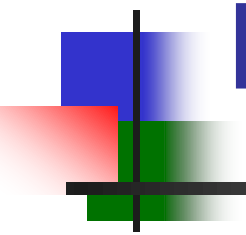
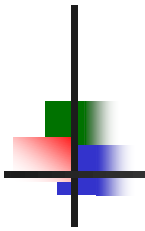


Mimicry Attacks on Host-Based Intrusion Detection

David Wagner Paolo Soto
University of California at Berkeley



Preview

- 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

	Design	Attacks
Block ciphers	81	100
Intrusion detection	120	7

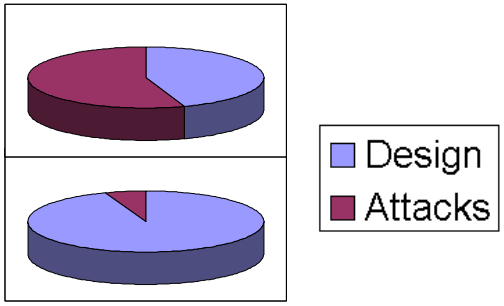
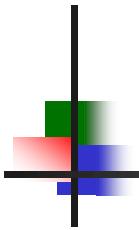


Table 1. Papers published in the past five years, by subject.

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



In This Talk…

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

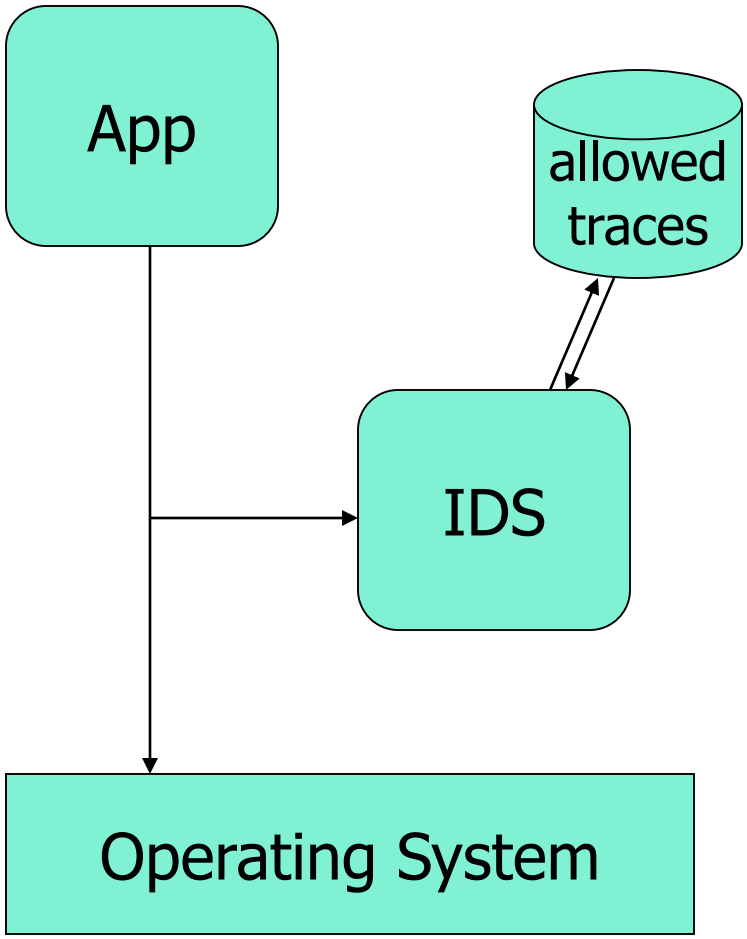
Organization of this talk:

- Host-based intrusion detection
- Mimicry attacks, and how to find them
- Attacking pH, a host-based IDS
- Concluding thoughts

Host-based Intrusion Detection

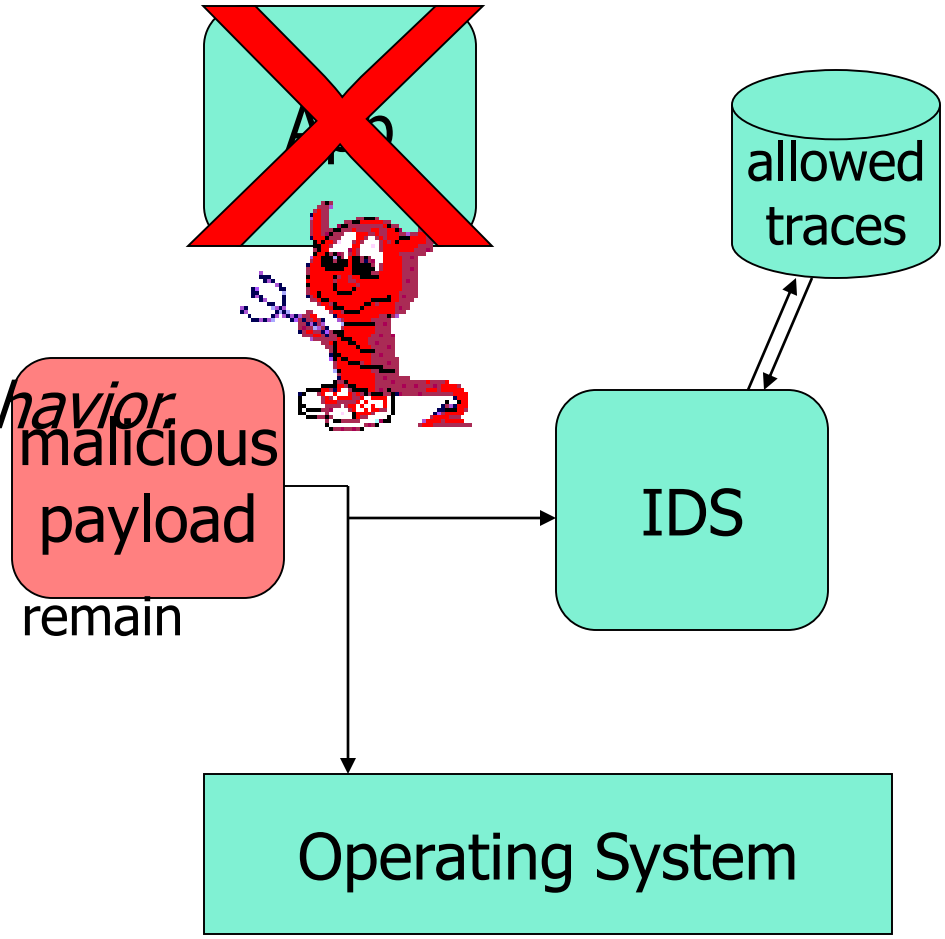
Anomaly detection:

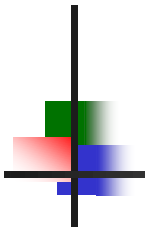
- IDS monitors system call trace from the app
- DB contains a list of subtraces that are allowed to appear
- Any observed subtrace not in DB sets off alarms



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.





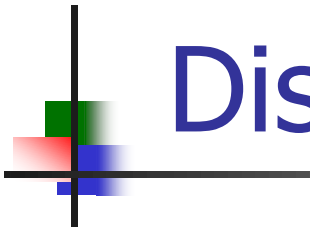
When Are Attacks Possible?

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 [*Kerkhoff*]
  - Can take control of app undetected [*Conservative design*]

Disguising the Payload



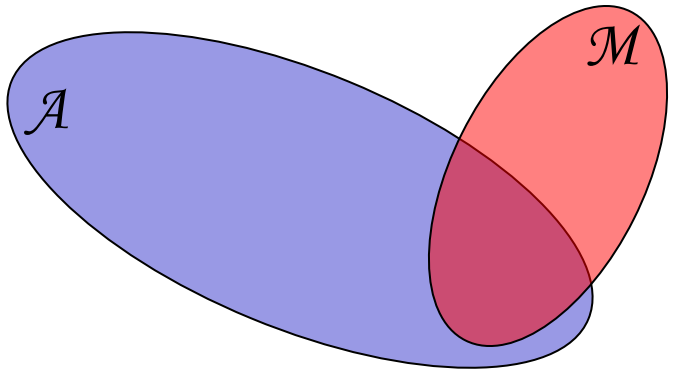
Attacker has many degrees of freedom:

- 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()` $\mathcal{N}^*$

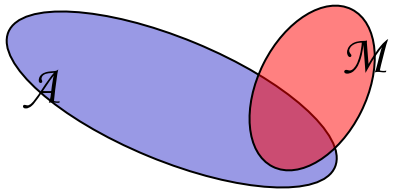
A Theoretical Framework

- To check whether there is a mimicry attack:
  - Let Σ = 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



A Theoretical Framework

- To check whether there is a mimicry attack:
  - Let Σ = 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



- Then just apply automata theory
  - $\mathcal{M}$: regular expression (regular language)
  - $\mathcal{A}$: finite-state system (regular language)
    - Works since IDS's are typically just finite-state machines

Experience: Mimicry in Action

```
setreuid(0,0), dup2(1,2), mkdir("sh"), chroot("sh"),
  9 × chdir(".."), chroot("/"), execve("/bin/sh").
```

```
setreuid(0,0), chroot("pub"),
 chdir("../../../../../../../../../../"), chroot("/"),
 open("/etc/passwd", O_APPEND|O_WRONLY),
 write(fd, "toor:AAaaaaaaaaaaa:0:0::/:/bin/sh", 33),
 close(fd), exit(0)
```

The experiment:

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

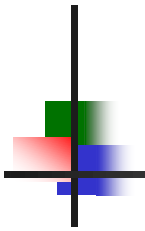
… but, after a slight modification …

A Successful Mimicry Attack

read() write() close() munmap() sigprocmask() wait4()
sigprocmask() sigaction() alarm() time() stat() read()
alarm() sigprocmask() *setreuid()* fstat() getpid()
time() write() time() getpid() sigaction() socketcall()
sigaction() close() flock() getpid() lseek() read()
kill() lseek() flock() sigaction() alarm() time()
stat() write() open() fstat() mmap() read() open()
fstat() mmap() read() close() munmap() brk() fcntl()
setregid() open() fcntl() *chroot()* *chdir()* setreuid()
lstat() lstat() lstat() lstat() open() fcntl() fstat()
lseek() getdents() fcntl() fstat() lseek() getdents()
close() write() time() open() fstat() mmap() read()
close() munmap() brk() fcntl() setregid() open() fcntl()
chroot() chdir() setreuid() lstat() lstat() lstat()
lstat() *open()* fcntl() brk() fstat() lseek() getdents()
lseek() getdents() time() stat() *write()* time() open()
getpid() sigaction() socketcall() sigaction() umask()
sigaction() alarm() time() stat() read() alarm()
getrlimit() pipe() fork() fcntl() fstat() mmap() lseek()
close() brk() time() getpid() sigaction() socketcall()
sigaction() chdir() sigaction() sigaction() write()
munmap() munmap() munmap() *exit()*

- 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…