Mimicry Attacks on Host-Based Intrusion Detection

David Wagner  Paolo Soto

University of California at Berkeley
The topic of this talk:

How do we evaluate the security of a host-based IDS against sophisticated attempts to evade detection?

One answer: “adversarial scholarship”
The Cryptographer’s Creed

- **Conservative design**
  - Systems should be evaluated by the worst failure that is at all plausible under assumptions favorable to the attacker*

- **Kerkhoff’s principle**
  - Systems should remain secure even when the attacker knows all internal details of the system

- **The study of attacks**
  - We should devote considerable effort to trying to break our own systems; this is how we gain confidence in their security

* Credits: Gwyn
Research Into Attacks

- We could benefit from a stronger tradition of research into attacks on intrusion detection.

Table 1. Papers published in the past five years, by subject.

<table>
<thead>
<tr>
<th>Subject</th>
<th>Design</th>
<th>Attacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block ciphers</td>
<td>81</td>
<td>100</td>
</tr>
<tr>
<td>Intrusion detection</td>
<td>120</td>
<td>7</td>
</tr>
</tbody>
</table>
In This Talk...

How do we evaluate the security of a host-based IDS against sophisticated attempts to evade detection?

Organization of this talk:
- Host-based intrusion detection
- Mimicry attacks, and how to find them
- Attacking pH, a host-based IDS
- Concluding thoughts
Host-based Intrusion Detection

Anomaly detection:
- IDS monitors system call trace from the app
- DB contains a list of subtraces that are allowed to appear
- Any observed subtrace not in DB sets off alarms
The Mimicry Attack

1. *Take control of the app.*
   - e.g., by a buffer overrun

2. *Execute payload while mimicking normal app behavior.*
   - If exploit sequence contains only allowed subtraces, the intrusion will remain undetected.
When Are Attacks Possible?

The central question for mimicry attacks:

- Can we craft an exploit sequence out of only allowed subtraces and still cause any harm?

Assumptions:

- IDS algorithm + DB is known to attacker [*Kerckhoff*]
- Can take control of app undetected [*Conservative design*]
Disguising the Payload

Attacker has many degrees of freedom:

- Wait until malicious payload would be allowed
- Vary the malicious payload by adding no-ops
  - e.g., `(void) getpid()` or `open(NULL, 0)`
  - In fact, nearly all syscalls can be turned into no-ops

- Note: the set of choices can be expressed as a regexp
  - Let \( \mathcal{N} \) denote the set of no-op-able syscalls
  - Then `open()` `write()` can be replaced by anything matching \( \mathcal{N}^* \) `open()` \( \mathcal{N}^* \) `write()` \( \mathcal{N}^* \)
To check whether there is a mimicry attack:

- Let $\Sigma =$ set of security-relevant events,
  $\mathcal{M} =$ set of “bad” traces that do damage to the system,
  $\mathcal{A} =$ set of traces allowed by the IDS ($\mathcal{M}, \mathcal{A} \subseteq \Sigma^*$)

- If $\mathcal{M} \cap \mathcal{A} \neq \emptyset$, then there is a mimicry attack
To check whether there is a mimicry attack:
- Let $\Sigma =$ set of security-relevant events,
  $M =$ set of “bad” traces that do damage to the system,
  $A =$ set of traces allowed by the IDS  
  $(M, A \subseteq \Sigma^*)$
- If $M \cap A \neq \emptyset$, then there is a mimicry attack

Then just apply automata theory
- $M$: regular expression (regular language)
- $A$: finite-state system (regular language)
  - Works since IDS’s are typically just finite-state machines
Experience: Mimicry in Action

The experiment:

- **pH**: a host-based IDS [SF00]
- **autowux**: a wuftpdp exploit
- No mimicry attacks with the original payload

... but, after a slight modification ...
We found a modified payload that raises no alarms and has a similar effect on the system.

- pH may be at risk for mimicry attacks
Conclusions

- Mimicry attacks: A threat to host-based IDS?
  - Practical implications not known

- The study of attacks is important
  - Unfortunately, there’s so much we don’t know...