

Introduction to Intrusion Detection

Auditing

Logging: the recording of events or statistics to provide information about system use and performance

Auditing: the analysis of log records to present information about the system in a clear and understandable manner

Problems

- what information to log
- what information to audit

Uses

- Describe security state
 - Determine if system enters unauthorized state
- Evaluate effectiveness of protection mechanisms
 - Determine which mechanisms are appropriate and working
 - Deter attacks because of presence of record

Components of an Auditing System

Logger – a mechanism that records information

- called “audit records”

Analyzer – analyzes the records to determine if the data needs to be changed and to detect some event or problem

Notifier – informs the analyst of the results of the audit

Audit Record Format

- Existing audit collection mechanisms often collect too much non-essential data and may not collect all of the data relevant to a specific tool
- However, in many cases all that is needed is an audit record preprocessor to translate an audit trail into the appropriate input for a specific tool
- Standards for audit record format have been proposed
 - POSIX
 - DNSIX

Definitions

Threat – The potential for deliberate and unauthorized

Access to information

Manipulation of information

Rendering a system inoperable or unreliable

Attack – A scenario or set of actions formulated to carry out a threat

Penetration or intrusion – A successful attack

Attack Prevention

- Authentication
 - Requires users to provide proof of identity
- Authorization
 - Grants access only to those who are authorized
- Cryptography
 - Protects transmitted data from eavesdropping & tampering

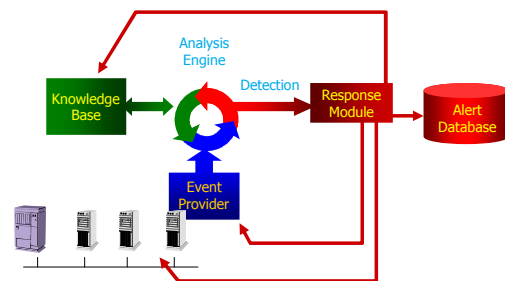
Despite Prevention Efforts

- Systems are not secure
- Solution is Intrusion Detection

Intrusion Detection Systems (IDSs)

- Provide an extra layer of defense
- Try to detect intruders by examining a time-ordered stream of events from a set of domains
 - Network (e.g., packets, streams)
 - Host (e.g., system calls, audit records)
 - Application (e.g., syslog, HTTP access logs)
- Act as an early warning system
- Produce alerts whenever an attack is detected

IDS Architecture



Three Classes of Users (Anderson '80)

- Masquerader
 - An individual who penetrates a system's authentication controls to exploit a legitimate user's account
- Mifeseasor
 - A legitimate user who participates in illicit activity
- Clandestine user
 - A user who seizes supervisory control of the system and operates below the level of audit collection or disables audit collection

Two Basic Analysis Techniques

- *Anomaly detection* (statistical, rule-based, specification-based)
 - Detects previously unknown attacks
- *Misuse detection* (signature analysis)
 - Detects only known attacks, needs continuous updating

Anomaly Detection (statistics, profiles, specs)

- Analyzes a set of characteristics of the system and compares their behavior with a set of expected behavior (*normal usage*)
- Reports when the computed statistics do not match the expected measurements
- Detects previously unknown attacks
- Difficult to configure (train), generates many false alarms
(*IDES, RST, ADAM*)

Intrusion Detection

CS177 2011 15

Misuse Detection (signature analysis)

- Models known attacks
- Reports when the modeled sequence of events occur
- Generates few false alarms
- Needs continuous updating
(*NFR, Emerald, Snort, STAT, Bro*)

Intrusion Detection

CS177 2011 16

Details of IDS Approaches

Anomaly

- Statistical-based
 - threshold
 - profile
- Rule-based
- Specification-based

Misuse

Intrusion Detection

CS177 2011 17

Statistical Anomaly Detection

- Measures the variations and type of audit data produced
- Can be applied to
 - individual user audit trails
 - all the audit records produced on a system
- Two types
 - Threshold detection
 - Profile-based detection

Intrusion Detection

CS177 2011 18

Threshold Detection

- Each occurrence of an event is recorded and the system detects when the number of occurrences surpasses what one might expect to occur normally
- Number of occurrences and analysis period is usually at discretion of the SSO
- Usually implemented as a small subcomponent because alone it is a poor detector of intrusions

Intrusion Detection

CS177 2011 19

Threshold Detection

Examples

- Large number of failed login attempts
- Extraordinary amount of deletions or references
- Unusually large number of data transfers

Intrusion Detection

CS177 2011 20

Profile-based Anomaly Detection

- Identifies variations between a user's past usage patterns and the user's most recent session
- May be at the user, group, system, or remote host level
- Main advantage is that it provides a means for detecting intrusions without *a priori* knowledge of the security flaw
- Very portable

Intrusion Detection

CS177 2011 21

Violations and Corresponding Anomalies

Masquerading

- Login times, locations, or connection types differ from that of the legitimate user
- Spend more time browsing
- Tend to raise more access denials and system errors

Intrusion Detection

CS177 2011 22

Attempted Break-ins

- Unusually large amount of failed logins

Misfeasance

- Larger than normal access denial records
- Successful access of objects owned by another user will produce anomalous access records

Intrusion Detection

CS177 2011 23

Illegal Dissemination of Information

- Use of remote printers
- Unusual time of printing
- Data transfers at unusual times
- Data transfers through unusual ports

Aggregation and Inference Attacks

- Unusually high number of data retrievals
- Unusual queries

Intrusion Detection

CS177 2011 24

Trojan Horse and Viruses

- Different amount of cpu cycles
- Different amount of i/o activity
- Abnormal modification to other executable files

Denial of Service

- Abnormally high activity of some resource for one user and abnormally low use by the other users

Intrusion Detection

CS177 2011 25

Issues that may hinder the effectiveness of profile-based approaches

- False positive rate
- False negative rate
- Gradual misbehavior
- Sporadic user environment

Intrusion Detection

CS177 2011 26

False Positive Rate

- For nearly every anomaly that corresponds to a violation there may be legitimate uses that could cause the same anomaly
- The higher the number of false positives the harder it is for the SSO to distinguish violations from false alarms
- Often deal in terms of suspicion ratings

False Negative Rate

- There are some penetrations that produce no identifiable anomaly

Intrusion Detection

CS177 2011 27

Gradual Misbehavior

- May avoid detection by gradually training the detector to accept misuse as acceptable behavior
- Group profiles may thwart this approach

Intrusion Detection

CS177 2011 28

Sporadic User Environments

- For some environments anomalous behavior will be the norm

e.g., university environment vs. bank teller

Intrusion Detection

CS177 2011 29

Rule-based Anomaly Detection

- Same advantages and disadvantages as profile-based anomaly detection
- Uses sets of rules to represent and store usage patterns
- Rules can be generated from historical audit data or entered manually
- Rules can express
 - Security policies
 - Administrative data

Intrusion Detection

CS177 2011 30

Rule-based Anomaly Detection (continued)

- Rules represent the past behavior patterns of users, privileges, time slots, etc.
- May focus on individual events or on sequence of events

Intrusion Detection

CS177 2011 31

Specification-based Anomaly Detection

- Same advantages and disadvantages as profile-based anomaly detection
- Uses specifications of how a program is supposed to behave
- Specifications can be written by the developer or can be generated by static analysis of the code
- Specifications can express
 - system call sequences
 - storage usage
 - allowable connections

Intrusion Detection

CS177 2011 32

Misuse Detection (Signature Analysis)

- A system whose rule base consists of rules that fire when audit records are parsed that appear to indicate illegal or suspicious user or system activity
- May recognize single audit events or sequences of audit events
- Rule base is very machine specific
- Nice complement to anomaly detection

Misuse Detection (continued)

- A weakness is the dependency on audit records to match and bind to the rules in the knowledge base
- For a given scenario there may be several potential audit sequences that will produce slight variations of the same penetration
 - A minor variation may slip by unnoticed

Challenges In Intrusion Detection

- Recognize malicious actions in the huge stream of events provided by network monitors and OS/application auditing facilities
- Keep-up with the increasing speed of networks
- Detect intrusions in real-time
- Perform detection at different abstraction levels
- Aggregate detection results gathered by distributed sensors
- Deal with sources characterized by different trust levels
- Correlate detection results within and across security domains
- Integrate different systems so that all techniques (anomaly, misuse) and domains (host, network, application) are covered

Challenges In Intrusion Detection

- Deploy ID systems in very different (distributed) environments
- Take into account the characteristics of the protected computer network
- Provide language-level support across systems
- Reconfigure sensors to adjust to changing environment, security requirements, etc.


Three Real Systems

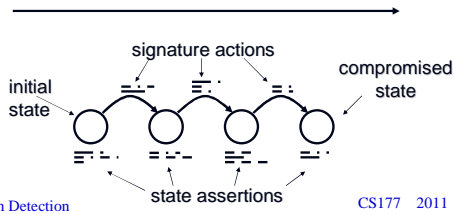
- STAT: State Transition Analysis Technique
 - Misuse-based
- WebAnomaly
 - Anomaly-based
- Alert Correlation Framework
 - Get the “big picture”

State Transition Analysis Technique

- STAT models penetrations as a sequence of state transitions
- Represents only key activities that lead from an initial safe state to a final compromised state
 - Signature Actions
 - State Assertions

State Transition Diagrams

Attacker has limited privileges  Attacker illicitly gains more privileges

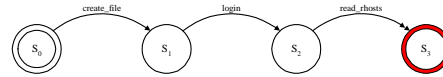


Intrusion Detection

CS177 2011 39

USTAT example ftp-write

- Exploit
 - use ftp to create .rhosts file in world-writable ftp home directory
 - rlogin using bogus .rhosts file



Intrusion Detection

CS177 2011 40

STATL

- A STATL specification is the description of a complete attack scenario (a signature) in terms of states and transitions
- Domain-independent language
 - Extensions for
 - IP networks
 - Solaris BSM
 - WinNT event logging facility
 - Apache event logs
 - Syslog facility
 - IDMEF Alerts
- Parameterized descriptions
 - Generic attacks customizable by installation or policy

Intrusion Detection

CS177 2011 41

STATL Basic Abstractions

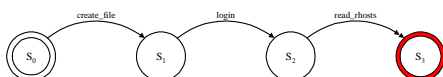
- Scenarios
 - States
 - Transitions (consuming, nonconsuming, unwinding)
 - Signature actions
 - Assertions
 - Global environment
 - Local environment
 - Code blocks
- Events
 - Defined as trees of generic events encapsulating domain-specific opaque events
- Timers

Intrusion Detection

CS177 2011 42

USTAT Example ftp-write

- Exploit
 - use ftp to create .rhosts file in world-writable ftp home directory
 - rlogin using bogus .rhosts file



Intrusion Detection

CS177 2011 43

ftp-write in STATL

```

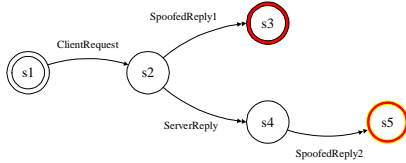
use ustat;
scenario ftp_write
{
  int user, pid, inode;
  string objname;
  initial state s0 {
    transition create_file (s0 -> s1) nonconsuming
    {
      [WRITE w]: (w.euid != 0) && (w.owner != w.ruid)
      {
        inode = w.inode;
        objname = w.objname;
      }
    }
  }
  state s1 {
    transition login (s1 -> s2) nonconsuming
    {
      [EXECUTE e]: match_name(e.objname, "login")
      {
        user = e.ruid;
        pid = e.ppid;
      }
    }
  }
  state s2 {
    transition read_rhosts (s2 -> s3) consuming
    {
      [READ r]: (r.pid == pid) && (r.inode == inode)
      {
        string username = userfd2name(user);
        log("%d: by user %s using %s", user, username, objname);
      }
    }
  }
  state s3 {
  }
}

```

Intrusion Detection

CS177 2011 44

NetSTAT Example UDP-race



UDP-race in STATL

```

use statl;
scenario UDP-race
{
  Host server, tracer;
  Service s1;
  IPAddress a_s, a_c, c;
  Interface I_c;

  IPAddress request_ip_src, request_ip_dst;
  Port request_udp_src, request_udp_dst;

  <? CONSTRAINT
  { server in Network.hosts } &&
  { a in server.services } &&
  { protocol == "UDP" } &&
  { authentication == "IPaddress" } &&
  { a_s in a.ipAddresses } &&
  { a_c in a.trustedAddrs } &&
  { a_c interface in ProtectedNetwork.Interfaces } &&
  { tracer in server } &&
  { a_c in tracer.ipAddresses } &&
  I_c in tracer.Interfaces
  ?>

  state S3 { ! log "compromised"; }
  state S5 { ! log "under attack"; }

  transition ClientRequest (S1 -> S2) nonconsuming
  {
    [IPDatagram d1 (IPDatagram u1)]
    <? ENDPOINT_PORTS a_c.interface, a_c.interface ?> :
    { d1.src == a_s } && { d1.dst == a_c } && { d1.dst == x.port }
    {
      request_ip_src = d1.src;
      request_ip_dst = d1.dst;
      request_udp_src = u1.src;
      request_udp_dst = u1.dst;
    }
  }

  transition ServerReply (S2 -> S4) consuming
  {
    [IPDatagram d2 (IPDatagram u2)]
    <? ENDPOINT_PORTS a_c.interface, a_c.interface ?> :
    { d2.src == request_ip_dst } &&
    { d2.dst == request_ip_src } &&
    { d2.src == request_udp_dst } &&
    { d2.dst == request_udp_src }
  }

  action SpooferReply
  {
    [Message m2 (IPDatagram d2 (IPDatagram u2))]
    <? ENDPOINT_PORTS I_c, a_c.interface ?> :
    { d2.src == request_ip_dst } &&
    { d2.dst == request_ip_src } &&
    { d2.dst == request_udp_dst } &&
    { m2.src == request_udp_dst }
  }

  transition SpooferReply1 (S2 -> S3) consuming [SpooferReply]
  transition SpooferReply2 (S4 -> S5) consuming [SpooferReply]
}

```

STATL Execution Model

- A STATL scenario has a runtime representation in terms of
 - Specification (global environment and STD definition)
 - Instances (local environment, occurrence of an attack)
- Event matching and assertions determine which enabled transitions fire
- Scenario evolution determined by transition type
 - *Nonconsuming*: New instance in new state and current instance stays in previous state
 - *Consuming*: Current instance changes its state
 - *Unwinding*: Backtracking to ancestor instance, possibly removing a subtree of the instance tree

OK, You Can Develop Your Own IDS, But...

- What if one wants to change the configuration of a sensor at run time, without having to stop the whole thing?
- How can one be sure that all the pieces (extensions, providers, scenarios) fit together?
- What if one wants to control a multitude of sensors deployed throughout the network?
- What if one wants to aggregate/fuse/correlate the alerts produced by the deployed sensors?

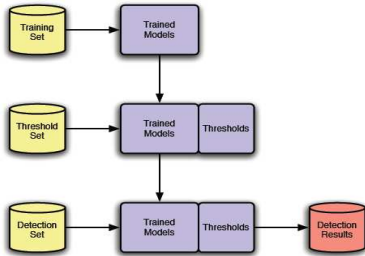
MetaSTAT

- A communication and control infrastructure for STAT-based sensors
- CommSTAT communication infrastructure allows for the exchange of alerts and control commands over secure connections
 - MetaSTAT Controller dispatches commands to the sensors
 - The STAT Proxy mediates communication
 - Performs local module management (installation/configuration)
 - Relays commands to sensors (loading/activation)

MetaSTAT

- MetaSTAT Configurator manages sensors
 - Database of available modules and corresponding dependencies
 - Database of current sensor configurations
 - Allows the security officer to submit reconfiguration requests
 - Checks for the meaningfulness of reconfiguration
- MetaSTAT Collector component aggregates sensor alerts in a centralized database to support analysis and correlation

Anomaly Detection Phases



Intrusion Detection

CS177 2011 57

Web-based Anomaly Detection

- Examines web requests sent from clients to server
 - Application to be executed
 - Application parameters (attribute names and values)

Example

```
GET /cgi-bin/show.cgi?sID=12345&file=images/foo.png
```

- Applies statistical models to each attribute of each application in two phases

Learning phase:

Builds profiles of normal behavior for each application parameter

Detection phase:

Detects deviations from learned profile

Intrusion Detection

CS177 2011 58

Web-based Anomaly Detection

- Examines web requests sent from clients to server
 - Application to be executed
 - Application parameters (attribute names and values)

Example

```
GET /cgi-bin/show.cgi?sID=12345&file=images/foo.png
```

- Applies statistical models to each attribute of each application in two phases

Learning phase:

Builds profiles of normal behavior for each application parameter

Detection phase:

Detects deviations from learned profile

Intrusion Detection

CS177 2011 59

Web-based Anomaly Detection

- Examines web requests sent from clients to server
 - Application to be executed
 - Application parameters (attribute names and values)

Example

```
GET /cgi-bin/show.cgi?sID=12345&file=images/foo.png
```

- Applies statistical models to each attribute of each application in two phases

Learning phase:

Builds profiles of normal behavior for each application parameter

Detection phase:

Detects deviations from learned profile

Intrusion Detection

CS177 2011 60

Web-based Anomaly Detection

- Examines web requests sent from clients to server
 - Application to be executed
 - Application parameters (attribute names and values)

Example

```
GET /cgi-bin/show.cgi?sID=12345&file=images/foo.png
```

- Applies statistical models to each attribute of each application in two phases

Learning phase:

Builds profiles of normal behavior for each application parameter

Detection phase:

Detects deviations from learned profile

Intrusion Detection

CS177 2011 61

Detection Models

- String length
- String character distribution
- Structural inference
- Token finder

Intrusion Detection

CS177 2011 62

String Length Model

- String variables often show regular behavior with respect to length
 - User ids
 - Search strings
 - Applications often read, write, and execute files in a few locations in filesystem
- Training
 - Compute mean and variance of observed argument lengths
- Detection
 - Chebyshev inequality

$$p(l) = p(|x - \mu| > |l - \mu|) = \frac{\sigma^2}{(l - \mu)^2}$$

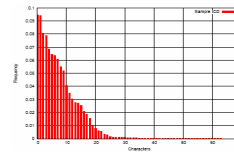
Intrusion Detection

CS177 2011 63

String Character Distribution Model

Observation

Many strings take values that have similar character distributions



- Model creates idealized character distribution (ICD) for strings observed during the training phase
- Anomaly score calculated using variant of Pearson Chi-squared test

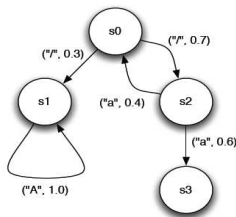
Intrusion Detection

CS177 2011 64

Structural Inference Model

Observation

Many attribute values can be modeled as strings generated by a regular grammar



- Model constructs probabilistic grammar from learning set
- Anomaly score calculated as product of transition probabilities along path through NFA for a given value

Intrusion Detection

CS177 2011 65

Token Finder Model

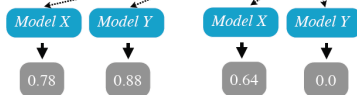
- Repeated values often appear at monitoring interfaces
 - Some parameters take on values drawn from a small enumeration, others may rarely repeat parameters
 - Token finder model detects this difference
- Training
 - Store parameter value history
 - If argument values are drawn from a limited set of possibilities, model stores this enumeration
 - If not, model asserts "no enumeration"
- Evaluation
 - Model returns score of 1 or 0 only

Intrusion Detection

CS177 2011 66

WWW Requests

```
169.229.60.105 - [6/Nov/2002:23:59:59 -0800]
"GET /scripts/access.pl?user=johndoe&cred=admin" 200 2122
```

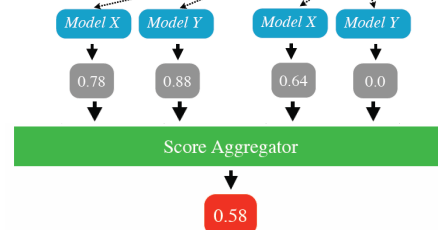


Intrusion Detection

CS177 2011 67

Multi-score Aggregation

```
169.229.60.105 - [6/Nov/2002:23:59:59 -0800]
"GET /scripts/access.pl?user=johndoe&cred=admin" 200 2122
```



Intrusion Detection

CS177 2011 68

Multi-score Aggregation

- Multiple models per event imply a need for score combination
- The simple approach (weighted summation) cannot represent dependencies between models
 - When one feature is anomalous, another feature may be expected to be anomalous as well
 - A human-readable string that contains binary data is likely to also violate its normal structure
 - An anomalous feature might indicate that the quality of another model output increases
 - An abnormally long string increases confidence in corresponding character distribution
- Dependencies can be represented using Bayesian decision networks

Intrusion Detection

CS177 2011 69

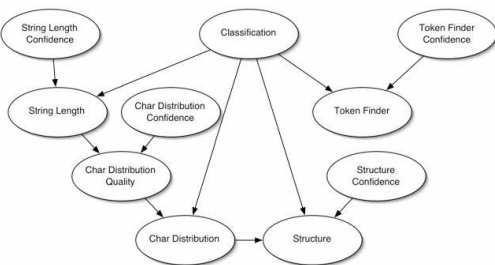
Bayesian Models

- Graphical models model a domain of uncertainty
- Directed acyclic graph
 - Node: discrete random variable of interest
 - Edge: a causal relationship between two variables
- Conditional probability table (CPT)
 - Defines the probability of each state of a random variable given the state of its parent nodes

Intrusion Detection

CS177 2011 70

Bayesian Network Example



Intrusion Detection

CS177 2011 71

Limitations of Anomaly Detection

- Anomaly detectors prone to producing false positives
- Anomaly detectors give no indication as to the nature of a detected attack

Intrusion Detection

CS177 2011 72

Webanomaly Summary

- Multi-model approach fundamental to detecting a variety of attacks
- Capturing inter-model dependencies improves the accuracy of combining multiple model scores
- Major limitations of anomaly detectors can be mitigated through use of generalization and characterization techniques

“Using Generalization and Characterization Techniques in the Anomaly-based Detection of Web Attacks,” NDSS, Feb. 2006

Intrusion Detection

CS177 2011 73

Intrusion Detection Systems Summary

- Provide an extra layer of defense
- Try to detect intruders by monitoring system activity
 - Network packets
 - OS audit records
 - Application log files
- Act as an early warning system
- Produce alerts whenever an attack is detected

Intrusion Detection

CS177 2011 74

Problems with IDSs

- Large number of alerts
- Large number of false positives
- Large number of non-relevant positives
- Alerts do not contain enough information
 - Attribute values are ‘unknown’

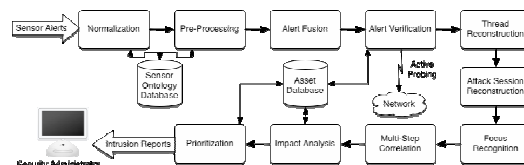
Alert Correlation Systems

- Try to fix the problems of IDSs
- Process the output of other sensors
- Key ideas
 - Reduce the number of alerts
 - Reduce false and non-relevant positives
 - Find higher-level attack patterns
 - Prioritize alerts
- Goal is to get the “big picture” of what is happening

Our Approach to Correlation

- Identified the different parts of the correlation process
- Developed a set of correlation components
- Developed a framework for managing the correlation components
- Result is
 - A comprehensive correlation process composed of clearly defined components
 - Each component contributes differently to the correlation goals

Our Correlation Process



Example Attack

- Vulnerable Apache web server on 10.0.0.1
 - Runs host based IDS (H)
 - Application based IDS (A)
- Two Network based IDSs (N1 and N2) monitor the network
- Attacker is at 31.3.3.7

Example Attack Timeline

- Attacker scans victim host
- A worm tries to exploit an IIS vulnerability
- Attacker exploits apache vulnerability
- Attacker escalates to root
 - Takes two tries

Example Attack Alerts

IIS Exploit S: 80.0.0.1 T:10.0.0.1:80 Sensor: N1	Scanning S: 31.3.3.7 T:10.0.0.1 Sensor: N2	Portscan S: 31.3.3.7 T:10.0.0.1 Sensor: N1	Apache Exploit S: 31.3.3.7 T:10.0.0.1:80 Sensor: N1
Bad Request S: T:localhost Sensor: A	Local Exploit S: T:linuxconf Sensor: H	Local Exploit S: T:linuxconf Sensor: H	

Intrusion DetectionCS177 2011 81

Algorithms Used

- Normalization
- Pre-Processing
- Alert Fusion
- Alert Verification
- Thread Reconstruction
- Attack Session Reconstruction
- Focus Recognition
- Multi-Step Correlation
- Impact Analysis
- Prioritization

Intrusion DetectionCS177 2011 82

Normalization

- Transforms alerts to a standardized format
 - Different alert formats are used by different sensors
 - IDMEF is a syntactic standard, not an ontology
- Utilizes an adaptor module to interface with the sensors

Intrusion DetectionCS177 2011 83

Effect of Normalization

IIS Exploit S: 80.0.0.1 T:10.0.0.1:80 Sensor: N1	Scanning S: 31.3.3.7 T:10.0.0.1 Sensor: N2	Portscan S: 31.3.3.7 T:10.0.0.1 Sensor: N1	Apache Exploit S: 31.3.3.7 T:10.0.0.1:80 Sensor: N1
Bad Request S: T:localhost Sensor: A	Local Exploit S: T:linuxconf Sensor: H	Local Exploit S: T:linuxconf Sensor: H	

Intrusion DetectionCS177 2011 84

Effect of Normalization

IIS Exploit S: 80.0.0.1 T:10.0.0.1:80 Sensor: N1	Portscan S: 31.3.3.7 T:10.0.0.1 Sensor: N2	Portscan S: 31.3.3.7 T:10.0.0.1 Sensor: N1	Apache Exploit S: 31.3.3.7 T:10.0.0.1:80 Sensor: N1
Bad Request S: T:localhost Sensor: A	Local Exploit S: T:linuxconf Sensor: H	Local Exploit S: T:linuxconf Sensor: H	

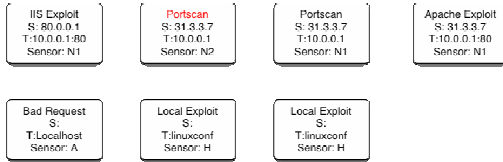
Intrusion DetectionCS177 2011 85

Pre-processing

- Operates on normalized alerts
- Adds missing attributes
 - Time-stamps
 - Source IPs of host-based alerts
 - Attack type
- Utilizes a set of rules to generate missing attributes

Intrusion DetectionCS177 2011 86

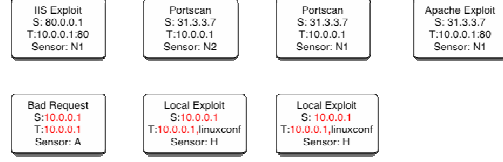
Effect of Pre-processing



Intrusion Detection

CS177 2011 87

Effect of Pre-processing



Intrusion Detection

CS177 2011 88

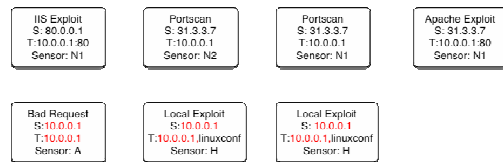
Alert Fusion

- Combines alerts representing the same attack occurrence
 - Removes duplicate alerts
- Utilizes a sliding time window
- Fuses alerts only if the overlapping attributes are identical

Intrusion Detection

CS177 2011 89

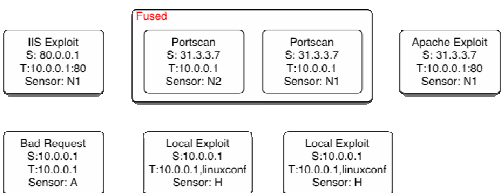
Effect of Fusion



Intrusion Detection

CS177 2011 90

Effect of Fusion



Intrusion Detection

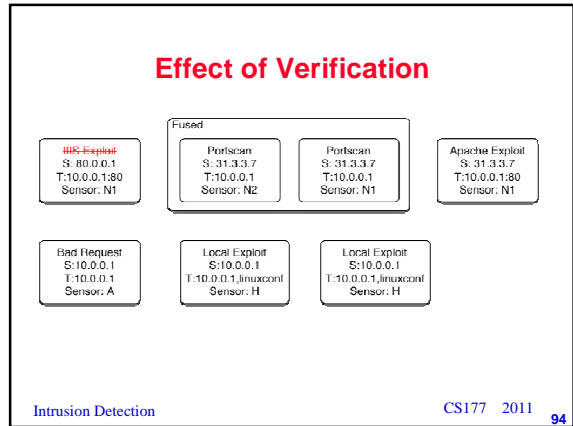
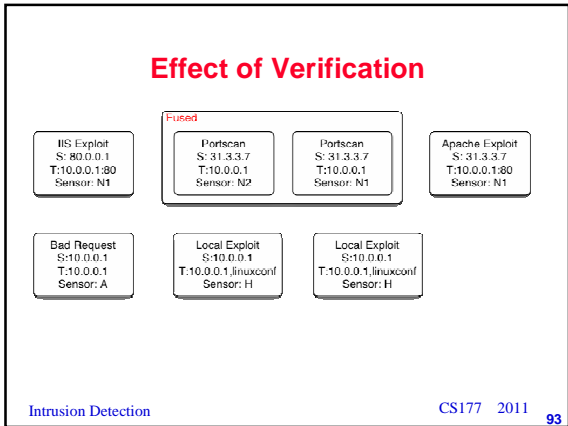
CS177 2011 91

Alert Verification

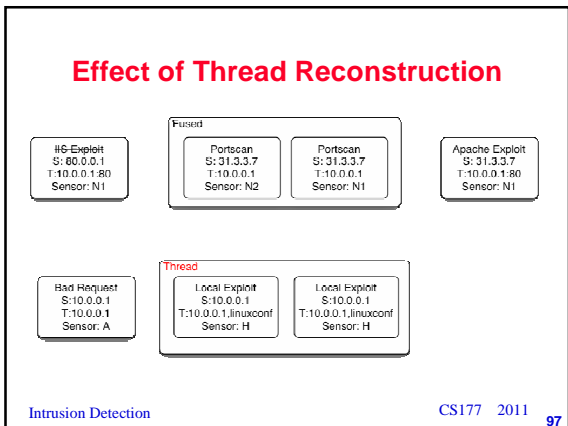
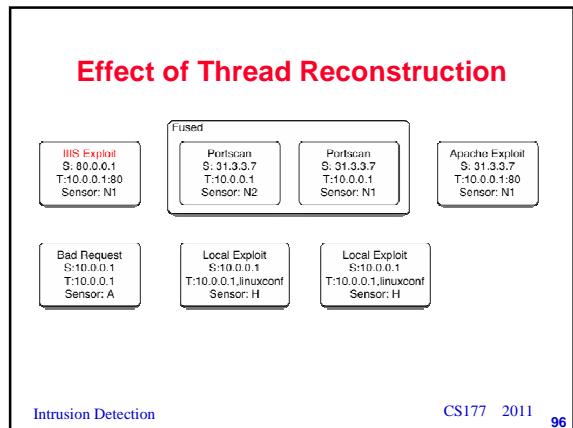
- Tries to verify that a reported attack is successful
- Suppresses false positives
- Several techniques
 - Utilize information about installed services
 - On-demand vulnerability scan
 - “Negative” rules

Intrusion Detection

CS177 2011 92

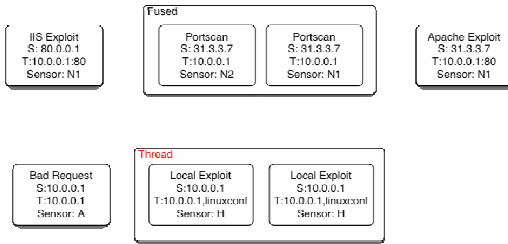


- ### Thread Reconstruction
- Combines attacks by one attacker against a single target
 - Reduces alerts caused by an attacker that runs the same attack numerous times
 - Offset brute-forcing
 - Password guessing
 - Denial of Service
 - Merges alerts with matching source and destination address (within a time window)
- Intrusion Detection CS177 2011 95



- ### Attack Session Reconstruction
- Links network-based and host-based alerts
 - Needs more information than provided by the alerts analyzed
 - Network-based alerts contain IP addresses & network ports
 - Host-based alerts contain path, UID & PID
 - Additional information can be provided by a DB or by on-demand probing
- Intrusion Detection CS177 2011 98

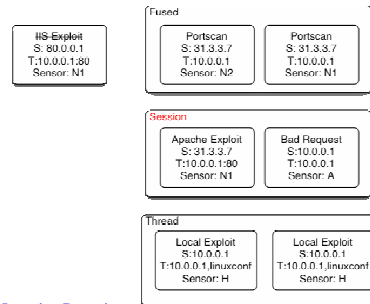
Effect of Session Reconstruction



Intrusion Detection

CS177 2011 99

Effect of Session Reconstruction



Intrusion Detection

CS177 2011 100

Focus Recognition

- Identifies hosts that are either the source or target of a large amount of alerts
- Consists of two sub-components
 - 1-N and N-1
- Efficiently identifies worms and DDoS attacks

Intrusion Detection

CS177 2011 101

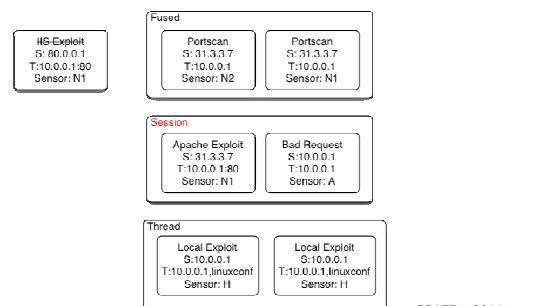
Multi-step Correlation

- Infers higher-level attack patterns
 - Scan, break-in, escalation
 - Island hopping
- Could be implemented using requires/provides modeling

Intrusion Detection

CS177 2011 102

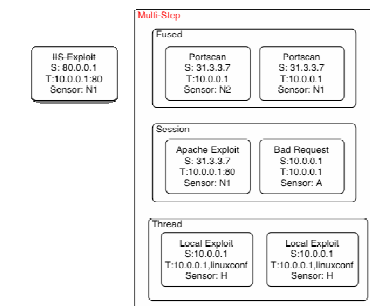
Effect of Multi-step Correlation



Intrusion Detection

CS177 2011 103

Effect of Multi-step Correlation



Intrusion Detection

CS177 2011 104

Impact Analysis

- Determines impact of attack on the protected network
- Utilizes an asset database and heartbeat monitors on important assets
- Analyzes network service dependencies, e.g., mail servers need DNS
- Links service failures to attacks

Intrusion Detection

CS177 2011 105

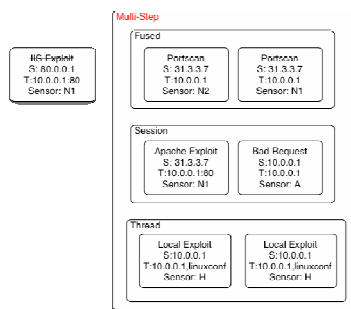
Prioritization

- Enables the operator to quickly identify the important alerts
- Dependent on the security policy of the site
 - Most sites ignore reports of portscans
 - Scans could be extremely important in a heavily firewalled network

Intrusion Detection

CS177 2011 106

Result of Correlation Process



Intrusion Detection

CS177 2011 107