Email encryption
- Technology became available with PGP in the early 90s
- Classic depressing study: "Why Johnny can't encrypt: a usability evaluation of PGP 5.0" (USENIX Security 1999)
- Still an open "challenge problem"
- Also some other non-UI difficulties: adoption, govt. policy

Phishing
- Attacker sends email appearing to come from an institution you trust
- Links to web site where you type your password, etc.
- Spear phishing: individually targeted, can be much more effective

Phishing defenses
- Educate users to pay attention to X:
  - Spelling → copy from real emails
  - URL → homograph attacks
  - SSL "lock" icon → fake lock icon, or SSL-hosted attack
- Extended validation (green bar) certificates
- Phishing URL deny-lists

SSL warnings: prevalence
- Browsers will warn on SSL certificate problems
- In the wild, most are false positives
  - foo.com vs. www.foo.com
  - Recently expired
  - Technical problems with validation
  - Self-signed certificates (HA2)
- Classic warning-fatigue danger

SSL warnings: effectiveness
- Early warnings fared very poorly in lab settings
- Recent browsers have a new generation of designs:
  - Harder to click through mindlessly
  - Persistent storage of exceptions
- Recent telemetry study: they work pretty well
Spam-advertised purchases

- "Replica" Rolex watches, herbal Viagra, etc.
- This business is clearly unscrupulous; if I pay, will I get anything at all?
- Empirical answer: yes, almost always
- Not a scam, a black market
- Importance of credit-card bank relationships

Advance fee fraud

- "Why do Nigerian Scammers say they are from Nigeria?" (Herley, WEIS 2012)
- Short answer: false positives
  - Sending spam is cheap
  - But, luring victims is expensive
  - Scammer wants to minimize victims who respond but ultimately don't pay

Trusted UI

- Tricky to ask users to make trust decisions based on UI appearance
  - Lock icon in browser, etc.
- Attacking code can draw lookalike indicators
  - Lock favicon
  - Picture-in-picture attack

Smartphone app permissions

- Smartphone OSes have more fine-grained per-application permissions
  - Access to GPS, microphone
  - Access to address book
  - Make calls
- Phone also has more tempting targets
- Users install more apps from small providers

Permissions manifest

- Android approach: present listed of requested permissions at install time
- Can be hard question to answer hypothetically
  - Users may have hard time understanding implications
- User choices seem to put low value on privacy
**Time-of-use checks**
- iOS approach: for narrower set of permissions, ask on each use
- Proper context makes decisions clearer
- But, have to avoid asking about common things
- iOS app store is also more closely curated

**Trusted UI for privileged actions**
- Trusted UI works better when asking permission (e.g., Oakland’12)
- Say, “take picture” button in phone app
  - Requested by app
  - Drawn and interpreted by OS
  - OS well positioned to be sure click is real
- Little value to attacker in drawing fake button

**Outline**
- Usable security example areas
- More causes of crypto failure
- Time reserved for SRTs

**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
### 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**
- **Data non-erasure**
  - Hard disks, "cold boot" on RAM

### 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 such a problem for other RC4 users like SSL
  - Key from a hash, skip first output bytes

### New 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 $\pi$

### 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

- Usable security example areas
- More causes of crypto failure
- Time reserved for SRTs