Институт
Компьютерных
Технологий и
Информационной
Безопасности

Лабораторная работа №3

«Построение простейших графов атак»

по дисциплине

«Технологии и методы программирования в задачах защиты информации»

Разработал:
ассистент кафедры БИТ ЮФУ
Алексеев Дмитрий Михайлович

г. Таганрог, 2021

1. Цели и задачи работы

Графы атак являются одним из наиболее распространенным методом моделирования и анализа атак. Причиной этого является относительная простота построения таких графов, а соответственно высокая скорость моделирования. При этом, графы атак дают достаточно полную картину развития атаки и могут быть легко интерпретированы администратором или специалистом по безопасности. Существует значительное количество различных моделей атак.

Целью данной работы является практическое подкрепление теоретических знаний в области работы моделирования атак. В лабораторной работе необходимо разработать программу моделирования простейшего графа атак и визуализировать полученный граф.

2. Графы атак

2.1 Общие сведения

Академические исследования графов атак начались в середине 1990-х годов. Причиной интереса к графам атак является тот факт, что моделирование атак позволяет проводить верификацию безопасности системы без фактической проверки сетевых узлов и программного обеспечения, что позволяет существенно (в сотни раз) ускорить процесс анализа атаки, а также проверить предположения о различных контр-мерах в случае различного развития атак.

В настоящее время существует значительное количество моделей графов атак, как академических, так и применяемых в продукционных коммерческих системах анализа безопасности.

Определения: *граф атак* – это граф, представляющий всевозможные последовательности действий нарушителя для достижения целей. Целью атаки является реализация некоторой угрозы безопасности. Последовательности действий по достижению цели атаки называются *трассами атак*.

Например на рисунке 1 изображен простейший граф атаки (state enumeration graph) [1].

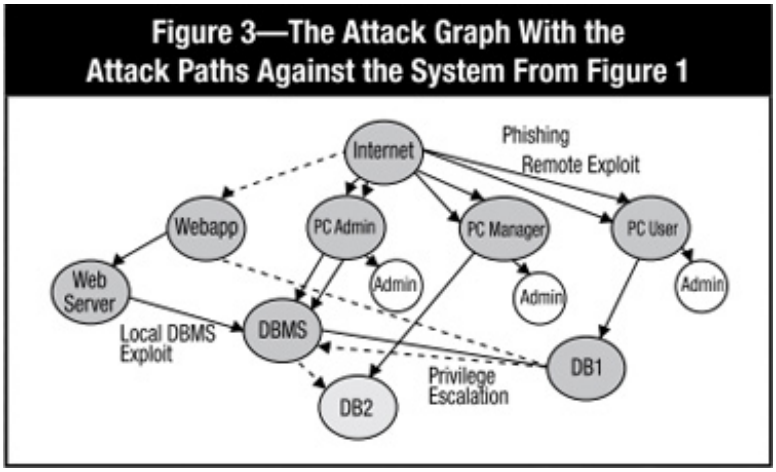

Рисунок 1 - Пример графа атак (по материалам [1])

Выделяют следующие виды графов атак:

State enumeration graph – в таких графах вершинам соответствуют тройки (*s*, *d*, *a*), где *s* – источник атаки, *d* – цель атаки, *a* – элементарная атака; дуги обозначают переходы из одного состояния в другое;

Condition-dependency graph – вершинам соответствуют результаты атак, а дугам – элементарные атаки, приводящие к таким результатам;

Exploit dependency graph – вершины соответствуют результатом атак или элементарным атакам, дуги отображают зависимости между вершинами – условия, необходимые для выполнения атаки и следствие атаки.

Host-oriented graph – вершины соответствуют узлам, а дуги отображают действия (элементарные или комплексные) атакующего по распространению этапа атаки по локальной сети.

Выделяют также понятие этапа атаки. **Этап атаки** - некоторые законченные действия атакующего по достижению цели атаки. Действие может быть явно вредоносным (эксплуатацией уязвимости), так и легитимным действием – например, подключением к некоторому сетевому порту, записью файла и т.д.

2.2. Упрощенный граф атак

Моделируемый в лабораторной работе граф атак является упрощенным графом атак, предложенным в работе [3]. Данный вид графов является host-ориентированным графом, в котором узел отображает сетевой узел, а дуга действие атакующего по продвижению атаки от одного узла к другому. Для понимания полной модели в работе [3] необходимо ознакомиться с соответствующим лекционным материалом.

Для реализации предлагается следующая простая модель:

1) Сетевой узел идентифицируется IP-адресом.
2) Каждый сетевой узел принадлежит только к одной подсети, на границе которой находится только один маршрутизатор.
3) Каждый сетевой узел имеет связи – это логическая связь данного сетевого узла с другими:
   - считаем, что все узлы находящиеся внутри одной подсети имеют связь друг с другом;
   - связи между маршрутизаторами задаются явно;
   - некоторые узлы доступны из внешних подсетей (задается знаком +), некоторые узлы недоступны из внешних подсетей (задается знаком -).
4) Каждый сетевой узел содержит набор уязвимостей.
5) Каждая уязвимость идентифицируется уникальным идентификатором (произвольной тестовой строкой, например – V-012)
6) Каждый уязвимость определяется только одним параметром - *уровень доступа, который предоставляет уязвимость на данном узле root/user/dos/other:*

- *Root – полномочия администратора;*
- *User – полномочия пользователя;*
- *Dos – нарушение работы системы;*
- *Other – другие нарушения целостности и доступности информации.*

7) Требования для срабатывания уязвимости:
   - Атакующий узел имеет связь с атакуемым узлом
   - У злоумышленника на атакующем узле есть права root или user
   - На атакуемом узле присутствует хотя бы одна уязвимость

Шаги работы метода моделирования:

1) По данным из файла сетевой топологии выделить набор узлов.
2) По данным из файла связей выделить набор связей всех узлов. Для этого построить матрицу переходов. Матрица переходов M – это квадратная матрица размера NxN, где N – число всех узлов модели, такая что если M[i,j] = 1 - то между узлами i и j есть связь, если M[i,j] = 0 – то связи нет.
3) По данным из файла сетевой топологии и данным из файла с информацией об уязвимостях построить таблицу уязвимостей вида:

   $$V[i] = L,$$

   где L – максимальный уровень прав, который предоставляет уязвимость на узле *i. Если:*
   - L = 4 – как минимум одна уязвимость, которая предоставляет права root
   - L = 3 – как минимум одна уязвимость, которая предоставляет user
   - L = 2 –как минимум одна уязвимость, которая предоставляет dos
   - L = 1 –как минимум одна уязвимость, которая предоставляет other
   - L = 0 – на узле нет уязвимостей
4) Выбрать начальную вершину для моделирования (в программе она должна вводиться с клавиатуры пользователем)
5) Начиная с начального узла провести цикл обхода графа и вычислить трассы атак.
6) Построить граф атаки и пометить на нем трассы атак.
7) Вывести трассы атак в текстовом виде.

Предлагается следующий алгоритм обхода графа (студент может разработать свой алгоритм обхода графа):

1. Добавить начальный узел в список обрабатываемых узлов.
2. Из списка обрабатываемых узлов взять текущий узел.
3. Для текущего узла определить права атакующего.
4. Если права < 3, то данный узел является конечным в трассе атаки, перейти к п.9. Иначе – к п.5.
5. Для текущего узла по матрице переходов найти все связанные вершины.
6. По таблице уязвимостей найти все связанные вершины, на которых есть уязвимости.

7. Добавить все данные вершины в список обрабатываемых узлов.
8. Записать трассу атаки.
9. Удалить текущий узел из списка обрабатываемых узлов.
10. Если в списке обрабатываемых узлов больше нет узлов – к п.11, если есть – к п.2

3. Ход работы

3.1. Порядок выполнения

1) Разработать программу моделирования простейшего графа атаки
2) Подготовить тестовые файлы
3) Провести моделирование
4) Получить результаты
5) Подготовить отчет

3.2. Требования в программе

Исходный код: на языке Python. Для построения графа использовать библиотеку NetworkX [3]. Для работы с матрицами лучше использовать библиотеку numpy.

Для работы программа должна использовать три файла: файл сетевой топологии, файл связей и файл с информацией об уязвимостях

Сетевая топология описывается текстовым файлом в формате:

```
<IP-адрес узла> : <ID-уязвимости>, <ID-уязвимости>, <ID-уязвимости> \n
<IP-адрес узла> : <ID-уязвимости>, <ID-уязвимости>, <ID-уязвимости> \n
```

Например:

192.168.134.1 : V-1, V-2, V-3 192.168.134.2 : V-3, V-4, V-5

Связи описываются текстовым файлом:

```
<IP-адрес маршрутизатора>:
> IP-адрес связанного маршрутизатора
> IP-адрес связанного маршрутизатора
+<IP-адрес узла>
+<IP-адрес узла>
```

здесь:

<IP-адрес маршрутизатора> - внешний IP-адрес маршрутизатора находящегося на границе подсети.

<IP-адрес связанного маршрутизатора> – внешний адрес маршрутизатора с которым связан данный.

<IP-адрес узла> - IP-адрес узла, который находится внутри данной подсети

\+ - узел доступен для связанных маршрутизаторов

\- - узел недоступен для связанных маршрутизаторов

Например:

192.168.1.5 : > 192.168.1.6 > 192.168.1.7

+192.168.134.1 - 192.168.134.2

Информация об уязвимостях представляется текстовым файлом следующего формата:

`<ID-уязвимости>:<Права>`

здесь:

<ID-уязвимости> - уникальный идентификатор уязвимости

<Права> - права злоумышленника при эксплуатации данной уязвимости, права могут быть root/user/dos/other

Программа должна:

1) Запрашивать пути к файлам сетевой топологии, связей и файл с информацией об уязвимостях как аргументы командной строки
2) Строить граф сети с использованием NetworkX
3) Запрашивать начальный узел для атаки в форме IP-адреса (с клавиатуры)
4) Расчитывать какие узлы достижимы из данного начального узла. Расчет производить в соответствии с п. 2.2.
5) На графе сети раскрашивать трассу атаки красным цветом.

3.3. Требования к отчету

Подготовить протокол, который должен включать :

1) Титульный лист.
2) Цель и задачи работы.
3) Файлы сетевой топологии, связей и файл с информацией об уязвимостях.
4) Исходный код программы
5) Скриншоты графов атак

Список использованных источников

[1] Baiardi F., Telmon C., Sgandurra D. Haruspex—Simulation-driven Risk Analysis for Complex Systems .- URL://http://www.isaca.org/JOURNAL/ARCHIVES/2012/VOLUME-3/Pages/Haruspex-Simulation-driven-Risk-Analysis-for-Complex-Systems.aspx

[2] *Lippmann R.P., Ingols K.W., Piwowarski K.* Practical Attack Graph Generation for Network Defense. –http://www.ll.mit.edu/IST/pubs/70.pdf

[3] NetworkX для удобной работы с сетевыми структурами URL:// https://habrahabr.ru/post/125898/

[4] Установка NetworkX URL://https://networkx.github.io/documentation/networkx-1.10/install.html