Week 12: Lecture A
Adversarial Machine Learning

Tuesday, November 14, 2023
Announcements

- **Project 3** grades are now available on Canvas

- **Statistics:**
  - Average score: **90%**

- **Fantastic job!**
Announcements

- **Project 3** grades are now available on **Canvas**

- Think we made an error? Request a regrade!
  - Valid regrade requests:
    - You have verified your solution is correct (i.e., we made an error in grading)

---

**Project 3 Regrade Requests** (see Piazza pinned link):
Submit by **11:59 PM** on **Monday 11/20** via **Google Form**
■ **Project 4: NetSec** released
  - **Deadline:** Thursday, December 7th by 11:59PM

---

### Project 4: Network Security

**Deadline: Thursday, December 7 by 11:59PM.**

Before you start, review the course syllabus for the Lateness, Collaboration, and Ethical Use policies.

You may optionally work alone, or in teams of at most two and submit one project per team. If you have difficulties forming a team, post on Piazza's Search for Teammates forum. Note that the final exam will cover project material, so you and your partner should collaborate on each part.

The code and other answers your group submits must be entirely your own work, and you are bound by the University's Student Code. You may consult with other students about the conceptualization of the project and the meaning of the questions, but you may not look at any part of someone else's solution or collaborate with anyone outside your group. You may consult published references, provided that you appropriately cite them (e.g., in your code comments). **Don't risk your grade and degree by cheating!**

Complete your work in the **CS 4440 VM** — we will use this same environment for grading. You may not use any external dependencies. Use only default Python 3 libraries and/or modules we provide you.
<table>
<thead>
<tr>
<th>Task</th>
<th>Progress</th>
</tr>
</thead>
<tbody>
<tr>
<td>Working on Part 1.0: Password Cracking</td>
<td>0%</td>
</tr>
<tr>
<td>Working on Part 1.1: Port Scanning</td>
<td>0%</td>
</tr>
<tr>
<td>Working on Part 1.2: Anomalous Activity</td>
<td>0%</td>
</tr>
<tr>
<td>Finished Part 1, working on Part 2!</td>
<td>0%</td>
</tr>
<tr>
<td>None of the above</td>
<td>0%</td>
</tr>
</tbody>
</table>
Participation

- **Preliminary Participation** score on Canvas
  - Designed to help you gauge where you’re at
  - Not your final participation score!

- Cliff notes:
  - Everyone receives full Piazza participation
  - Remaining 50% of score based on lectures

- Total of **22 lectures** (11 weeks * 2 lectures)
  - Participation capped at 70% (~15 / 22) of lectures
Participation

- **Preliminary Participation** score on Canvas
  - Designed to help you gauge where you’re at
  - Not your final participation score!

- Cliff notes:
  - Everyone receives full Piazza participation
  - Remaining 50% of score based on lectures

- Total of **22 lectures** (11 weeks * 2 lectures)
  - Participation capped at 70% (~15 / 22) of lectures

**Score lower than expected?**

Check that you have a Poll Everywhere account

(or just insert your UID in your account name)

**Haven’t been attending?**

There’s still time to get the points!
See Discord for meeting info!

www.utahsec.com
Questions?
Last time on CS 4440...

Security in Practice: Tor—The Onion Router
Anonymity Primitive: Onion Routing

- Each message is ???
Anonymity Primitive: Onion Routing

- Each message is **repeatedly encrypted**
  - **Analogy:** multiple layers of an onion

- Sent through **multiple network nodes**
  - These nodes are called **onion routers**
  - Each node removes an encryption layer to uncover the message **routing instructions**
  - Process repeats when sent to next router

- **Anonymity:** prevents ???

Anonymity Primitive: Onion Routing

- Each message is **repeatedly encrypted**
  - **Analogy**: multiple layers of an onion

- Sent through **multiple network nodes**
  - These nodes are called **onion routers**
  - Each node removes an encryption layer to uncover the message **routing instructions**
  - Process repeats when sent to next router

- **Anonymity**: prevents any intermediary nodes from knowing message **origin**, **destination**, and **contents**
Onion Routing Visualized

Sending data to a website

Client → Entry → Middle → Exit → Website

Receiving data from a website

Client ← Entry ← Middle ← Exit ← Website
Tor: The Onion Router

- **Tor**: a distributed overlay network
  - Anonymizes TCP-based applications
    - Secure shell
    - Web browsing
    - Instant messaging
**Tor: The Onion Router**

- **Tor**: a distributed overlay network
  - Anonymizes TCP-based applications
    - Secure shell
    - Web browsing
    - Instant messaging

- Clients choose ???
**Tor: The Onion Router**

- **Tor**: a distributed overlay network
  - Anonymizes TCP-based applications
    - Secure shell
    - Web browsing
    - Instant messaging

- Clients choose the **circuit paths**
  - Messages unwrapped at each onion router using a symmetric key

- Onion routers only know ???
**Tor: The Onion Router**

- **Tor**: a distributed overlay network
  - Anonymizes TCP-based applications
    - Secure shell
    - Web browsing
    - Instant messaging

- Clients choose the **circuit paths**
  - Messages unwrapped at each onion router using a symmetric key

- Onion routers only know their **successor** or **predecessor** nodes
  - They don’t know of any other nodes
How Tor Works

Tor Client

Entry guard

Encrypted by Tor
Not encrypted by Tor

Tor Network

Middle relay

Exit relay

Destination
Possible attacks against Tor?
Attacking Tor

- Possible attacks against Tor?
  - **Leak DNS requests** when they aren’t transmitted via Tor
  - Perform **volume/timing analysis** to characterize behavior
  - **Add malicious nodes** to intercept unencrypted exit traffic
Attacking Tor

- **Possible attacks against Tor?**
  - **Leak DNS requests** when they aren’t transmitted via Tor
    - Defense: ???
  - Perform **volume/timing analysis** to characterize behavior
    - Defense: ???
  - **Add malicious nodes** to intercept unencrypted exit traffic
    - Defense: ???
Possible attacks against Tor?

**Leak DNS requests** when they aren’t transmitted via Tor
- **Defense:** enforce all DNS requests through Tor encryption

Perform **volume/timing analysis** to characterize behavior
- **Defense:** inject noisy data to throw off analysis heuristics

**Add malicious nodes** to intercept unencrypted exit traffic
- **Defense:** never use unencrypted protocols—use HTTPS
Who uses Tor?

- ???
Who uses Tor?

- **Normal People**
  - Privacy-conscious folks

- **Intelligence Agencies**
  - Secret agents in the field

- **Law Enforcement**
  - Online “undercover” operations

- **Journalists and Bloggers**
  - Citizen journalists inspiring social change

- **Activists and Whistleblowers**
  - Raising their voice and avoiding persecution

- **White-hat and Black-hat Hackers**
  - And everyone in between!
Who uses Tor?

The anonymous Internet

Daily Tor users per 100,000 Internet users

- > 200
- 100 - 200
- 50 - 100
- 25 - 50
- 10 - 25
- 5 - 10
- < 5
- no information

Average number of Tor users per day calculated between August 2012 and July 2013

data sources:
- Tor Metrics Portal
- metrics.torproject.org
- World Bank
- data.worldbank.org

by Mark Graham (@geoloci) and Stefano De Sabbata (@map4tough)

Internet Geographies at the Oxford Internet Institute
2014 • geography.ox.ac.uk

Oxford Internet Institute
University of Oxford
What services get hidden?

THIS HIDDEN SITE HAS BEEN SEIZED
by the Federal Bureau of Investigation,
in conjunction with the IRS Criminal Investigation Division,
ICE Homeland Security Investigations, and the Drug Enforcement Administration,
in accordance with a seizure warrant obtained by the
United States Attorney’s Office for the Southern District of New York
and issued pursuant to 18 U.S.C. § 983(j) by the
United States District Court for the Southern District of New York
Positive Tor Use Cases

Privacy is a human right

HANDS OFF MY DATA
Questions?
Recap: Project 4 Overview

- Focuses on **network packet analysis**
  - Leveraging data contained within packets to achieve network defenses and attacks

- **Scenario:** helping a fictional university secure its enterprise campus network
  - Detect and characterizing likely attacks
  - Demonstrate how info can be intercepted
We provide a series of network packet traces (pcaps)

- **Your job:** write scripts to analyze them!

**Part 1:** detecting **network attacks**
- Password cracking, port scanning, SYN floods

**Part 2:** stealing **sensitive information**
- Unencrypted credentials, browsing history
- **Extra credit:** stealing transferred files

Recap: Project 4 Overview
We provide a series of network packet traces (pcaps)
  - Your job: write scripts to analyze them!

**Part 1: detecting network attacks**
- Password cracking, port scanning, SYN floods

**Part 2: stealing sensitive information**
- Unencrypted credentials, browsing history
- Extra credit: stealing transferred files

You will use Python 3’s Scapy library
- A huge and powerful packet analysis API...
- But we’ll really only use a few parts of it
Recap: Scapy Fundamentals

- Python API for programmatic packet capture and analysis
  - Think of it as "Wireshark in API form"

- We provide **skeleton code** template
  - Sets-up the packet parsing workflow

```python
#!/usr/bin/python3
import logging
logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
from scapy.all import *
import re

def parsePacket(packet):
    if not packet.haslayer("TCP"):
        return
    # ------------------------------
    # TODO: finish implementing parsePacket()!
    # ------------------------------
    return

if __name__ == "__main__":
    for packet in rdpcap(sys.argv[1]):
        parsePacket(packet)
```
Recap: Scapy Fundamentals

- Python API for programmatic packet capture and analysis
  - Think of it as "Wireshark in API form"

- We provide **skeleton code** template
  - Sets-up the packet parsing workflow
  - **Your job:** finish implementing the function `parsePacket()`

```python
#!/usr/bin/python3
import logging
logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
from scapy.all import *
import re

def parsePacket(packet):
    if not packet.haslayer("TCP"):
        return
    # -----------------------------------------------
    # TODO: finish implementing parsePacket()!
    # -----------------------------------------------
    return

if __name__ == "__main__":
    for packet in rdpcap(sys.argv[1]):
        parsePacket(packet)
```
Recap: Scapy Fundamentals

- Python API for programmatic packet capture and analysis
  - Think of it as "Wireshark in API form"

- We provide **skeleton code** template
  - Sets-up the packet parsing workflow
  - **Your job:** finish implementing the function `parsePacket()`

- You may also add **additional code**
  - E.g., global variables or data structures
  - E.g., printing functionality in `main()`

```python
#!/usr/bin/python3
import logging
logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
from scapy.all import *

import re

def parsePacket(packet):
    if not packet.haslayer("TCP"): return
    # ____________________________
    # TODO: finish implementing parsePacket()
    # ____________________________
    return

if __name__ == "__main__":
    for packet in rdpcap(sys.argv[1]):
        parsePacket(packet)
```
Recap: Scapy Fundamentals

- Only a few things you’ll need...
  - Get a packet’s **TCP flags**:
    ```python
    packet[“TCP”].flags
    ```
  - Get a packet’s **destination port**
    ```python
    packet[“TCP”].dport
    ```
  - Get a packet’s **source IP address**
    ```python
    packet[“IP”].src
    ```
  - Get a packet’s **TCP payload**:
    ```python
    bytes(packet["TCP"].payload).decode(‘utf-8’, ‘replace’)```
Recap: Suggested Workflow

- Before you start writing a **Scapy** script, inspect the trace *manually* via **Wireshark**
  - Super helpful for viewing a packet’s contents
  - Use this to bootstrap your script’s approach!
Recap: Suggested Workflow

- Before you start writing a **Scapy** script, inspect the trace *manually* via **Wireshark**
  - Super helpful for viewing a packet’s contents
  - Use this to bootstrap your script’s approach!

- For each target, answer the following:
  - What **packet fields** matter?
  - How to **extract** relevant data?
  - How to **store and process** this data?
Recap: Suggested Workflow

- Before you start writing a Scapy script, inspect the trace manually via Wireshark
  - Super helpful for viewing a packet’s contents
  - Use this to bootstrap your script’s approach!

- For each target, answer the following:
  - What packet fields matter?
  - How to extract relevant data?
  - How to store and process this data?

- Finalize your high-level game plan first!
  - Then start developing your solution scripts!
Questions?
This time on CS 4440...

Adversarial Machine Learning
Data Poisoning Attacks
Evasion Attacks
ML Ethics
What is ML?

- What is it?
What is ML?

- **What is it?**
  - Cars that no longer require *people* to cause accidents
  - The algorithm that feeds you videos of such accidents
  - ChatGPT and our other forthcoming AI overlords
What really is ML?

“Machine Learning (ML) is a field of inquiry devoted to understanding and building methods that ‘learn’ – that is, methods that leverage data to improve performance on some set of tasks.”
Uses of ML

- **ML is all around us!**
  - By now, we interact with ML-based systems daily
  - Expanding every year

- **Real-world examples:**
  - ???
Uses of ML

- **ML is all around us!**
  - By now, we interact with ML-based systems daily
  - Expanding every year

- **Real-world examples:**
  - Marketing optimization
  - Fraud detection
  - Stock trading
  - Self-guided cars
  - Cancer screening
  - Others?
Do you trust ML?
Do you personally trust ML / AI?

Yes! 0%

"It's complicated..." 0%

No way! 0%
Do you trust ML?

- To separate apples and oranges?
- To send you interesting content?
- To drive your car while you nap?
- To determine your creditor risk?
- To recognize possible criminals?
Do you trust ML?

- To separate apples and oranges?
- To send you interesting content?
- To drive your car?
- To determine your creditor risk?
- To recognize possible criminals?

How does ML even work?
Brief Intro to ML
ML in a Nutshell

- ML aims to ???
ML aims to produce a model that correctly makes inferences about inputs:

- E.g., category of a given input fruit (apple or orange)
- E.g., likelihood of a person being a reliable creditor
Most ML is rooted in three statistical principles
- Regression, classification, and clustering

Regression:
- ???
Most ML is rooted in three statistical principles
- Regression, classification, and clustering

**Regression:**
- Model that predicts a **continuous quantity**
- E.g., tomorrow’s temperature
- E.g., happiness versus income
- E.g., age, monthly spending, cost
- **Examples:**
  - Linear regression
  - Logistic regression
Most ML is rooted in three statistical principles
- Regression, classification, and clustering

Classification:
- ???
Most ML is rooted in three statistical principles
- Regression, classification, and clustering

Classification:
- Model that predicts an input’s category
  - E.g., pass/fail final exam performance
  - E.g., positive/negative teaching evals
  - E.g., whether a fruit is an apple/orange
- Examples:
  - K-nearest neighbors
  - Decision trees
Most ML is rooted in three statistical principles
- Regression, classification, and clustering

Clustering:
- ???

Diagram:
- Weight axis
- Height axis
- Cluster 1: w/ Child Characteristics
- Cluster 2: w/ Adult Characteristics
Most ML is rooted in three statistical principles
- Regression, classification, and clustering

Clustering:
- Model that **groups inputs** by category
  - E.g., group dogs by breed phenotypes
  - E.g., person credit w.r.t. spending history
  - E.g., different types of apples and oranges
- **Examples:**
  - K-means clustering
  - Centroid-based clustering
Feature Dimensions

- Features:
  - ???
Feature Dimensions

- **Features:**
  - The *data representation* we care about
  - Can be affixed in advance
  - Can be inferred and/or augmented
Feature Dimensions

- **Features:**
  - The *data representation* we care about
  - Can be affixed in advance
  - Can be inferred and/or augmented

- **Dimensions:**
  - ???

<table>
<thead>
<tr>
<th>Make</th>
<th>Honda</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Accord</td>
</tr>
<tr>
<td>Shape</td>
<td>Sedan</td>
</tr>
<tr>
<td>Color</td>
<td>Red</td>
</tr>
<tr>
<td>Trim</td>
<td>Black</td>
</tr>
</tbody>
</table>
Feature Dimensions

- **Features:**
  - The *data representation* we care about
  - Can be affixed in advance
  - Can be inferred and/or augmented

- **Dimensions:**
  - *How many* features considered
  - Provided in advance or learned

- **Goal:** map *higher* dimension to *lower*
  - **Example:** map 3-D image to a 2-D vector

Make : Honda  
Model : Accord  
Shape : Sedan  
Color : Red  
Trim : Black
**Dimensionality Reduction**

- **ML algorithms perform better when dimensionality is reduced**
  - More features = harder to model (higher resource overhead, mathematical complexity)
Training vs. Testing

- **Training Data:**
  - ???
Training vs. Testing

- **Training Data:**
  - The data used to **build** the model
  - May be labeled or unlabeled
  - **Example:** pictures of all orange types

- **Testing Data:**
  - ???
Training vs. Testing

- **Training Data:**
  - The data used to **build** the model
  - May be labeled or unlabeled
  - **Example:** pictures of all orange types

- **Testing Data:**
  - Data used to test model’s **correctness**
  - Should be **separate** from training data!
  - **Example:** tangerines, clementines, kiwis
Machine Learning Types

- Supervised
  - ???
Machine Learning Types

- **Supervised**
  - Training data *includes desired labels*
  - **Goal:** learn general input-to-output *mapping*

- **Unsupervised**
  - ???

Apples vs. oranges
Machine Learning Types

- **Supervised**
  - Training data includes desired labels
  - **Goal:** learn general input-to-output mapping

- **Unsupervised**
  - Training data comes without labels
  - **Goal:** deduce input patterns and/or features

- **Reinforcement**
  - ???

Diagram:

- Supervised: Task Driven (Predict next value)
- Unsupervised: Data Driven (Identify clusters)
- Apples vs. oranges
- Group risky creditors
Machine Learning Types

- **Supervised**
  - Training data includes desired labels
  - **Goal:** learn general input-to-output mapping

- **Unsupervised**
  - Training data comes without labels
  - **Goal:** deduce input patterns and/or features

- **Reinforcement**
  - Model interacts with rewarding environment
  - **Goal:** produce outputs that maximize reward
When to use ML

What do you want the machine learning system to do?

I want to see if there are natural clusters or dimensions in the data I have about different situations.

I want to learn what actions to take in different situations.

Do you want the ML system to be active or passive?

ACTIVE
The system’s own actions will affect the situations it sees in the future.

PASSIVE
The system will learn from data I give it.

Do you have access to data that describes a lot of examples of situations and appropriate actions for each situation?

Yes

Do a knowledgeable human decide what actions to take based on the data you have about the situation?

Yes

Could there be patterns in these situations that humans haven’t recognized before?

Yes

Could there be patterns in these situations that humans haven’t recognized before?

No

MACHINE LEARNING IS NOT USEFUL

REINFORCEMENT LEARNING MAY BE APPROPRIATE

UNSUPERVISED LEARNING MAY BE APPROPRIATE
clustering
anomaly detection

SUPERVISED LEARNING MAY BE APPROPRIATE
neural nets
support vector machines
regression machines
recommender systems

Will the system be able to gather a lot of data by trying sequences of actions in many different situations and seeing the results?

Yes

No

Credit: Thomas Malone, MIT Sloan | Design: Laura Wentzel
Questions?
Attacks on ML
Threat Model

- **Data poisoning** (causative attack):
  - ???
Threat Model

- **Data poisoning** (causative attack):
  - Attackers **perturb training set** to fool the model (insert/modify data, alter data labels)
  - Attackers attempt to **influence or corrupt** the ML model or the ML algorithm itself
Threat Model

- **Evasion attack** (exploratory attack):
  - ???
**Threat Model**

- **Evasion attack** (exploratory attack):
  - Attackers don’t tamper with ML model, but instead cause it to produce **adversarial outputs**
  - Attack on the model’s **testing phase**—by far the most common attack type seen today
Questions?
Evasion Attacks
Simple Perturbations

- **Goal:** break developer’s assumptions about *friendliness* of input data
  - Simple transformations such as image resizing, flipping, etc.

Model: *gorilla*  
Model: *not gorilla*
Model: *not gorilla*
Noise Injection

- **Goal:** inject *random noise* to fool model into an incorrect classification
  - Taking negative pixel values, injecting random noise, etc.

Model: gorilla

Model: weimaraner

Model: weimaraner
**Goal:** inject *random noise* to fool model into an incorrect classification

- Taking negative pixel values, injecting random noise, etc.

Model: *gorilla*  
Model: *???*  
Model: *fountain*
Targeted Perturbation

- **Goal:** influence model into *incorrectly classifying* image as something else
  - Easier for white-box attack scenarios; to human eye looks *indiscernible*

Model: hippo

Model: maraca

Model: maraca
Adversarial Examples

- Possible attacker use cases?
Adversarial Examples

Synthesizing Robust Adversarial Examples: Adversarial Turtle
Adversarial Examples

<table>
<thead>
<tr>
<th>Distance/Angle</th>
<th>Subtle Poster</th>
<th>Subtle Poster Right Turn</th>
<th>Camouflage Graffiti</th>
<th>Camouflage Art (LISA-CNN)</th>
<th>Camouflage Art (GTSRB-CNN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5' 0°</td>
<td>STOP</td>
<td></td>
<td>STOP</td>
<td>STOP</td>
<td>STOP</td>
</tr>
<tr>
<td>5' 15°</td>
<td>STOP</td>
<td></td>
<td>STOP</td>
<td>STOP</td>
<td>STOP</td>
</tr>
<tr>
<td>10' 0°</td>
<td>STOP</td>
<td></td>
<td>STOP</td>
<td>STOP</td>
<td>STOP</td>
</tr>
<tr>
<td>10' 30°</td>
<td>STOP</td>
<td></td>
<td>STOP</td>
<td>STOP</td>
<td>STOP</td>
</tr>
<tr>
<td>40' 0°</td>
<td>STOP</td>
<td></td>
<td>STOP</td>
<td>STOP</td>
<td>STOP</td>
</tr>
</tbody>
</table>

Targeted-Attack Success: 100% 73.33% 66.67% 100% 80%
Adversarial Examples
Adversarial Examples
Figure 1: We create an adversarial patch that is successfully able to hide persons from a person detector. Left: The person without a patch is successfully detected. Right: The person holding the patch is ignored.
Adversarial Examples

Deep Fake via **Generative Adversarial Network (GAN)**
ML Ethics
Who is responsible for ethical ML?

- This (and likely the next) century’s **existential question**...

- The government?
- The user?
- The developer?
Who is responsible for ethical ML/AI?

- The government: 0%
- The user: 0%
- The developer: 0%
- All of the above: 0%
- None of the above: 0%
Unethical Uses

One Month, 500,000 Face Scans: How China Is Using A.I. to Profile a Minority

In a major ethical leap for the tech world, Chinese start-ups have built algorithms that the government uses to track members of a largely Muslim minority group.
New AI can guess whether you're gay or straight from a photograph

An algorithm deduced the sexuality of people on a dating site with up to 91% accuracy, raising tricky ethical questions

Machine learning-driven credit risk: a systemic review

SI Shi, Rita Tse, Wuman Luo, Stefano D'Addona & Giovanni Pau

Neural Computing and Applications 34, 14327–14339 (2022) | Cite this article

Abstract

Credit risk assessment is at the core of modern economies. Traditionally, it is measured by statistical methods and manual auditing. Recent advances in financial artificial intelligence stemmed from a new wave of machine learning (ML)-driven credit risk models that gained tremendous attention from both industry and academia. In this paper, we systematically review a series of major research contributions (76 papers) over the past eight years using statistical, machine learning and deep learning techniques to address the problems of credit risk. Specifically, we propose a novel classification methodology for ML-driven credit risk algorithms and their performance ranking using public datasets. We further discuss the challenges including data imbalance, dataset inconsistency, model transparency, and inadequate utilization of deep learning models. The results of our review show that: 1) most deep learning models outperform classic machine learning and statistical algorithms in credit risk estimation, and 2) ensemble methods provide higher accuracy compared with single models. Finally, we present summary tables in terms of datasets and proposed models.
Training Data Leakage
Training Data Leakage

henearkr on July 5, 2021 | parent | context | favorite | on: GitHub Copilot regurgitates valid secrets
Has Copilot been trained on private repos as well?
If so, it means that you wouldn't find them by a search, but they would still be revealed by the A.I.

intricatedetail on July 5, 2021 | parent | prev | next [-]
Who can you verify it is true?

mrfusion on July 5, 2021 | root | parent | next [-]
We trust corporations to tell us the truth every day. What do you think the news is.
Will Automation Reduce Trucking Jobs?
August 10, 2022
By Jack Dunn

The automation of jobs has become an increasingly prominent development across the entire U.S. labor force. However, the trucking industry may be affected unlike others.

What Is Labor Automation?

Labor automation is a practice that dates back centuries in our country. It can best be described as substituting technology for human labor when performing jobs or tasks. Eli Whitney’s cotton gin was able to remove the seeds from 50 pounds of cotton in a single day. Doing the job by hand only yielded about one pound of cotton per day.

Remember switchboard operators? Early telephone infrastructure used humans to link calls. Operators would manually insert phone jacks into the appropriate spot to connect calls. Those jobs are now few and far between as automatic switching has taken over.
Economic Displacement
Questions?
Next time on CS 4440...

Binary Reverse Engineering