Accountable Virtual Machines

Andreas Haeberlen
University of Pennsylvania

Paarijaat Aditya   Rodrigo Rodrigues   Peter Druschel
Max Planck Institute for Software Systems (MPI-SWS)

Max
Planck
Institute
for
Software Systems
Scenario: Multiplayer game

- Alice decides to play a game of Counterstrike with Bob and Charlie

I'd like to play a game
What Alice sees
Could Bob be cheating?

- In Counterstrike, ammunition is local state
  - Bob can manipulate counter and prevent it from decrementing
  - Such cheats (and many others) do exist, and are being used
This talk is not (just) about cheating!

- Cheating is a serious problem in itself
  - Multi-billion-dollar industry

- But we address a more general problem:
  - Alice relies on software that runs on a third-party machine
  - Examples: Competitive system (auction), federated system...
  - How does Alice know if the software running as intended?
Goal: Accountability

- We want Alice to be able to
  - Detect when the remote machine is faulty
  - Obtain evidence of the fault that would convince a third party

- Challenges:
  - Alice and Bob may not trust each other
    - Possibility of intentional misbehavior (example: cheating)
  - Neither Alice nor Bob may understand how the software works
    - Binary only - no specification of the correct behavior
Problem: Detecting faults on remote machines
  - Example: Cheating in multiplayer games

Solution: Accountable Virtual Machines

Evaluation
  - Using earlier example (cheating in Counterstrike)

Summary
Bob runs Alice's software image in an AVM
- AVM maintains a log of network in-/outputs

Alice can check this log with a reference image
- AVM correct: Reference image can produce same network outputs when started in same state and given same inputs
- AVM faulty: Otherwise
Tamper-evident logging

- Message log is tamper evident [SOSP'07]
  - Log is structured as a hash chain
  - Messages contain signed authenticators

- Result: Alice can either...
  - ... detect that the log has been tampered with, or 😊
  - ... get a complete log with all the observable messages 😊
Execution logging

How does Alice know whether the log matches a correct execution of her software image?

Idea: AVMM can specify an execution
  - AVMM additionally logs all nondeterministic inputs
  - AVM correct: Can replay inputs to get execution
  - AVM faulty: Replay inevitably (!) fails

© 2010 Andreas Haeberlen
Auditing and replay

371: SEND(Alice, Firing)
370: SEND(Alice, Firing)
369: SEND(Alice, Firing)
368: Mouse button clicked
367: SEND(Alice, Got medipack)
366: Mouse moved left

373: SEND(Alice, Firing)
372: SEND(Alice, Firing)
371: SEND(Alice, Firing)
370: SEND(Alice, Firing)
369: SEND(Alice, Firing)
368: Mouse button clicked
367: SEND(Alice, Got medipack)
366: Mouse moved left
...
AVM properties

- Strong accountability
  - Detects faults
  - Produces evidence
  - No false positives

- Works for arbitrary, unmodified binaries
  - Nondeterministic events can be captured by AVM Monitor

- Alice does not have to trust Bob, the AVMM, or any software that runs on Bob's machine
  - If Bob tampers with the log, Alice can detect this
  - If Bob's AVM is faulty, ANY log Bob could produce would inevitably cause a divergence during replay
Problem: Detecting faults on remote machines
- Example: Cheating in multiplayer games

Solution: Accountable Virtual Machines

Evaluation
- Using earlier example (cheating in Counterstrike)

Summary
Methodology

- We built a prototype AVMM
  - Based on logging/replay engine in VMware Workstation 6.5.1
  - Extended with tamper-evident logging and auditing

- Evaluation: Cheat detection in games
  - Setup models competition / LAN party
  - Three players playing Counterstrike 1.6
  - Nehalem machines (i7 860)
  - Windows XP SP3
Evaluation topics

- Effectiveness against real cheats
- Overhead
  - Disk space (for the log)
  - Time (auditing, replay)
  - Network bandwidth (for authenticators)
  - Computation (signatures)
  - Latency (signatures)
- Impact on game performance
- Online auditing
- Spot checking tradeoffs
  - Using a different application: MySQL on Linux

Please refer to the paper for additional results!
AVMs can detect real cheats

If the cheat needs to be installed in the AVM to be effective, AVM can trivially detect it

- Reason: Event timing + control flow change
- Examined real 26 cheats from the Internet; all detectable
AVMs can detect real cheats

- Couldn't cheaters adapt their cheats?

- There are three types of cheats:
  1. Detection impossible (Example: Collusion)
  2. Detection not guaranteed, but evasion technically difficult
  3. Detection guaranteed (≥15% of the cheats in our sample)
Impact on frame rate

Frame rate is ~13% lower than on bare hw

- 137fps is still a lot! 60--80fps generally recommended
- 11% due to logging; additional cost for accountability is small
Cost of auditing

- When auditing a player after a one-hour game,
  - How big is the log we have to download? 148 MB
  - How much time is needed for replay? ~1 hour
Online auditing

- Idea: Stream logs to auditors during the game
- Result: Detection within seconds after fault occurs
- Replay can utilize unused cores; frame rate penalty is low
Summary

- Accountable Virtual Machines (AVMs) offer strong accountability for unmodified binaries
  - Useful when relying on software executing on remote machines: Federated system, multiplayer games, ...
  - No trusted components required

- AVMs are practical
  - Prototype implementation based on VMware Workstation
  - Evaluation: Cheat detection in Counterstrike

Questions?
Thank you!

Our enthusiastic Counterstrike volunteers