CSci 8271
Security and Privacy in Computing
Day 12: Authenticated call stack

Stephen McCamant
University of Minnesota

Existing shadow stacks

- Provides strong (dynamic) CFI for return addresses
- Software implementations have performance/security trade-offs
- Can be cheap with specialized hardware
  - Research designs, Intel CET
- Today: based on a more general-purpose protection

ARM pointer authentication

- Support for computing and checking MACs on code and data pointers
- Tag stored in top bits (16 or so out of 64)
- Keys can be restricted to kernel management
- Verification failure just makes an invalid pointer

Chaining and masking

- Previous approaches tie the SP with the return pointer
  - Good, but some replays are still possible
- Instead, chain by including the caller's value in the MAC
  - C.f. Merkle tree, blockchain
- To hide hash collisions, XOR with another MAC value

Security analysis

- Memory-based attacker needs two steps to influence control flow
- Without masking, finding a collision is practical
  - Easiest is getting elsewhere in the legal call graph
- With masking, attacker must just be lucky in getting a collision

Performance and applicability

- Real benchmarks aren't possible yet
  - Simulated PA overhead
- Slower than less-secure shadow stacks
  - 3% overhead on SPEC and up to 13% on NGINX
- Multi-threading is automatic, `longjmp` not
- Degraded but some benefit if only part of a program is protected