CSci 5271
Introduction to Computer Security
Day 23: Firewalls, NATs, and IDSes

Stephen McCamant
University of Minnesota, Computer Science & Engineering

Outline

Design robustness principles

- Use timestamps or nonces for freshness
- Be explicit about the context
- Don't trust the secrecy of others' secrets
- Whenever you sign or decrypt, beware of being an oracle
- Distinguish runs of a protocol

Implementation principles

- Ensure unique message types and parsing
- Design for ciphers and key sizes to change
- Limit information in outbound error messages
- Be careful with out-of-order messages

Outline

Random numbers and entropy

- Cryptographic RNGs use cipher-like techniques to provide indistinguishability
- But rely on truly random seeding to stop brute force
  - Extreme case: no entropy → always same "randomness"
- Modern best practice: seed pool with 256 bits of entropy
  - Suitable for security levels up to 2^{256}

Netscape RNG failure

- Early versions of Netscape SSL (1994-1995) seeded with:
  - Time of day
  - Process ID
  - Parent process ID
- Best case entropy only 64 bits
  - (Not out of step with using 40-bit encryption)
- But worse because many bits guessable

Debian/OpenSSL RNG failure (1)

- OpenSSL has pretty good scheme using `/dev/urandom`
- Also mixed in some uninitialized variable values
  - "Extra variation can't hurt"
- From modern perspective, this was the original sin
  - Remember undefined behavior discussion?
- But had no immediate ill effects

Debian/OpenSSL RNG failure (2)

- Debian maintainer commented out some lines to fix a Valgrind warning
  - "Potential use of uninitialized value"
- Accidentally disabled most entropy (all but 16 bits)
- Brief mailing list discussion didn't lead to understanding
- Broken library used for ~2 years before discovery

Detected RSA/DSA collisions

- 2012: around 1% of the SSL keys on the public net are breakable
  - Some sites share complete keypairs
  - RSA keys with one prime in common (detected by large-scale GCD)
- One likely culprit: insufficient entropy in key generation
  - Embedded devices, Linux `/dev/urandom` vs. `/dev/random`
- DSA signature algorithm also very vulnerable

Newer factoring problem (CCS'17)

- An Infineon RSA library used primes of the form $p = k \cdot M + (65537^a \bmod M)$
- Smaller problems: fingerprintable, less entropy
- Major problem: can factor with a variant of Coppersmith's algorithm
  - E.g., 3 CPU months for a 1024-bit key

Side-channel attacks

- Timing analysis:
  - Number of 1 bits in modular exponentiation
  - Unpadding, MAC checking, error handling
  - Probe cache state of AES table entries
- Power analysis
  - Especially useful against smartcards
- Fault injection

WEP "privacy"

- First WiFi encryption standard: Wired Equivalent Privacy (WEP)
- F&S: designed by a committee that contained no cryptographers
- Problem 1: note "privacy": what about integrity?
  - Nope: stream cipher + CRC = easy bit flipping

WEP shared key

- Single key known by all parties on network
- Easy to compromise
- Hard to change
- Also often disabled by default
- Example: a previous employer

WEP key size and IV size

- Original sizes: 40-bit shared key (export restrictions) plus 24-bit IV = 64-bit RC4 key
  - Both too small
- 128-bit upgrade kept 24-bit IV
  - Vague about how to choose IVs
  - Least bad: sequential, collision takes hours
  - Worse: random or everyone starts at zero

WEP RC4 related key attacks

- Only true crypto weakness
- RC4 "key schedule" vulnerable when:
  - RC4 keys very similar (e.g., same key, similar IV)
  - First stream bytes used
- Not a practical problem for other RC4 users like SSL
  - Key from a hash, skip first output bytes

Newer problem with WPA (CCS'17)

- Session key set up in a 4-message handshake
- Key reinstallation attack: replay #3
  - Causes most implementations to reset nonce and replay counter
  - In turn allowing many other attacks
  - One especially bad case: reset key to 0
- Protocol state machine behavior poorly described in spec
  - Outside the scope of previous security proofs

Trustworthiness of primitives

- Classic worry: DES S-boxes
- Obviously in trouble if cipher chosen by your adversary
- In a public spec, most worrying are unexplained elements
- Best practice: choose constants from well-known math, like digits of π

Dual_EC_DRBG (1)

- Pseudorandom generator in NIST standard, based on elliptic curve
- Looks like provable (slow enough!) but strangely no proof
- Specification includes long unexplained constants
- Academic researchers find:
  - Some EC parts look good
  - But outputs are statistically distinguishable

Dual_EC_DRBG (2)

- Found 2007: special choice of constants allows prediction attacks
  - Big red flag for paranoid academics
- Significant adoption in products sold to US govt. FIPS-140 standards
  - Semi-plausible rationale from RSA (EMC)
- NSA scenario basically confirmed by Snowden leaks
  - NIST and RSA immediately recommend withdrawal

Outline

More crypto protocols, cont'd

Examples of crypto failure

Announcements intermission

Firewalls and NAT boxes

Intrusion detection systems

Malware and the network

Note to early readers

- This is the section of the slides most likely to change in the final version
- If class has already happened, make sure you have the latest slides for announcements

Outline

More crypto protocols, cont'd

Examples of crypto failure

Announcements intermission

Firewalls and NAT boxes

Intrusion detection systems

Malware and the network

Internet addition: middleboxes

- Original design: middle of net is only routers
  - End-to-end principle
- Modern reality: more functionality in the network
- Security is one major driver

Security/connectivity tradeoff

- A lot of security risk comes from a network connection
  - Attacker could be anywhere in the world
- Reducing connectivity makes security easier
- Connectivity demand comes from end users

What a firewall is

- Basically, a router that chooses not to forward some traffic
  - Based on an a-priori policy
- More complex architectures have multiple layers
  - *DMZ*: area between outer and inner layers, for outward-facing services

Inbound and outbound control

- Most obvious firewall use: prevent attacks from the outside
- Often also some control of insiders
  - Block malware-infected hosts
  - Employees wasting time on Facebook
  - Selling sensitive info to competitors
  - Nation-state Internet management
- May want to log or rate-limit, not block

Default: deny

- Usual allow-list approach: first, block everything
- Then allow certain traffic
- Basic: filter packets based on headers
- More sophisticated: *proxy* traffic at a higher level

IPv4 address scarcity

- Design limit of 2^{32} hosts
  - Actually less for many reasons
- Addresses becoming gradually more scarce over a many-year scale
- Some high-profile exhaustions in 2011
- IPv6 adoption still quite low, occasional signs of progress

Network address translation (NAT)

- Middlebox that rewrites addresses in packets
- Main use: allow inside network to use non-unique IP addresses
  - RFC 1918: 10.*, 192.168.*, etc.
  - While sharing one outside IP address
- Inside hosts not addressable from outside
  - De-facto firewall

Packet filtering rules

- Match based on:
  - Source IP address
  - Source port
  - Destination IP address
  - Destination port
  - Packet flags: TCP vs. UDP, TCP ACK, etc.
- Action, e.g. allow or block
- Obviously limited in specificity

Client and server ports

- TCP servers listen on well-known port numbers
  - Often $<$ 1024, e.g. 22 for SSH or 80 for HTTP
- Clients use a kernel-assigned random high port
- Plain packet filter would need to allow all high-port incoming traffic

Stateful filtering

- In general: firewall rules depend on previously-seen traffic
- Key instance: allow replies to an outbound connection
- See: port 23746 to port 80
- Allow incoming port 23746
  - To same inside host
- Needed to make a NAT practical

Circuit-level proxying

- Firewall forwards TCP connections for inside client
- Standard protocol: SOCKS
  - Supported by most web browsers
  - Wrapper approaches for non-aware apps
- Not much more powerful than packet-level filtering

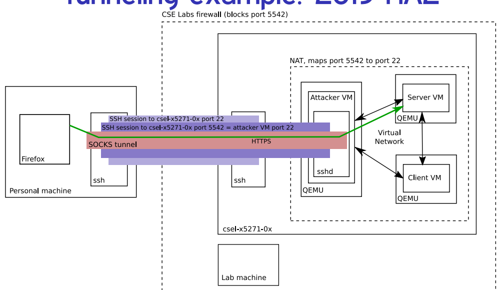
Application-level proxying

- Knows about higher-level semantics
- Long history for, e.g., email, now HTTP most important
- More knowledge allows better filtering decisions
  - But, more effort to set up
- Newer: "transparent proxy"
  - Pretty much a middleperson

Tunneling

- Any data can be transmitted on any channel, if both sides agree
- E.g., encapsulate IP packets over SSH connection
  - Compare covert channels, steganography
- Powerful way to subvert firewall
  - Some legitimate uses

Tunneling example: 2019 HA2



Outline

- More crypto protocols, cont'd
- Examples of crypto failure
- Announcements intermission
- Firewalls and NAT boxes
- Intrusion detection systems
- Malware and the network

Basic idea: detect attacks

- The worst attacks are the ones you don't even know about
- Best case: stop before damage occurs
  - Marketed as "prevention"
- Still good: prompt response
- Challenge: what is an attack?

Network and host-based IDSes

- Network IDS: watch packets similar to firewall
  - But don't know what's bad until you see it
  - More often implemented offline
- Host-based IDS: look for compromised process or user from within machine

Signature matching

- *Signature* is a pattern that matches known bad behavior
- Typically human-curated to ensure specificity
- See also: anti-virus scanners

Anomaly detection

- Learn pattern of normal behavior
- "Not normal" is a sign of a potential attack
- Has possibility of finding novel attacks
- Performance depends on normal behavior too

Recall: FPs and FNs

- False positive: detector goes off without real attack
- False negative: attack happens without detection
- Any detector design is a tradeoff between these (ROC curve)

Signature and anomaly weaknesses

- Signatures
  - Won't exist for novel attacks
  - Often easy to attack around
- Anomaly detection
  - Hard to avoid false positives
  - Adversary can train over time

Base rate problems

- If the true incidence is small (low base rate), most positives will be false
  - Example: screening test for rare disease
- Easy for false positives to overwhelm admins
- E.g., 100 attacks out of 10 million packets, 0.01% FP rate
  - How many false alarms?

Adversarial challenges

- FP/FN statistics based on a fixed set of attacks
- But attackers won't keep using techniques that are detected
- Instead, will look for:
  - Existing attacks that are not detected
  - Minimal changes to attacks
  - Truly novel attacks

Wagner and Soto mimicry attack

- Host-based IDS based on sequence of syscalls
- Compute $A \cap M$, where:
  - A models allowed sequences
  - M models sequences achieving attacker's goals
- Further techniques required:
  - Many syscalls made into NOPs
  - Replacement subsequences with similar effect

Outline

Malicious software

- Shortened to Mal...ware
- Software whose inherent goal is malicious
  - Not just used for bad purposes
- Strong adversary
- High visibility
- Many types

Trojan (horse)

- Looks benign, has secret malicious functionality
- Key technique: fool users into installing/running
- Concern dates back to 1970s, MLS

(Computer) viruses

- Attaches itself to other software
- Propagates when that program runs
- Once upon a time: floppy disks
- More modern: macro viruses
- Have declined in relative importance

Worms

- Completely automatic self-propagation
- Requires remote security holes
- Classic example: 1988 Morris worm
- "Golden age" in early 2000s
- Internet-level threat *seems* to have declined

Fast worm propagation

- Initial hit-list
  - Pre-scan list of likely targets
  - Accelerate cold-start phase
- Permutation-based sampling
  - Systematic but not obviously patterned
  - Pseudorandom permutation
- Approximate time: 15 minutes
  - "Warhol worm"
  - Too fast for human-in-the-loop response

Getting underneath

- Lower-level/higher-privilege code can deceive normal code
- Rootkit: hide malware by changing kernel behavior
- MBR virus: take control early in boot
- Blue-pill attack: malware is a VMM running your system

Malware motivation

- Once upon a time: curiosity, fame
- Now predominates: money
  - Modest-size industry
  - Competition and specialization
- Also significant: nation-states
  - Industrial espionage
  - Stuxnet (not officially acknowledged)

User-based monetization

- Adware, mild spyware
- Keyloggers, stealing financial credentials
- Ransomware
  - Application of public-key encryption
  - Malware encrypts user files
  - Only \$300 for decryption key

Bots and botnets

- Bot: program under control of remote attacker
- Botnet: large group of bot-infected computers with common "master"
- Command & control network protocol
  - Once upon a time: IRC
  - Now more likely custom and obfuscated
  - Centralized → peer-to-peer
  - Gradually learning crypto and protocol lessons

Bot monetization

- Click (ad) fraud
- Distributed DoS (next section)
- Bitcoin mining
- Pay-per-install (subcontracting)
- Spam sending

Malware/anti-virus arms race

- "Anti-virus" (AV) systems are really general anti-malware
- Clear need, but hard to do well
- No clear distinction between benign and malicious
- Endless possibilities for deception

Signature-based AV

- Similar idea to signature-based IDS
- Would work well if malware were static
- In reality:
  - Large, changing database
  - Frequent updated from analysts
  - Not just software, a subscription
  - Malware stays enough ahead to survive

Emulation and AV

- Simple idea: run sample, see if it does something evil
- Obvious limitation: how long do you wait?
- Simple version can be applied online
- More sophisticated emulators/VMs used in backend analysis

Polymorphism

- Attacker makes many variants of starting malware
- Different code sequences, same behavior
- One estimate: 30 million samples observed in 2012
- But could create more if needed

Packing

- Sounds like compression, but real goal is obfuscation
- Static code creates real code on the fly
- Or, obfuscated bytecode interpreter
- Outsourced to independent "protection" tools

Fake anti-virus

- Major monentization strategy recently
- Your system is infected, pay $19.95 for cleanup tool
- For user, not fundamentally distinguishable from real AV

Next time

- Network anonymity with overlay networks