.00	0.00	1.00
Execution	0.00	0.00	1.00

Files: Task 3 Template

- **Subtask 3.0** Draw a factor graph for each time t from $t = 1$ to $t = 9$. You may use the factor graph given above as a starting point. First, draw all observed variables and corresponding unknown variables. Second, draw the provided factor functions. Third, make corresponding connections between the provided factor functions and the observed/unknown variables. Hint: Use the input of each provided factor function to make the corresponding connections. Upload a visual representation of the factor graph you defined in this subtask to Compass2G for **Checkpoint 2**. Remember to label each variable and factor function.
- **Subtask 3.1** Using the Task 3 code template, build a factor graph for each time t from $t = 1$ to $t = 9$. You may define a separate factor graph for each time step, but we strongly recommend you parametrize a general model that accepts a list of events and a time step t.
- **Subtask 3.2** Infer the hidden attack states corresponding to each time point from $t = 1$ to $t = 9$. Provide the marginal probability of each state and the most probable state (using argmax) in your report.
- **Subtask 3.3** What action should your model recommend for each time step? How did you arrive at your decision?

- **Subtask 3.4** What is the earliest stage in which your model should recommend stopping the attack?
- **Subtask 3.5** Consider the factor graph you have used from Task 3.0 to 3.4. We want to investigate how the inference of the hidden attack stages change if we do not observe e_7. Create a new factor graph using the same variables and factor functions used in Task3.0 but do not include e_7 and s_7. Perform the inference on this new factor graph and report the most probable states. Do the most probable states for $s_1 - s_6, s_8, s_9$ remain the same as Task3.2? Why or why not?
- **Subtask 3.6** We will compare using an HMM to model this attack scenario vs using an FG. Note that the stage-action table is not included in the HMM.

Draw an HMM model for the attack scenario given the provided states and events.

What parameters are needed for this HMM model to work?

Give an example of an advantage of the FG over the HMM model?

Task 4 - Another Equifax attack scenario

On October 12, 2017, it was reported that the Equifax website was compromised and was redirecting users to download malware in the form of a fake Flash Update.

We propose a hypothetical attack scenario based on this publicly reported attack. The sequence of attacker's actions in this hypothetical attack scenario is as follows.

Attackers logged in to the Struts server remotely and downloaded a malicious and executable file from attacker's controlled servers. Next, attackers overwrote the web server's home page to prompt users to download the malicious and executable file.

In the HTTP and DNS packet traces and OSQuery logs, system administrators observed the following security events:

- ϵ_1 - Scan
- ϵ_2 - Login
- ϵ_3 - Sensitive URI (i.e., the URI indicates an executable file)
- ϵ_4 - New Executable File
- ϵ_5 - Homepage overwritten with a new link
- ϵ_6 - Webserver restarted

in the following order:

{Scan, Login, Sensitive URI, Sensitive URI, Sensitive URI, New Executable File, Homepage overwritten with a new link, Webserver restarted}

Event	Attack States
Scan	{ discovery, benign }
Login	{ benign }
Sensitive URI	{ privilege escalation, benign }
New Executable File	{ persistence, benign }
Homepage overwritten with a new link	{ command and control, execution, benign }
Webserver restarted	{ command and control, execution, benign }

Subtask 4.0 Is it possible to *accurately* detect this attack using only one, e.g., ϵ_1, of the six listed events? For each of the six listed events, give an example of a situation in which a false positive (i.e., mis-detecting a legitimate user as an attacker) could happen.

Subtask 4.1 Provide a visual representation of a factor graph that can model the attack described above. This can be hand drawn.

Subtask 4.2 What variables and factor functions are common to the factor graph in Task 3 and your factor graph in 4.1? *Hint: Are any events common to both factor graphs? Repeating?*