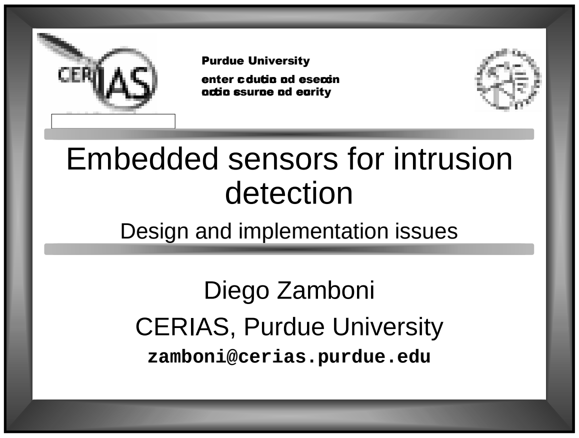
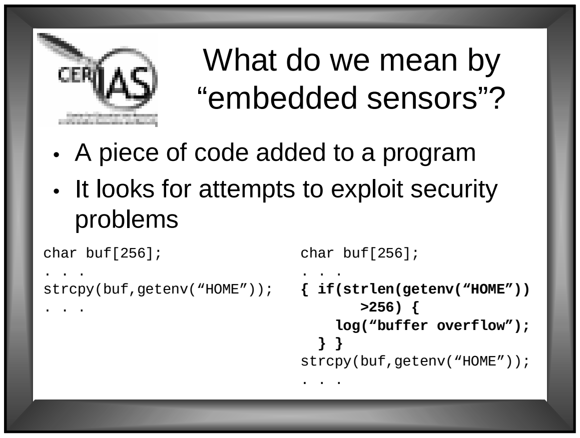


Purdue University
enter cdutio od esecin
octio esurce od earity

Embedded sensors for intrusion detection

Design and implementation issues

Diego Zamboni
CERIAS, Purdue University
zamboni@cerias.purdue.edu

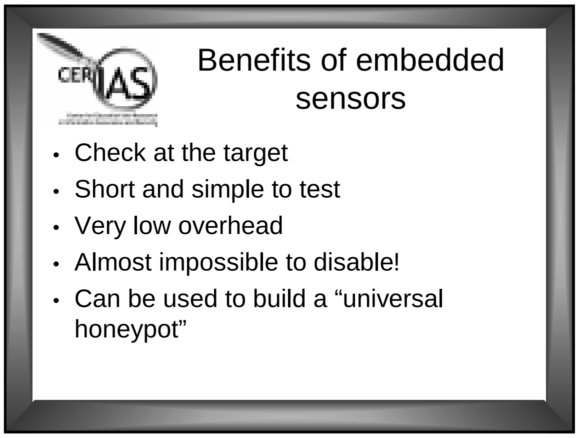


What do we mean by "embedded sensors"?

- A piece of code added to a program
- It looks for attempts to exploit security problems

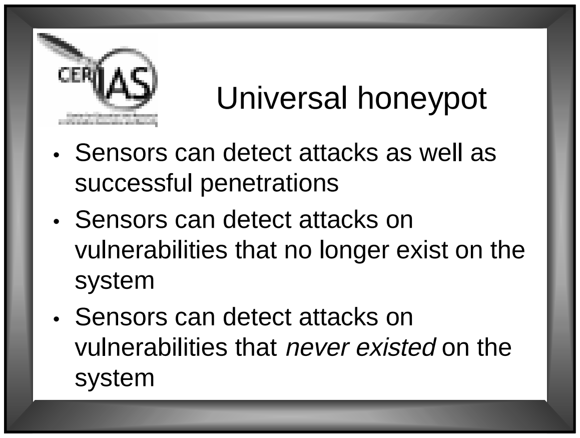
```
char buf[256];
. . .
strcpy(buf,getenv("HOME"));
. . .
```

```
char buf[256];
. . .
{ if(strlen(getenv("HOME"))
         >256) {
    log("buffer overflow");
  } }
strcpy(buf,getenv("HOME"));
. . .
```



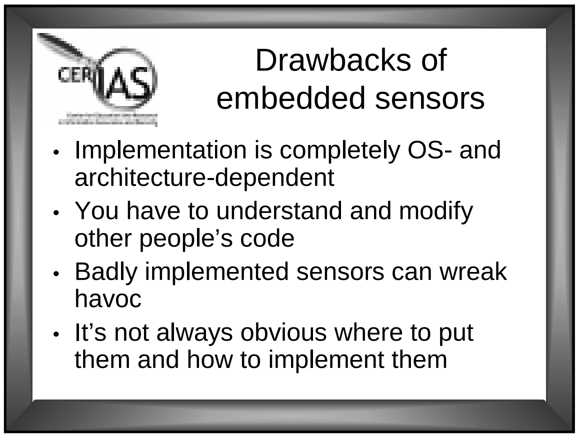
Benefits of embedded sensors

- Check at the target
- Short and simple to test
- Very low overhead
- Almost impossible to disable!
- Can be used to build a "universal honeypot"



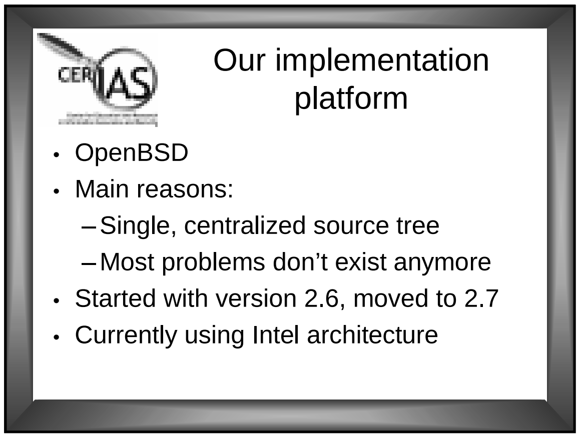
Universal honeypot

- Sensors can detect attacks as well as successful penetrations
- Sensors can detect attacks on vulnerabilities that no longer exist on the system
- Sensors can detect attacks on vulnerabilities that *never existed* on the system



Drawbacks of embedded sensors

- Implementation is completely OS- and architecture-dependent
- You have to understand and modify other people's code
- Badly implemented sensors can wreak havoc
- It's not always obvious where to put them and how to implement them



Our implementation platform

- OpenBSD
- Main reasons:
  – Single, centralized source tree
  – Most problems don't exist anymore
- Started with version 2.6, moved to 2.7
- Currently using Intel architecture

Our methodology for implementing sensors

- Use the CVE (version 20000712)
- For now, working on specific areas or programs (e.g. sendmail)
- For each entry, collect information, implement and test sensor
- Lather, rinse, repeat

Supporting infrastructure for sensors

- Reporting mechanism
  - A system call
  - A device file (for reading messages)
  - A corresponding library for both writing and reading messages

Sensors we have implemented

- Network attacks
  - Land, Teardrop, SYN flood, ping-of-death, Smurf, Fraggle, echo-chargen, WinNuke and others
- Sendmail attacks
  - MIME buffer overflows, debug/decode attacks, other root exploits

Are they any good?

- No false negatives (100% detection rate)
- Very few false positives (only with half of one sensor - Fraggle as reflector)
- No noticeable impact on the host

Some sensor statistics

- Added or changed 2034 lines total
- Of these, 193 are sensor code
- Average of 7.72 LOC per sensor
- Most sensors (60%) are 1-5 lines

Observations about CVE

- Useful as a checklist and for the references
- Bad entries
  - Entries that correspond to more than one specific attack
  - Badly-defined entries

Observations about sensor design

- Some are already built into the system, we just add the notification
- Some attacks are difficult to detect by program behavior alone
- What we have done is use heuristics (e.g. look at data)
- Other ideas?

Our current state

- Using OpenBSD 2.7
- Have implemented 25 sensors

The future

- Plan to have 100+ sensors by the end of the year
- Then – testing against new attacks
- After that – analysis, see what we can learn from the behavior of the sensors
- Finally – graduate?

Thank you

Diego Zamboni
CERIAS, Purdue University
`zamboni@cerias.purdue.edu`

Sample sensor – vacation program

```
close(pvect[0]);
close(pvect[1]);
close(fileno(mfp));
#ifdef ESP_CVE_1999_0057
if (from[0] == '-' && from[1] == 'C') {
  esp_logf("CVE-1999-0057: from='%s'\n", from);
}
#endif
execl(_PATH_SENDMAIL, "sendmail", "-f", myname,
      "--", from, NULL);
```