Detecting Targeted Attacks Using Shadow Honeypots

Presented By Archana Viswanath
Honeypots - Where did the name come from???

• Honeypot is often understood to refer to the English children's character Winnie-the-Pooh, a stuffed bear who was lured into various predicaments by his desire for pots of honey.
Honeypots – Network Parlance

- An Internet-attached server.
- Aim - acts as a decoy to lure potential hackers
- Actions - Studies their activities and monitors how they are able to break into a system.
- The intruder will have no idea that she/he is being tricked and monitored.
Honeypots – Purpose

- Learn the weaknesses in the system.
- To catch and stop the hacker.
- Create more secure systems that are potentially invulnerable to future hackers.
HoneyPots...

- They are a resource that has no authorized activity and no production value.
- This means that any interaction with a honeypot is most likely malicious or unauthorized.
- Any connections sent to the honeypot are most likely a probe, scan or attack.
Honeypots Effective Against:

- Effective against Scanning worms.
- Ineffective against – Topological, hit-list worms.....(Why???)
- Typically used only for server-type applications.
Anomaly Detection Systems

- Detects anything anomalous from normal behavior of the system.

- **Advantage** – Possibility of detecting and responding to previously unknown attacks.

- **Disadvantage** –
  (i). Tune the system to detect more potential attacks (Low FN, High FP)
  (ii). Tune the system to be more insensitive to attacks (High FN, Low FP)
Shadow Honeypots

- Combines the features of Honeypots and Anomaly Detection Systems.
- ADS monitors the incoming traffic.
- Anomalous traffic is further processed by the shadow honeypot.
- Shadow is an instance of the protected software.
Shadow Honeypot Architecture

![Diagram of Shadow Honeypot Architecture]

- Regular Service Code
- Protected System
- State Rollback
- Process State
- Anomaly Detection Sensors
- Filtering
- Traffic from the network
- Update Predictors
- Update filters
- OS Kernel
- User processes
- Protected Service Address Space
- Shadow Honeypot Code
**System Workflow**

1. **Input arrives**
   - **Known Bad Input?**
     - Yes: Drop request
     - No: **Suspect Input Based on AD?**
       - Yes: Use Shadow
         - Yes: Attack Detected?
           - Yes: Indicate False Positive to AD
             - Update AD Model
           - No: Indicate Hit to AD
             - Update AD Model
             - Update Filtering Component
         - No: Indicate False Negative to AD
           - Update AD Model
       - No: Randomly Use Shadow Anyway
         - Yes: Use Shadow
           - Yes: Attack Detected?
             - Yes: Indicate False Positive to AD
               - Update AD Model
             - No: Indicate Hit to AD
               - Update AD Model
               - Update Filtering Component
           - No: Indicate False Negative to AD
             - Update AD Model
         - No: Indicate False Positive to AD
           - Update AD Model
           - Update Filtering Component

2. **Handle request normally:** if attack, system gets compromised
Shadow Honeypot Prototype

- Network Processor – custom load balancer & filter
- Snort Sensors – connected to the network processor
- Shadow HoneyPot – connected to the sensors.
Shadow Honeypot Prototype

Diagram showing the components and connections of a shadow honeypot prototype. The diagram includes:

- Web server farm
- Shadow server
- Internal network
- Switch
- Router
- External network
- IXP1200-based firewall/load balancer
- Loosely-coupled shadow services
- PC-based modified snort sensors for anomaly detection
Implementation

Filtering and Anomaly Detection

- Two Anomaly Detection Systems – Payload Sifting & Abstract Payload Execution

- Payload Sifting – fingerprinting- High False Positives

- Abstract Payload Execution – Buffer Overflow detection – searches for sufficiently long sequences of valid instructions.
Implementation...

**Shadow Honeypot Creation**
- Use `pmalloc()` instead of `malloc()` for heap allocation.
- `pmalloc()` allocates two additional zero filled, write-protected pages that bracket the requested buffer.
- Buffer overflow – will cause the process to receive a Segmentation Violation signal.
- This is caught by the signal handler which notifies the OS to abort all changes.
- If no violation then changes are persisted.
Implementation...Memory Allocation

3 Memory Pages Allocated by pmalloc

Write Protected Memory Page

1024 bytes

Write Protected Memory Page

x000

ptr

x4096
Limitations

- Effectiveness of the rollback mechanisms for the transactions depends on their proper placement for committing state changes and latency of the detector.
- Loosely coupled shadows – weak against attacks that depend on the system state.
- Not explored – feedback from shadow honeypots to tune the anomaly detection components.
Take Away

- It's obvious that fine tuning will get you a step closer to efficient detection.

![Computers Make Very Fast, Very Accurate Mistakes](image)