Lecture 25
November 25, 2024

Persistent vs. Volatile Data

- Persistent data: remains when system or data storage is powered off
  - Data on hard drive or secondary storage
- Volatile data: transient, disappearing at some point in time (like when system is powered off)
  - Data in memory
  - More difficult to capture than persistent data

Capturing Volatile Data

- Problem: using software to capture memory contents alters memory
- One approach: use specialized hardware
  - Carrier and Grand built custom PCI card; attached to bus
    - When computer boots, card configures itself, disables its controller so it is invisible to programs scanning PCI bus
    - Throw switch, card re-enables controller, suspends CPU, dumps memory to a non-volatile storage medium
    - When done, disables its controller and restart CPU

Capturing Volatile Data

- Second approach: store memory-reading software in trusted location
  - Attacker cannot alter it
  - Software freezes operating system and all associated processes, captures and dumps memory contents, unfreezes operating system and all associated processes
  - Intel IA-32 platforms have System Management Mode to provide such an area
    - SMM has software drivers for standard network PCI card
    - SMM grabs contents of CPU registers, and PCI grabs contents of memory; these transmitted to waiting server
    - Using SMM suspends operating system so memory contents in consistent state

Capturing Volatile Data

- Third approach: put acquisition software between operating system, hardware
  - Virtual machine introspection does this; to capture memory contents, virtual machine monitor stops VM, copies contents of memory
- Fourth approach: remanence effect
  - Memory retains contents for very short time after power lost
  - Cooling memory increases this time significantly
  - This used for forensics on Android phones

Extracting Information

- Analyze to produce a timeline
- Example for the disk mentioned earlier; work done from disk image
  1. Analysts obtain list of files on disk
  2. They check for deleted files; find several corresponding to undeleted files
  3. They examine free space; find large number of files there

Analyze the Data

- Goal is to answer specific questions that depend on nature of attack, resources involved, and the data
- Example for the disk mentioned earlier; information gathered from disk image
- Analysts examine files stored in free space as they are hidden; turn out to be copies of recently released movies
- Key question: how did they get there?
- Analysts extract log files of network server, user actions; find a login name with control characters in it, and no corresponding logout; possible buffer overflow
  - Validation: run login program, give it user name of 1000 characters; it crashes

Analyze the Data

How did attackers gain access to system (to run login program)?

- Analysts examine server logs, server configuration files; nothing suspicious
- Analysts look through other network log files, find an entry made by a program starting the *telnet* service
  - This is a remote terminal interface and should never run
  - Find the program in a sysadmin's directory
- Analysts look at network logs
  - IDS captures packets, stored for 30 days
  - After that, deletes packet bodies and saves headers for 5 months

Analyze the Data

- Analysts look for *telnet* packets; find several, including one containing the user name matching the one with control characters
- Analysts copy these packets to separate file, create a textual representation in another file
  - And these are checksummed and saved on read-only media
- How did movies get put into free space?
  - Obvious answer: attackers simply deleted them or wrote them directly to free space
    - But then disk would not have been full as deleted blocks would simply be overwritten
  - More probable answer: attacker created file, opened it, deleted file from file system
    - Program checking disk space by traversing file hierarchy will miss it; looking at disk map won't; this also explains discrepancy

Report the Findings

Must take into account the audience (principle of presenting information in an understandable way)

- If non-technical audience, report should say movie files stored in unused disk space, and give data on number of movies found, titles, and so forth
- If technical audience, also describe how movies stored, how they were found

This suggests preparing a detailed technical report for reference, then use that as basis for writing other reports as needed

Anti-Forensics

- *Anti-forensics*: the attempt to compromise the availability or usefulness of evidence to forensics process
- Goals:
  - Interfere with forensic analysis tools gathering information, by hiding data or obscuring type, sequence of evidence
  - Hinder the validation of authenticity of digital image
  - Exploit weaknesses in forensic analysis tools
  - Attacking users of forensic analysis tools, for example by crashing analyst's system or increasing time needed to analyze data
  - Cast doubt on results of forensic analysis; will diminish its credibility in court, for example

Examples

- *timestomp*: enables user to change file access times
- *event_manager*: enables user to delete entries from log files
- JPEG image data compresses digital representation of image into multiple bands of transform coefficients, which generally follow a smooth distribution; altering image perturbs coefficients, so distribution different; anti-forensic tools add dithering to change coefficients back to approximate original one
- Forensic tool determines if Windows files are executable by looking at file extension (“.exe”) and first 2 bytes of file (“MZ”), so anti-forensics tools can just change the extension

Intrusion Detection

- Detect wide variety of intrusions
  - Previously known and unknown attacks
  - Suggests need to learn/adapt to new attacks or changes in behavior
- Detect intrusions in timely fashion
  - May need to be be real-time, especially when system responds to intrusion
    - Problem: analyzing commands may impact response time of system
  - May suffice to report intrusion occurred a few minutes or hours ago

Intrusion Detection Systems

- Present analysis in simple, easy-to-understand format
  - Ideally a binary indicator
  - Usually more complex, allowing analyst to examine suspected attack
  - User interface critical, especially when monitoring many systems
- Be accurate
  - Minimize false positives, false negatives
  - Minimize time spent verifying attacks, looking for them

Principles of Intrusion Detection

- Characteristics of systems not under attack
  - User, process actions conform to statistically predictable pattern
  - User, process actions do not include sequences of actions that subvert the security policy
  - Process actions correspond to a set of specifications describing what the processes are allowed to do
- Systems under attack do not meet at least one of these

Example

- Goal: insert a back door into a system
  - Intruder will modify system configuration file or program
  - Requires privilege; attacker enters system as an unprivileged user and must acquire privilege
    - Nonprivileged user may not normally acquire privilege (violates #1)
    - Attacker may break in using sequence of commands that violate security policy (violates #2)
    - Attacker may cause program to act in ways that violate program's specification

Basic Intrusion Detection

- *Attack tool* is automated script designed to violate a security policy
- Example: *rootkit*
  - Includes password sniffer
  - Designed to hide itself using Trojaned versions of various programs (*ps*, *ls*, *find*, *netstat*, etc.)
  - Adds back doors (*login*, *telnetd*, etc.)
  - Has tools to clean up log entries (*zapper, etc.)*

Detection

- *Rootkit* configuration files cause *ls*, *du*, etc. to hide information
  - *ls* lists all files in a directory
    - Except those hidden by configuration file
  - *dirdump* (local program to list directory entries) lists them too
    - Run both and compare counts
    - If they differ, *ls* is doctored
- Other approaches possible

Key Point

- *Rootkit* does *not* alter kernel or file structures to conceal files, processes, and network connections
  - It alters the programs or system calls that *interpret* those structures
  - Find some entry point for interpretation that *rootkit* did not alter
  - The inconsistency is an anomaly (violates #1)

Denning's Model

- Hypothesis: exploiting vulnerabilities requires abnormal use of normal commands or instructions
  - Includes deviation from usual actions
  - Includes execution of actions leading to break-ins
  - Includes actions inconsistent with specifications of privileged programs

Models of Intrusion Detection

- Anomaly detection
  - What is usual, is known
  - What is unusual, is bad
- Misuse detection
  - What is bad, is known
  - What is not bad, is good
- Specification-based detection
  - What is good, is known
  - What is not good, is bad

Anomaly Detection

- Analyzes a set of characteristics of system, and compares their values with expected values; report when computed statistics do not match expected statistics
  - Threshold metrics
  - Statistical moments
  - Markov model

Misuse Detection

- Determines whether a sequence of instructions being executed is known to violate the site security policy
  - Descriptions of known or potential exploits grouped into *rule sets*
  - IDS matches data against rule sets; on success, potential attack found
- Cannot detect attacks unknown to developers of rule sets
  - No rules to cover them

Types of Learning

- *Supervised learning methods*: begin with data that has already been classified, split it into “training data”, “test data”; use first to train classifier, second to see how good the classifier is
- *Unsupervised learning methods*: no pre-classified data, so learn by working on real data; implicit assumption that anomalous data is small part of data
- Measures used to evaluate methods based on:
  - TP: true positives (correctly identify anomalous data)
  - TN: true negatives (correctly identify non-anomalous data)
  - FP: false positives (identify non-anomalous data as anomalous)
  - FN: false negatives (identify anomalous data as non-anomalous)

Specification Modeling

- Determines whether execution of sequence of instructions violates specification
- Only need to check programs that alter protection state of system
- System traces, or sequences of events t_1, … t_i, t_{i+1}, …, are basis of this
  - Event t_i occurs at time $C(t_i)$
  - Events in a system trace are totally ordered

Comparison and Contrast

- Misuse detection: if all policy rules known, easy to construct rulesets to detect violations
  - Usual case is that much of policy is unspecified, so rulesets describe attacks, and are not complete
- Anomaly detection: detects unusual events, but these are not necessarily security problems
- Specification-based vs. misuse: spec assumes if specifications followed, policy not violated; misuse assumes if policy as embodied in rulesets followed, policy not violated

Measuring Effectiveness

- *Accuracy*: percentage (or fraction) of events classified correctly
  - ((*TP* + *TN*) / (*TP* + *TN* + *FP* + *FN*)) × 100%
- *Detection rate*: percentage (or fraction) of reported attack events that are real attack events
  - (*TP* / (*TP* + *FN*)) × 100%
  - Also called the *true positive rate*
- *False alarm rate*: percentage (or fraction) of non-attack events reported as attack events
  - (*FP* / (*FP* + *TN*)) × 100%
  - Also called the *false positive rate*

Usefulness of Measurement

- Data at installation should be similar to that used to measure effectiveness
- Example: military, academic network traffic different
  - KDD-CUP-99 dataset derived from unclassified and classified network traffic on an Air Force Base
  - Network data captured at Florida Institute of Technology
- FIT data showed anomalies not in KDD-CUP-99
  - FIT data: TCP ACK field nonzero when ACK flag not set
  - KDD-CUP-99 data: HTTP requests all regular, all used GET, version 1.0; in FIT data, HTTP requests showed inconsistencies, some commands not GET, versions 1.0, 1.1
- Conclusion: using KDD-CUP-99 data would show some techniques performing better than they would on the FIT data

Clustering

- Clustering
  - Does not assume *a priori* distribution of data
  - Obtain data, group into subsets (*clusters*) based on some property (*feature*)
  - Analyze the clusters, not individual data points

Example: Clustering

proc	user	value	percent	clus#1	clus#2
p_1	matt	359	100%	4	2
p_2	holly	10	3%	1	1
p_3	heidi	263	73%	3	2
p_4	steven	68	19%	1	1
p_5	david	133	37%	2	1
p_6	mike	195	54%	3	2

- Cluster 1: break into 4 groups (25% each); 2, 4 may be anomalous (1 entry each)
- Cluster 2: break into 2 groups (50% each)

Finding Features

- Which features best show anomalies?
  - CPU use may not, but I/O use may
- Use training data
  - Anomalous data marked
  - Feature selection program picks features, clusters that best reflects anomalous data

Example

- Analysis of network traffic for features enabling classification as anomalous
- 7 features
  - Index number
  - Length of time of connection
  - Packet count from source to destination
  - Packet count from destination to source
  - Number of data bytes from source to destination
  - Number of data bytes from destination to source
  - Expert system warning of how likely an attack

Feature Selection

- 3 types of algorithms used to select best feature set
  - Backwards sequential search: assume full set, delete features until error rate minimized
    - Best: all features except index (error rate 0.011%)
  - Beam search: order possible clusters from best to worst, then search from best
  - Random sequential search: begin with random feature set, add and delete features
    - Slowest
    - Produced same results as other two

Results

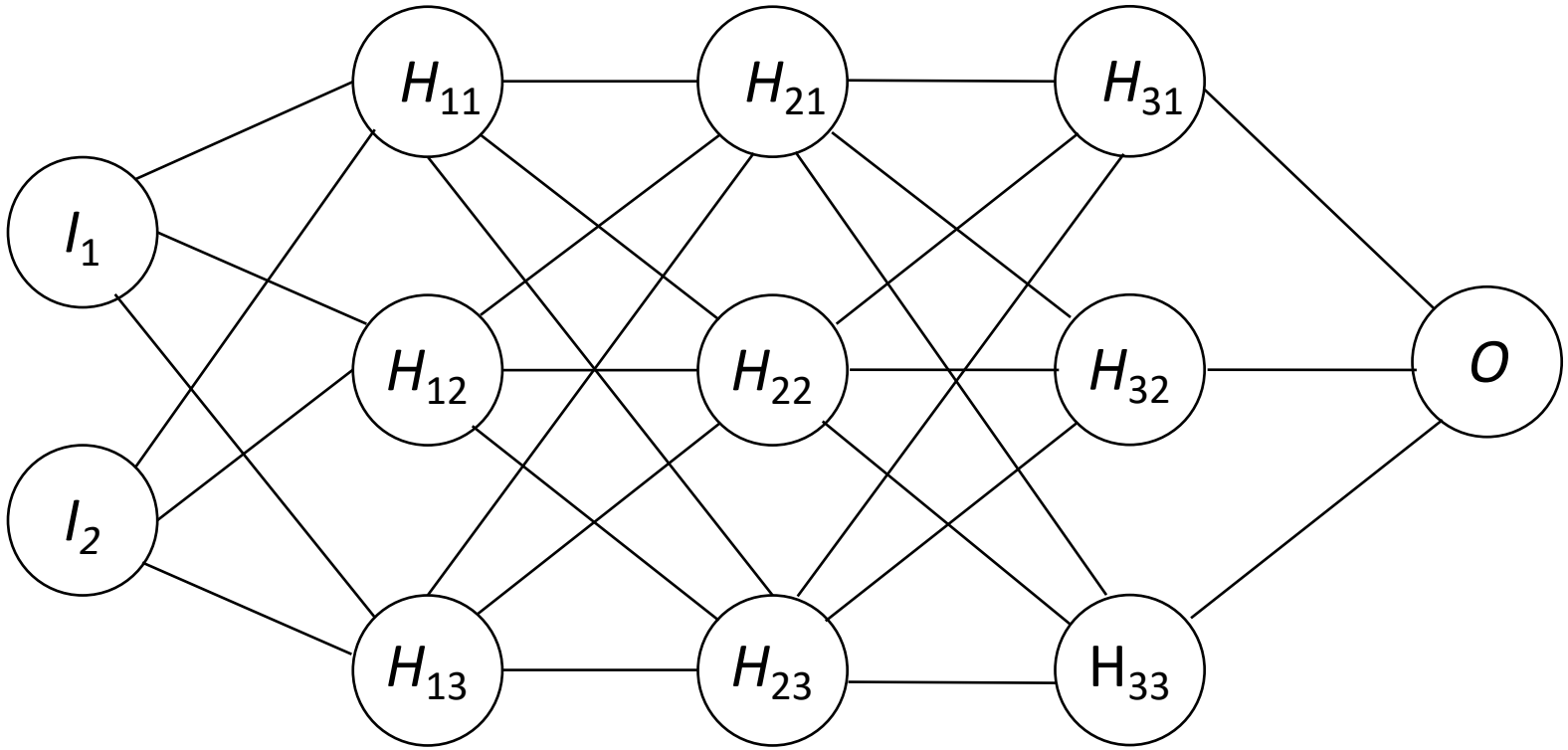
- If following features used:
  - Length of time of connection
  - Number of packets from destination
  - Number of data bytes from source

  Classification error less than 0.02%
- Identifying type of connection (like SMTP)
  - Best feature set omitted index, number of data bytes from destination (error rate 0.007%)
  - Other types of connections done similarly, but used different sets

Neural Nets

- Structure with input layer, output layer, at least 1 layer between them
- Each node (neuron) in layer connected to all nodes in previous, following layer
  - Nodes have an internal function transforming inputs into outputs
  - Each connection has associated weight
- Net given training data as input
  - Compare resulting outputs to ideal outputs
  - Adjust weights according to a function that takes into account the discrepancies between actual, ideal outputs
  - Iterate until actual output matches ideal output
  - Called “back propagation"

Neural Net



Neural net:

- 2 inputs I_1, I_2
- 3 hidden layers of 3 neurons each (H_{ij})
- 1 output O

Note neurons in a layer are not connected

Weights on connections not shown

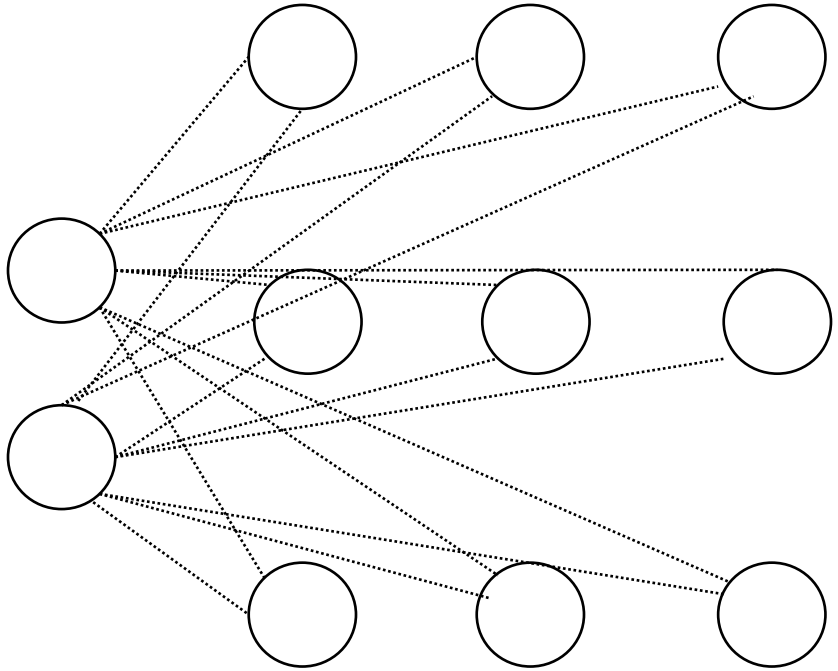
Example

- Neural nets used to analyze KDD-CUP-99 dataset
  - Dataset had 41 features, so neural nets had 41 inputs,, 1 output
  - Split into training data (7312 elements), test data (6980 elements)
- Net 1: 3 hidden layers of 20 neurons each; accuracy, 99.05%
- Net 2: 2 hidden layers of 40 neurons each; accuracy, 99.5%
- Net 3: 2 hidden layers, one of 25 neurons, other of 20 neurons; accuracy, 99%

Self-Organizing Maps

- Unsupervised learning methods that map nonlinear statistical relationships between data points to geometric relationships between points in a 2-dimensional map
- Set of neurons arranged in lattice, each input neuron connected to every neuron in lattice
  - Each lattice neuron given vector of *n* weights, 1 for each of *n* input features
- Input fed to each neuron
  - Neuron with highest output is “winner”; input classified as belonging to that neuron
  - Neuron adjusts weights on incoming edges so weights on edges with weaker values move to edges with stronger signals

Self-Organizing Maps



Self-organizing map with 2 inputs

Each input connected to each neuron in lattice

No lattice neurons connected to any other lattice neuron

Example

- SOM used to examining DNS, HTTP traffic on academic network
- For DNS, lattice of 19 x 25 neurons initialized using 8857 sample DNS connections
  - Tested using set of DNS traffic with known exploit injected, and exploit correctly identified
- For HTTP, lattice of 16 x 27 neurons initialized using 7194 HTTP connections
  - Tested using HTTP traffic with an HTTP tunnel through which *telnet* was run, and the commands setting up tunnel were identified as anomalous

Distance to Neighbor

- Anomalies defined by distance from neighborhood elements
  - Different measured used for this
- Example: *k* nearest neighbor algorithm uses clustering algorithm to partition data into disjoint subsets
  - Then computes upper, lower bounds for distances of elements in each partition
  - Determine which partitions are likely to contain outliers

Example

- Experiment looked at system call data from processes
  - Training data used to construct matrix with rows representing system calls, columns representing processes
  - Elements calculated using weighting taking into account system call frequency over all processes, and compensates for some processes using fewer system calls than others
- New process tested using a similarity function to compute distance to processes
  - *k* closest selected; average distance computed and compared to threshold
- Experiment tested values of *k* between 5, 25
  - *k* = 10, threshold value of 0.72 detected all attacks, false positive rate 0.44%
- Conclusion: this method could detect attacks with acceptably low false positive rate

Support Vector Machines

- Works best when data can be divided into 2 distinct classes
- Dataset has *n* features, so map each data point into *n*-dimensional space, each dimension representing a feature
- SVM is supervised machine learning method that splits dataset in 2
  - Technically, it derives a hyperplane bisecting the space
- Use *kernel function* to derive similarity of points
  - Common one: Gaussian radial base function (RBF), $e^{-\gamma\|x-y\|^2}$ where *x*, *y* are points, γ constant function, $||x–y||2 = \sum_{i=1}^{n}(x_i - y_i)^2$
- New data mapped into space, and so falls into either class

Example

- SVM used to analyze KDD-CUP-99 dataset
  - Dataset had 41 features, so used 41-dimensional space
  - Split into training data (7312 elements), test data (6980 elements)
- Accuracy of 99.5%
- SVM training time 18 seconds; neural net training time 18 minutes