Development of Secure Software

Conclusions

We expect too much of developers!

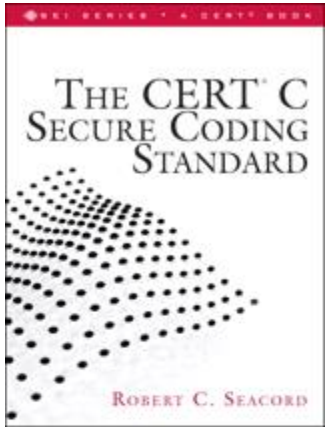
- Understanding whether a piece of C code is secure requires:
    - Understanding of the C language
        - Approx complexity: 700 pages of spec
    - Understanding the details of the compiler
        - Approx complexity: 3.7 million lines of code
    - Understanding the runtime library implementations
        - Approx complexity: 1.7 million lines of code
    - Understanding the operating system
        - Thousands of pages of specs and millions of lines of code
    - Understanding the details of the processor and other hardware

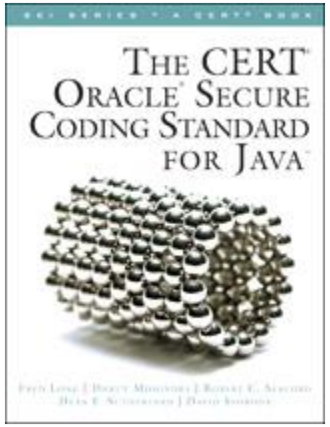
And the web is even worse!

- HTTP is an extensible standard with separate standards for each header
- The HTML 5 spec is several hundreds of pages
- The ECMAScript spec is several hundreds of pages
- A browser is as complex as an operating system
- And attacks against the web **include** the low-level attacks

How do we deal with this today?

- Coding guidelines and tooling
  - For instance: 89 Rules and 132 Recommendations in the CERT C Secure Coding Standard
  - Source code analysis tools implement heuristic checks to detect deviations from these rules
- Ad-hoc countermeasures in compiler / OS / middleware / frameworks
  - Stack canaries / ASLR / taint-mode / …
  - Anti-CSRF tokens / taint-tracking / …
- This can lead to substantial software security improvement
  - But is not the long-term solution

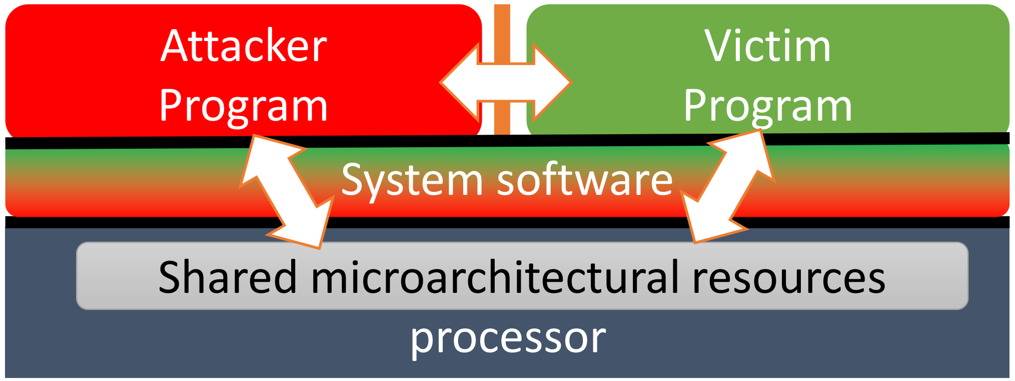




The way forward

- More principled approaches to software security
  - Programming language support: can we express security objectives within the code?
  - Compiler support: can the compiler provide complete protection against certain classes of attacks?
  - OS/hardware support: can we reduce the Trusted Computing Base? Can we make sure lower layers do not introduce new security issues?
  - …
- These are central questions in the software/system security research happening at DistriNet
  - Come talk to us about master theses

Research questions in software/system security

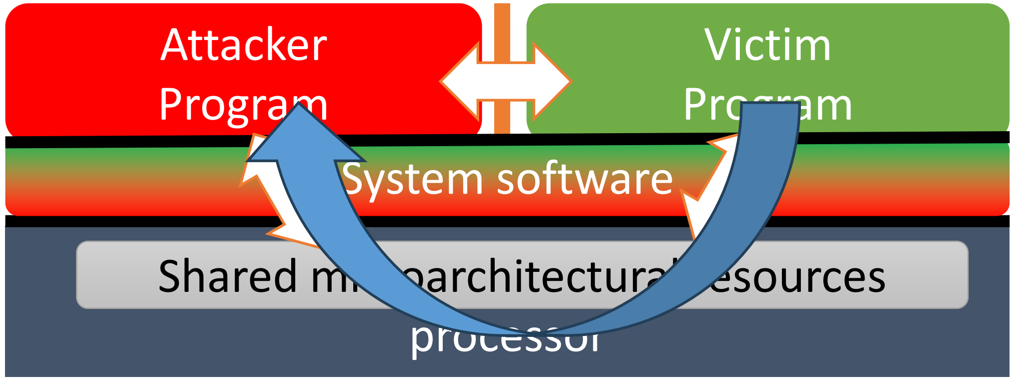


Syscalls

ISA specification

Research questions in software/system security

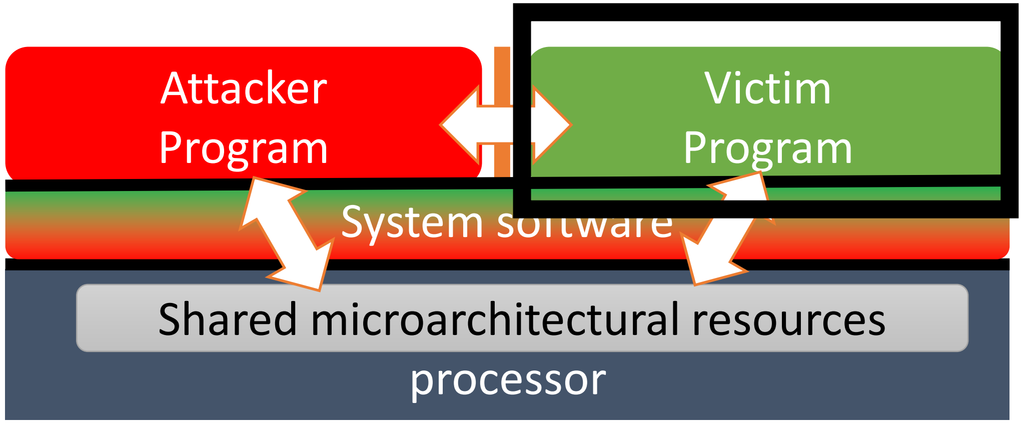
Cross-layer security: e.g., transient execution attacks



Syscalls

ISA specification

Research questions in software/system security

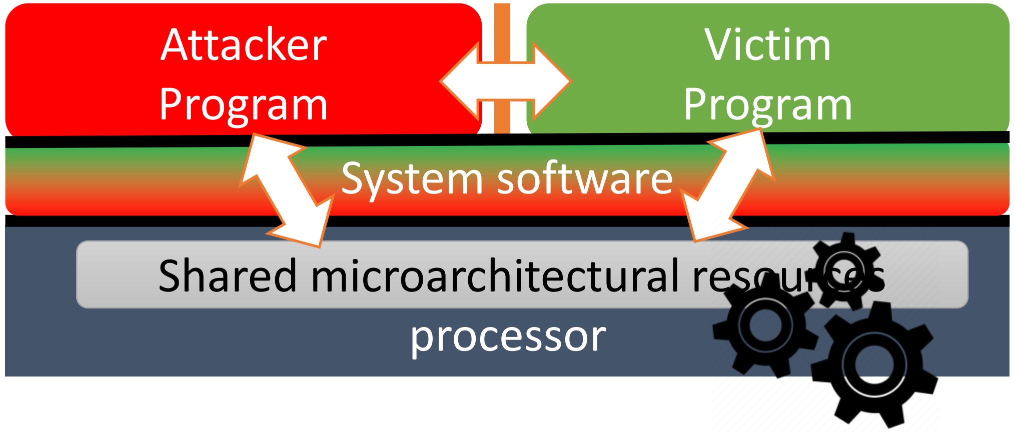


System level defense for mitigating remaining vulnerabilities within the isolated protection domain e.g., ASLR, Stack canaries

Syscalls

ISA specification

Research questions in software/system security



Syscalls

ISA specification

Hardware support: processor extensions, e.g. track secrets in the microarchitecture

Examination

- Closed-book written examination
- Typical structure of the exam:
    - Three questions, each on 5 points [the project is also on 5 points]
    - Typical questions
        - Define a number of terms
            - CSRF, non-interference, attacker model, …
        - Broad theory questions
            - Give an overview of attacks and countermeasures for low-level software vulnerabilities
        - Exercises
            - Specify a security automaton for friends-based access control
- Project can be redone in the Summer [but NOT recommended!]

Feedback welcome!

- Topic selection
  - Things I missed
  - Things that could be removed
- Study material
  - Suggestions for textbooks, background reading
- Project
  - What did you like, what could be improved?
- …

Q&A

Good luck with the exams!