Alert Verification

Determining the Success of Intrusion Attempts

Christopher Kruegel Technical University Vienna
William Robertson University of California, Santa Barbara

Overview

- Motivation
  - problem of *irrelevant* alerts
- Alert verification
  - verify success of attack
  - passive and active mechanisms
- Prototype
  - Snort extension
- Evaluation
- Conclusions

Motivation

- Intrusion detection systems produce large amounts of alerts
- Often, administrators ignore these alerts because
  - there are too many of them
  - there are too many *irrelevant* ones
- Two main strategies to reduce alerts
  1. combine, summarize, and correlate alerts
  2. remove irrelevant alerts (or greatly reduce their priorities)

Irrelevant Alerts

- Alert classification
  - Type 1 (true positive)
    alert raised in response to successful attack
  - Type 2 (non-relevant positive)
    alert raised in response to actual attack that failed its objectives
  - Type 3 (false positive)
    alert raised in response to benign event

Irrelevant Alerts

Irrelevant Alerts

Non-relevant positive example

- Infected machine launches a Code Red attack against Apache web server (running on Linux host)
- Intrusion detection system (IDS) faithfully reports attack

Problem

- IDS reports an actual attack (cannot call it a false alarm)
- However, target host is not vulnerable (cannot call it a relevant attack)
- Even worse when web server is a Microsoft IIS, but it is patched

Alert Verification

- Alert verification
  - *process of verifying the success of attacks*
  - allows IDS to distinguish between true positives (Type 1 alerts) and non-relevant and false positives (Type 2 and Type 3 alerts)
  - allows IDS to suppress an alert or reduce its priority

- Requirements
  - accuracy
    - the alert verification process should correctly tag all successful and unsuccessful alerts
    - quality of input data
    - timeliness of input data

Alert Verification

- Requirements (cont.)
  - low impact
    - the verification process should not interfere with regular operations
  - ease-of-use

- Classification
  - according to verification technique
    1. context-based technique
    2. forensics-based technique

  - according to point in time when verification data is gathered
    1. passive alert verification
    2. active alert verification

Alert Verification Techniques

- Context-based verification
  - model properties of networks and hosts
  - model requirements of attacks (based on these properties)
  - check whether an attack can possibly success, given a particular network configuration
  - example
    - host operating system is a modeled property
    - Code Red attack requires a Microsoft Windows target
    - attacks against Linux hosts can be suppressed
  - related work
    - M2D2 [Morin, 2002]
    - Real-time Network Awareness [Roesch, 2003]

Alert Verification Techniques

- Forensics-based verification
  - check for known *outcome* of attacks
  - checkable and visible traces of attacks
  - known outcome has to be defined for attacks similar to misuse-based IDS signatures or virus signatures
  - example
    - worm is known to create a certain Windows Registry entry
  - related work
    - Cisco IDS [2004]

Alert Verification Classification

- Passive alert verification
  - gather context information once (or at regular intervals)
  - information is available previously to attack

- Active alert verification
  - gather context information or forensic data after alert is generated
  - information is gathered in response to attack
  - mechanisms can be divided into following groups
    - active with remote access
    - active with authenticated access
    - active with dedicated sensor support

Passive Alert Verification

- *A priori* information about
  - host operating system, services and configuration, and network topology
- Possibility to check
  - if target host and service exist,
  - if service is reachable, and
  - if service is potentially vulnerable

+ basically no impact on network operations
+ can be managed at network level (no host support needed)
- database of network and hosts must be created and maintained
- information can be stale (i.e., out-of-date)
- limitations to the amount of information that can be gathered

Active Alert Verification

With remote access

- a network connection to the target of the attack is needed
- allows active scanning in response to attack

- Information can be gathered about
  - status and changes of services (using also passive information)
  - actual vulnerabilities
- Vulnerability scanner
  - checks remotely for vulnerabilities
  - often ships with a large database of checks that can be performed

Active Alert Verification

- \+ information is current
- \+ can be managed at network level (no host support needed)
- \+ large amount of checks already exist
- \- possible impact on network operations and services
  - bandwidth consumption and service crashes
- \- vulnerability scanner is not completely accurate

- Vulnerability scanner can produce
  - – false positives (no loss compared to IDS only)
  - – false negatives (problematic, but unlikely as a vulnerability scanner performs a basic variation of corresponding attack)

Active Alert Verification

With authenticated access

- – verification process disposes of local (user) access to target host
- – run scripts and system commands

- Information gathered about
  - – file integrity or existence of suspicious files
  - – system status about processes and network connections

+ current and accurate information
+ basically no impact on network operations
- requires host support
- checks have to be developed

Active Alert Verification

<u>With dedicated sensor support</u>

- verification process disposes of local (user) access to target host
- dedicated sensors are installed and configured

- Information gathered about
  - kernel level events, system calls

\+ current information
\+ high-quality audit data
\+ basically no impact on network operations
\- requires sensors to be installed and configured
\- checks have to be developed

Prototype

- Active alert verification prototype
  - uses the remote access technique
  - based on NASL scripts written for Nessus vulnerability scanner
  - implemented as a patch to Snort IDS

- Nessus
  - widely-used, open source vulnerability scanner
  - many high quality checks available
  - very modular and easy to integrate
  - extensible NASL (Nessus Attack Scripting Language) language

Prototype

- Snort patch
  - extension of Snort's alert processing pipeline
  - intercepts alerts before being passed to output plug-ins
  - multiple verification threads
    - ensures high throughput if checks are waiting for time outs

- Selection of appropriate vulnerability check
  - based on CVE ID
  - both defined by Snort alerts and NASL scripts
  - when no matching script is found, alert remains unverified and is simply passed on

- All alerts are appropriately tagged and passed to output plug-ins

Prototype

- Snort-AV prototype system
  - no setup overhead
    - as easy as setting up Snort
  - covers a significant fraction of Snort alerts
  - well maintained
    - patch against latest Snort version 2.1.3
  - reasonably popular
    - about 5.000 downloads
  - readily available

http://www.cs.ucsb.edu/~wkr/projects/ids_alert_verification/

Evaluation

- Synthetic benchmark
  - Snort-AV on a test bed with an attacker and a victim host
  - evaluation set consisting of
    1. nine working exploits against popular services such as Apache, bind, sshd, sendmail, wu-ftpd
    2. full scan using Nessus
  - Snort generated 6,659 alerts, of which only 24 alerts were relevant
  - among those 24 relevant alerts were all nine exploits
  - all 24 relevant alerts were correctly verified, the rest was suppressed

Evaluation

- Real world benchmark
  - Snort-AV with two honeypots
    - Snort-2.0.2
    - Linux RedHat 7.2
    - Windows 2000
  - during a 14 days period
    - 164.415 alerts in response to attacks against RedHat 7.2
    - 79.198 alerts in response to attacks against Windows 2000
  - verification process results
    - 161.166 attacks (98.3%) against RedHat 7.2 tagged as unsuccessful
    - 78.785 attacks (99.4%) against Windows 2000 tagged as unsuccessful

Evaluation

- Real world benchmark (cont.)
  - most attacks were
    - Slammer and Nachia worms
    - scan activity against ports commonly used by web proxy and socks proxy
  - unsuccessful attacks were manually checked
    - possible because many attacks target non-existing services
  - significant fraction of alerts were non-relevant positives
    - despite the fact that an out-of-the-box Snort was used
- Limitations
  - alert verification quality depends on quality of Nessus
  - CVE ID sometimes imprecise

Conclusions

- Real world systems produce a large amount of alerts
  - in particular, non-relevant positives are a problem
- Alert verification is a process that determines the success of attacks to suppress irrelevant alerts
- Classification
  - context-based versus forensics-based techniques
  - passive versus active verification techniques
- Snort-AV
  - prototype of an active alert verification system with remote access
  - integrates the Nessus vulnerability scanner into the Snort IDS
  - effective in synthetic and real world experiments