CMSC626 Fall Semester
Projects 1 and 2

Introduction

In this course, you will undertake two projects that will offer a deep dive into the vulnerability aspects of machine learning malware detectors and their real-world applications. You will also build an encrypted file system with key revocation.

Project 1: Build and break a ML detector.

1. Follow the repo to understand how to build a detector**: https://github.com/uvasrg/EvadeML**
2. Use following link to get dataset: https://contagiodump.blogspot.com/2013/03/16800-clean-and-11960-malicious-fil es.html
    a. Password: The password scheme is infected666 followed by the last character before the zip extension. e.g abc.zip will have the password infected666c.
3. Define 10 features to train the model.
4. Store your detector with the 'pkl' file.
5. Use EvadeML to attack your detector.
6. Summarize your findings and submit a repo.
7. Use a Virtual Machine. This contains malware (DO NOT RUN IT ON THE SERVER!)

Understanding Vulnerabilities in Cybersecurity Detectors:

Objective: The goal of this project is to foster a comprehensive understanding of the susceptibilities inherent in machine learning-based detectors.

Project 1 To Do:

1. Construct your own malware detector, store it into a 'pkl' file, namely 'model.pkl'.

2. Integrate the repository EvadeML(https://github.com/uvasrg/EvadeML) to challenge and attack your detector.

3. Submit a detailed report, alongside your code repository, explain your observations,

and findings.

Duration: You will have 2 weeks to finish this project. This foundational project is designed in a tutorial-like manner to facilitate your grasp of the primary concepts.

Incorporate a detector, inspired by your knowledge from Project 1 to recognize potential threats and breaches.

In a test of resilience, attempt to exploit vulnerabilities in your own detector to understand its strengths and limitations.

Evaluation Criteria: Your final grade will hinge on the intricacy and thoroughness of your design. A rudimentary system that merely logs overt unauthorized attacks will yield a foundational score. You are encouraged to elevate the sophistication of your system by configuring it to detect intricate attacks. The real challenge is to find and exploit the weak points of your advanced detector.

Harness these projects as opportunities to delve into better understand the challenges of cybersecurity and malware detection, all while cultivating hands-on skills that stand crucial in the contemporary digital age.

You will work in groups of five for the final project. Your group is required to turn in the code and a short write-up describing the design and implementation of your project, and to make a short in-class demo presentation about your work. Each group will submit **one** copy of the project report and code. At the end of the semester, we will ask group members to evaluate their team members' performance. We want each group member to have equal contributions to the project. Usually, each member will get the same grade for a project.

Deliverables Project 1(One submission per group. DO NOT submit individually)

1. Form a Group (by the end of 1st week)

We've helped you form a group of five. Look in Blackboard to find your group. If you want to join other groups, you need to make such adjustments before the end of the 1st week of the semester.

2. Working on the project consistently and everyone must make equal contributions.

You will maintain your code in the git repository and have consistent commit history (From all team members). **Failure to do so will result in 0 grade for this project.**

3. Complete the first project (by the end of 2nd week of the class).

You must submit a repo link and report to describe your finding.

Project 2: Requirements for the Encrypted and Distributed File System Project

Tasks:

Develop a robust secure file-sharing system.

Your next requirement is to allow users to store data on untrusted distributed file servers (P2P). At a minimum, your file system should meet the following requirements:

Requirements for the distributed file system

- This is a P2P file system (https://en.wikipedia.org/wiki/Peer-to-peer).
- Users can create, delete, read, write, restore files.
- A client should always see the latest version of a file, or at least that a client should never see an older version of a file after it sees a newer one.
- Users should be able to set permissions on files and directories, which also requires that your file system be able to name users.
- The system should be able to deal with concurrent write and read.
- File names (and directory names) should be treated as confidential. The data stored in each peer node should be encrypted.
- The communication between Peer to Peer should be encrypted.
- Users should not be able to modify files or directories without being detected unless they are authorized to do so.
- A malicious file server should not be able to create or delete files or directories without being detected.
- **Incorporate a log into your file system to track each operation on your server. Using your knowledge from the first project, define features that will be used to detect attacks to your system.** Generate logs with unauthorized access/modifications and logs with authorized access/modifications. Build your detector.
- Define the vulnerability of your detector. Use your knowledge of the first

project to attack your detector.

When complete, these projects will result in a functional file system implementation that meets the above requirements. You can implement your prototype in any language you want, such as Python or C++ or Go. You can decide how the file system client and server should be run. One reasonable design would be to have the file system client provide a minimal shell environment that allows users to perform the operations described in the above requirements.

Due Dates and turn ins for Project 2

1. Complete the design doc of the second project (before the end of the 1st month).

Update your design doc into your repo. You can update after your submission.

2. Code (2 weeks before the final)

   a. You will maintain your code in the git repository and submit your git repository via blackboard.
   b. Benchmark your result with 100K requests: random read, random write, random read and write. Clearly define your evaluation criteria. You should at least measure the runtime.

3. Write-up wiki (2 weeks before the final)

Write a document describing your project in detail (benchmark should be put into the experiment part), and turn it in along with your project's code by the final deadline.

Below is a link to some good write ups from past years
(2012): http://css.csail.mit.edu/6.858/2012/projects.html
and 2013:
http://css.csail.mit.edu/6.858/2013/projects.html) to get a sense of what this writeup should look like.

4. Presentation (the last week (tentative))

Prepare a 5 mins in-class presentation of your project.

- 2 slides about your design and
- 2 slides about your benchmark.

The presentation will be held online. 5 mins for Q&A.

Grade rubric:

1. 20% first project + 80% second project.
2. For the second project:
    a. 50% distributed file system (requirements 1-9) + 15% (built detector) + 35% (attack your detector)