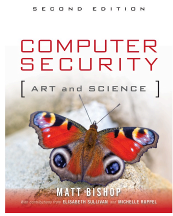
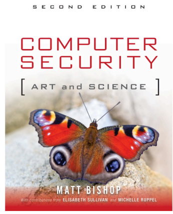


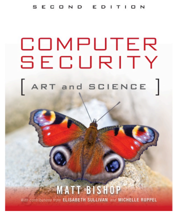
Attacks and Responses

Chapter 27



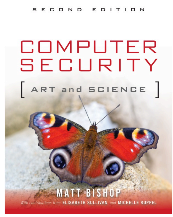
Outline

- Representing attacks
  - Attack trees
  - Attack graphs
- Intrusion response
  - Incident prevention
  - Incident handling
- Digital forensics
  - Principles and practices
  - Anti-forensics



Attacks

- *Attack*: a sequence of actions creating a violation of a security policy
  - *Multistage attack*: attack requiring several steps to achieve its goal
- *Goal of the attack*: what the attacker hopes to achieve
- *Target of the attack*: entity that the attacker wishes to affect
- Example: burglar stealing someone's jewelry
  - *Attack*: what she does to steal the jewelry; probably *multistage* (break window, find jewelry box, break it open, take jewelry, get out of house)
  - *Goal of the attack*: steal the jewelry
  - *Target of the attack*: the jewelry, also the owner of the jewelry

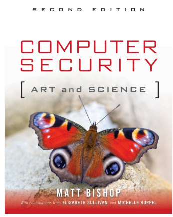


Representing Attacks

- Can be done at many levels of abstraction
- As you go deeper, some steps become more detailed and break down into multiple steps themselves
- *Subgoal*: the goal of each step to move the attacker closer to the goal of the attack

Example: Penetration of Corporate Computer System

- Goal: gain access to corporate computer system
- Procedure was to try to get people to reveal account information, change passwords to something the attackers knew
  - Target: newly-hired employees who hadn’t had computer security awareness briefing
  - Subgoal 1: find those people
  - Subgoal 2: get them to reveal account info, change passwords

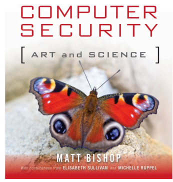


Focus on Subgoal 1

- For subgoal 1, needed to find list of these people
  - Subgoal 1-1: learn about company's organization
- Procedure was to get annual report (public), telephone directory (not public)
  - Subgoal 1-2: acquire the telephone directory (this required 2 numbers)
  - Subgoal 1-3: get the two numbers (only available to employees)
  - Subgoal 1-4: impersonate employees
- Had corporate controls blocked attackers from achieving subgoal, they would need to find other ways of doing it

Attack Trees

- Represent the goals and subgoals as a sequence of hierarchical nodes in a tree
  - Goal is the root

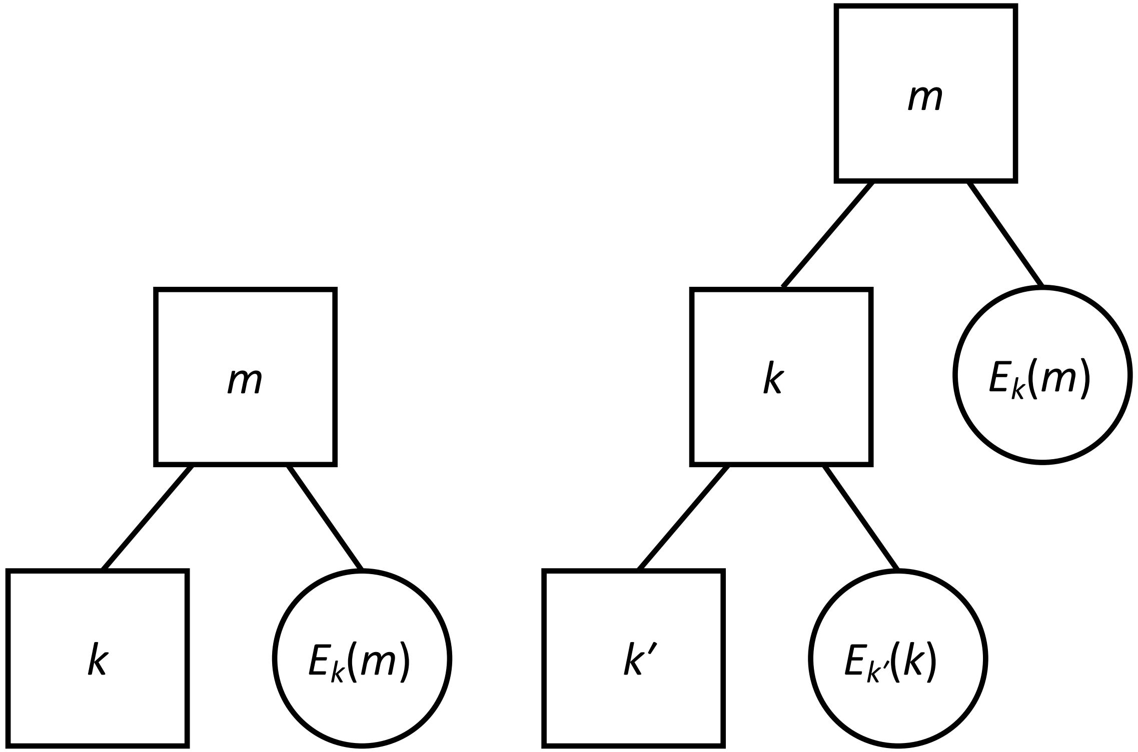


Security Flaws in Cryptographic Key Management Schemes

- Goal: develop package to allow attackers to ask what data is needed to determine encryption key
- System has only 2 functions, $E_k(m)$ and $D_k(c)$
- Attack (“search”) tree has the required information represented as root node, other nodes represent subgoals
- 2 types of nodes
  - Required: represents information necessary for parent; *satisfied* when that information becomes available
  - Available: represents known information
- As tree constructed, find leaf nodes that are required (using breadth-first search), construct additional layer

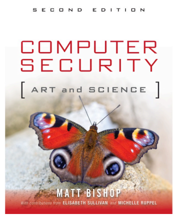
Example

- Assume Sage knows $E_k(m)$, $E_{k'}(k)$, k'
  - Nodes for these are available nodes
- Goal: determine m
  - Node representing m is required node
- Tree construction:
  - To get m, use k to decrypt $E_k(m)$ (left tree)
  - To get k, determine if it is encrypted and if so, try to decrypt it (right tree)
- Now all leaves are available nodes



Schneier’s Attack Trees

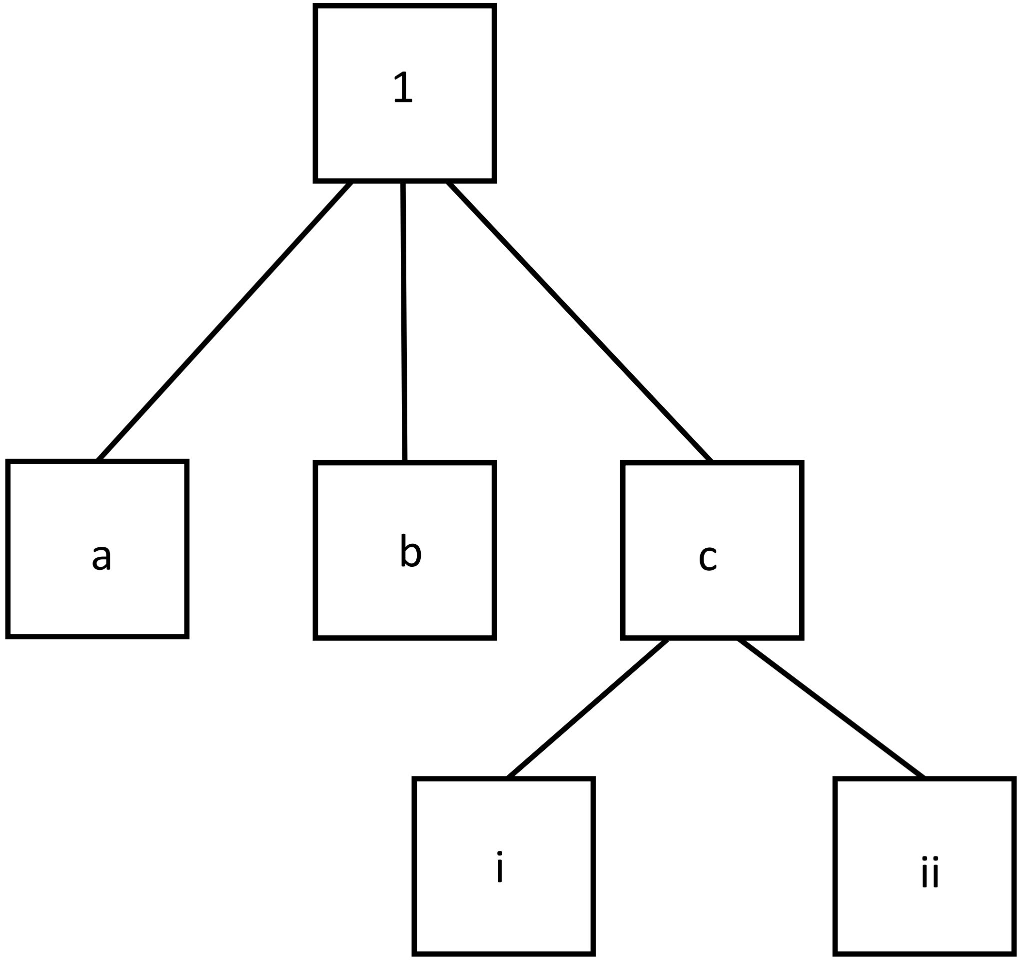
- Two types of nodes
  - *And* nodes require all children to be satisfied before it is satisfied
  - *Or* nodes require at least 1 of its children to be satisfied before it is satisfied
  - *Weight* of node indicates some relevant characteristic, like difficulty of satisfying node
    - Weights of interior nodes depend upon weights of child nodes
    - Weights of leaf nodes assigned externally
- Goal represented as root node of set of tree
- Determine the steps needed to satisfy the goal
  - These become children of the root
- Repeat that step for each child
  - Stop when leaf nodes are at appropriate level of abstraction



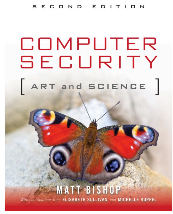
Example: Reading PGP-Encrypted Message

- Sage wants to read message Skyler sends to Caroline
- Five ways:
    1. Read message before Skyler encrypts it
    2. Read message after Caroline decrypts it
    3. Break encryption used to encrypt message
    4. Determine symmetric key used to encrypt message
    5. Obtain Caroline's private key
- Focus on 2, read message after Caroline decrypts it

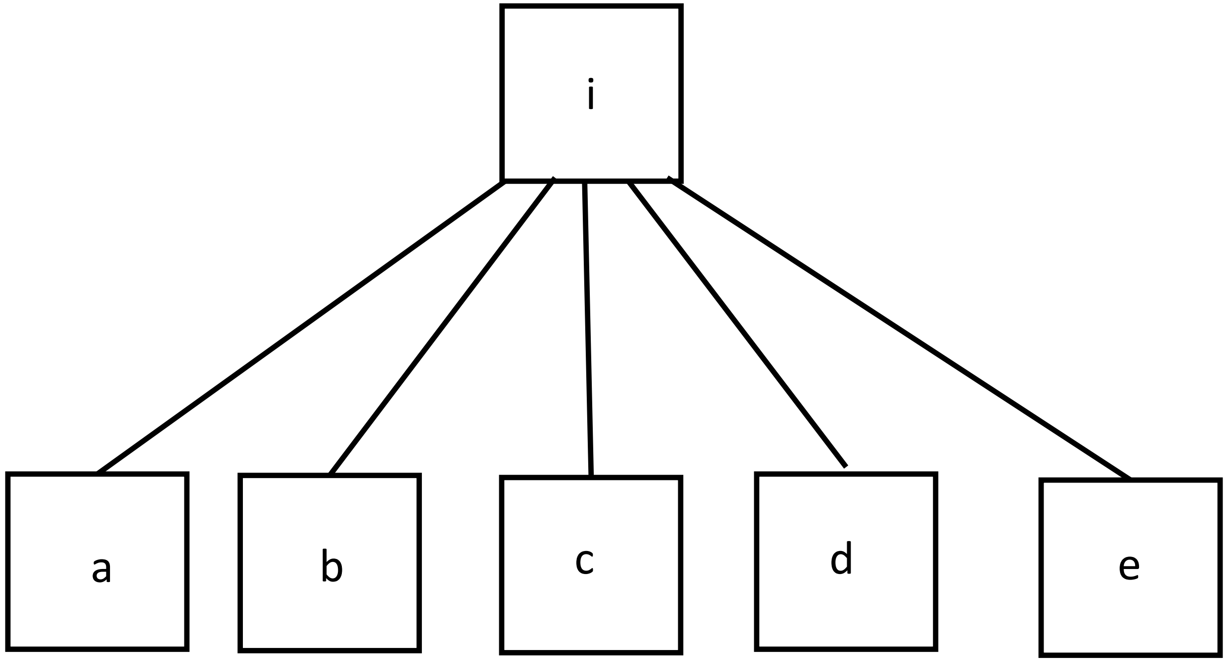
Beginning the Tree



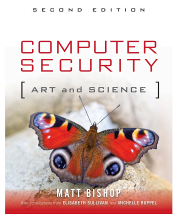
1. Read message after Caroline decrypts it
   a. Monitor Caroline’s outgoing mail; or
   b. Add a “Reply-To:” header (or change an existing one); or
   c. Compromise Caroline’s computer and read the decrypted message
      i. Compromise Caroline’s computer; and
      ii. Read the decrypted message



Next Layer

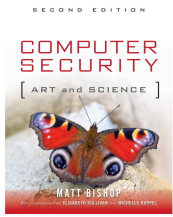


i. Read message after Caroline decrypts it
   a. Copy decrypted message from memory; or
   b. Copy decrypted message from secondary storage; or
   c. Copy decrypted message from backup; or
   d. Monitor network to observe Caroline sending the plaintext message; or
   e. Use a Van Eyk device to monitor the display of the message on Caroline's screen as it is displayed there



Textual Representation

1. Read a message that Skyler is sending to Caroline. (OR)
    1.1. Read the message before Skyler encrypts it.
    1.2. Read the message after Caroline decrypts it. (OR)
        1.2.1. Monitor Caroline’s outgoing mail.
        1.2.2. Add a “Reply-To” field to the header (or change the address in the existing “Reply-To” field).
        1.2.3. Compromise Caroline’s computer and read the decrypted message. (AND)
            1.2.3.1. Compromise Caroline’s computer. (OR)
                1.2.3.1.1. Copy decrypted message from memory.
                1.2.3.1.2. Copy decrypted message from secondary storage.
                1.2.3.1.3. Copy decrypted message from backup.
                1.2.3.1.4. Monitor network to observe Caroline sending the cleartext message.
                1.2.3.1.5. Use a Van Eck device to monitor the display of the message on Caroline’s monitor as it is displayed.
            1.2.3.2. Read the decrypted message.
    1.3. Break the encryption used to encrypt the message.
    1.4. Determine the symmetric key used to encrypt the message.
    1.5. Obtain Caroline’s private key.

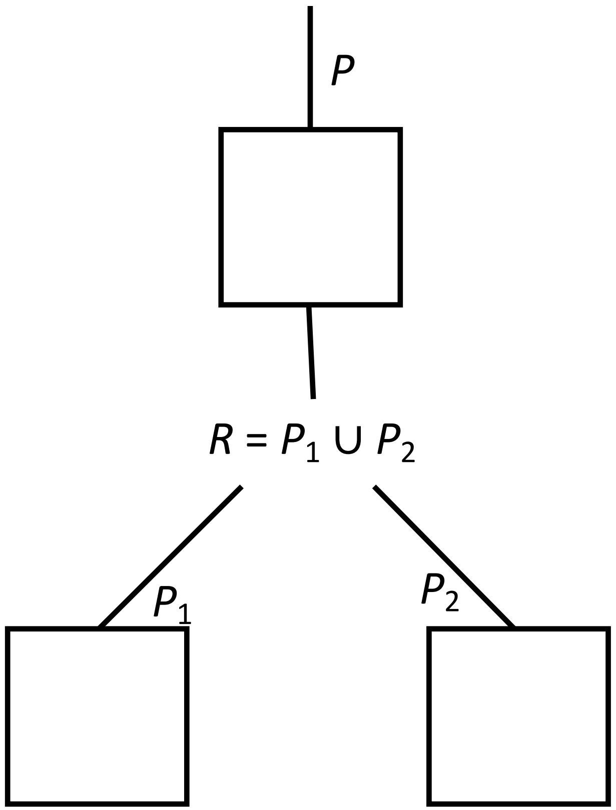


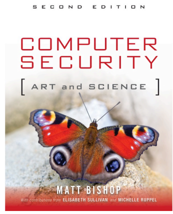
Requires/Provides Model

- Generalization of attack trees
- Based on *capabilities*, semantic objects encapsulating semantically typed attributes
  - Represent information or a situation to advance an attack
- *Concept* is a set *C* of capabilities and a mapping from *C* to another set of capabilities that are provided
  - Description of subgoal of attack
  - Attacker has a set of *required* capabilities *R* to reach subgoal; it then acquires a set *P* of provided capabilities

Concept

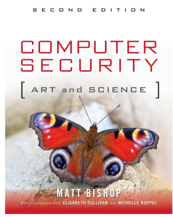
- *Concept* is a set *R* of capabilities and a mapping from *R* to another set *P* of capabilities that are provided
  - Description of subgoal of attack
- Interpretation: attacker has a set of *required* capabilities *R* to reach subgoal; it then acquires a set *P* of *provided* capabilities



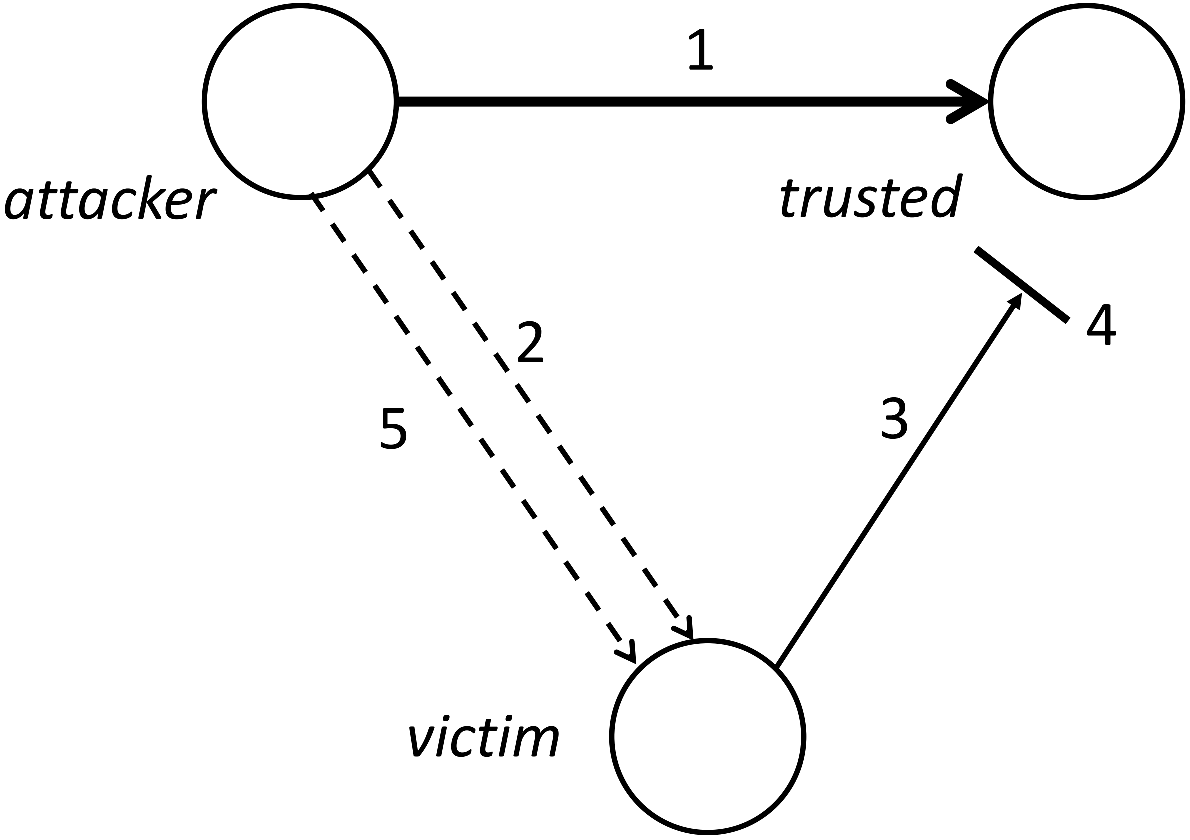


Concept

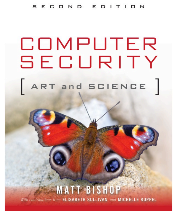
- Captures *effect* of attack
  - How the attack works (ie, how capabilities are required) irrelevant to concept; that attacker has them is what matters
- Moves away from having to know every method of attack to get to a step
  - Concept embodies the step, so all model needs is required capabilities
- Can compose attacks based solely on effects and not methods of attack



Example: *rsh* Attack

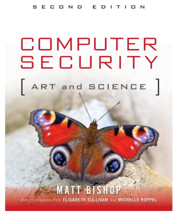


1. *attacker* launches a DoS against *trusted*
2. *attacker* sends *victim* forged SYN, apparently from *trusted*
3. *victim* sends SYN/ACK to *trusted*
4. It never gets there due to DoS
5. *attacker* sends forged SYN/ACK to *trusted*, with command in data segment of packet
   - Need to know right sequence number
   - If so, causes command to be executed as though *trusted* requested it



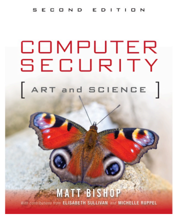
Example: *rsh* Attack

- *Requires* capability: blocking of a connection between the *trusted* and *victim* hosts
  - Contains source address, destination address
  - Also time interval indicating when communication is blocked (ie, when the DoS attack is under way, and how long it lasts)
- *Provides* capability: execute command on *victim* host as if command were from *trusted* host
- *Concept*: spoof *trusted* host to *victim* host



JIGSAW Language

- Implements requires/provides model
- Capabilities: sets of typed attributes and values
  - **`extern`** keyword means it is defined elsewhere
- Concepts: two sets of capabilities
  - Required capabilities in **`requires`** block
  - Provided capabilities in **`provides`** block
  - **`action`** block lists actions to take when a concept is active

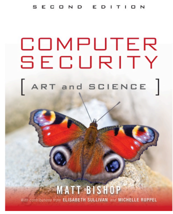


Example: JIGSAW Representation of *rsh* Attack

```
capability nosend is
      true_src, src, dst: type Host; # attacker, trusted, victim
      using: type Service;           # service to be exploited
end.
```

Structure of a capability:

- *using* is command to be executed, exploiting a service (here, *rsh*)



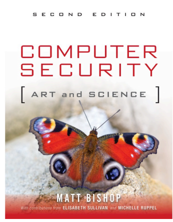
Example: JIGSAW Representation of *rsh* Attack

```
concept rsh_connection_spoofing is
      requires
            TP: type Trusted_Partner;       #- trusted host
            SA: type Active_Service;        #- service (here, rshd)
            PPS: type Prevent_Packet_Send;
            FPS: type Forged_Packet_Send;
            extern SNP: type SeqNumProbe;
```

PPS: capability for *true_src* to block *src* host receiving packets from *dst*

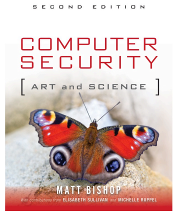
FPS: capability for *true_src* to send forget packet to *dst*

SNP: capability for *true_src* to determine next sequence number of *dst*



Example: JIGSAW Representation of *rsh* Attack

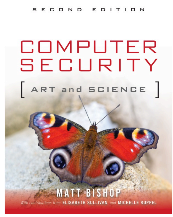
```
with          #- These instantiate the capabilities
      TP.service is RSH,       #- service is RSH
      PPS.host is TP.trusted,         #- blocked host = trusted host
      FPD.dst.host is TP.trustor,     #- spoofed packets go to host
                                      #-  trusting TP
      FPS.src is [PPS.host, PPS.port], #- apparent source of forged
                                      #-  packets is blocked
      SNP.dst is [SA.host, SA.port],  #- probed host must be
      SA.port is TCP/RSH,             #-  running RSH on usual port
      SA.service is RSH,
      SNP.dst is FPS.dst              #- forged packets go to probed
      active(FPS) during active(PPS)  #-  host while DoS of trusted
                                      #-  host is active
```



Example: JIGSAW Representation of *rsh* Attack

To meet **`requires`** conditions, relationships in **`with`** block must hold:

- Trusted host must be running *rsh* servicve
- Attacker must be able to block trusted host from sending packets to victim
- Attacker must be able to send spoofed packets ostensibly from trusted host to victim
- Attacker must know sequence number of packet victim sends to trusted host
- When attack on victim is being carried out, attack on trusted host must also be active

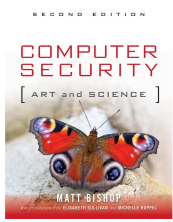


Example: JIGSAW Representation of *rsh* Attack

```
requires
        PSC: type push_channel;
        REX: type remote_execution;
```

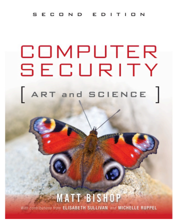
PSC: capability to send code, commands to *dst*

REX: capability to execute that code, commands on *dst*



Example: JIGSAW Representation of *rsh* Attack

```
with            #- These set the new capabilities
      PSC.src <- FPS.true_src,           #- capability to move code from
      PSC.dst <- FPS.dst,                #-  attacker to rsh server
      PSC.true_src <- FPS.true_src,      #-  (victim)
      PSC.using <- rsh;
      REX.src <- FPS.true_src,           #- capability to execute code,
      REX.dst <- FPS.dst,                #-  commands on rsh server
      REX.true_src <- FPS.true_src,      #-  (victim)
      REX.using <- rsh;
end;
action
      true -> report("rsh connection spoofing: " + TP.hostname)
end;
```

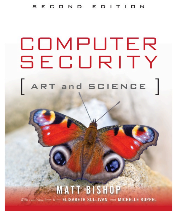


Example: JIGSAW Representation of *rsh* Attack

- When all conditions in **`requires`** block satisfied, concept *rsh_connection_spoofing* is realized
- Attacker gets capabilities defined in **`provides`** section
  - Here, *PSC* and *REX* capabilities
- Events in **`action`** block executed
  - Here, message is printed to alert observer an *rsh* spoofing attack under way

Attack Graphs

- Describe attacks in terms of a general graph
  - Generalization of attack trees
- Used to represent attacks, detect attacks, guide penetration testing



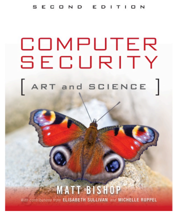
Attack Graph and Penetration Testing

Here attack graph is a Petri net

- Nodes $P = \{ p_1, \ldots, p_n \}$ states of entities relevant to system under attack
- Edges $T = \{ t_1, \ldots, t_m \}$ transitions between states
- Token on a node means attacker has appropriate control of that entity
- Tokens move to indicate progress of attack
- If node p_i precedes node p_j, attacker must get control of p_i before it can get control of p_j

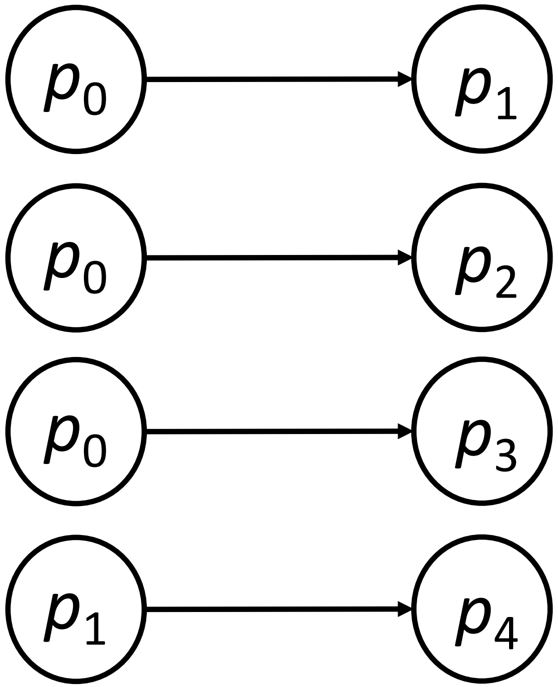
Attack Graph and Penetration Testing

- McDermott: hypothesize individual flaws as 2 nodes connected by transition; then examine nodes for relationships that allow them to be linked



Attack Graph and *rsh* Attack

- First steps in attack:



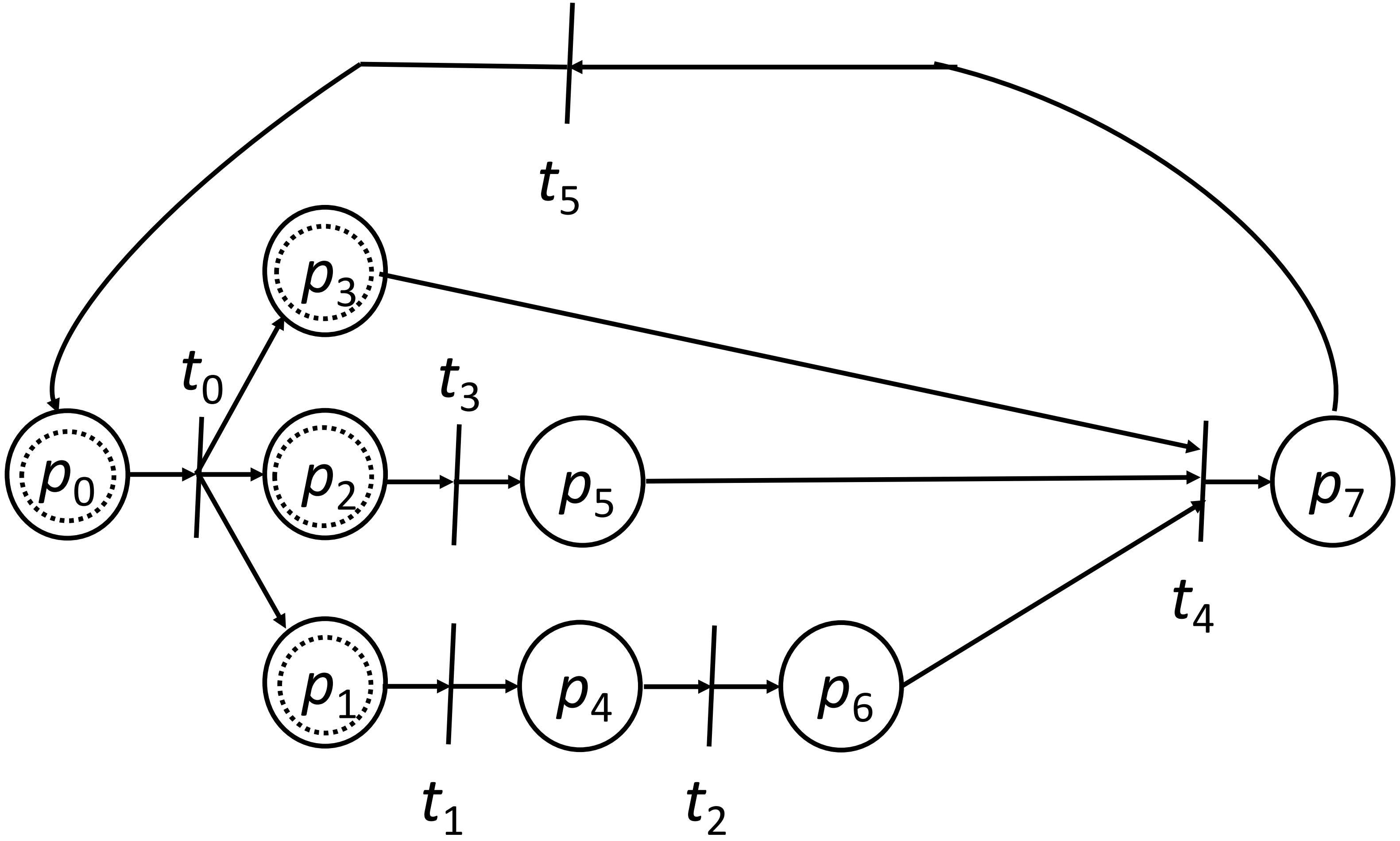
Initial scan of target

Identify an unused address

Establish that target trusts another host

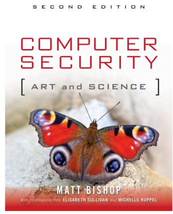
Forge SYN packet

Attack Graph and *rsh* Attack



Petri net represents *rsh* attack

1. Before attack
2. After attack



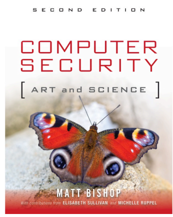
Attack Graph and *rsh* Attack

States

- p_0: starting state
- p_1: found unused address on target network
- p_2: found trusted host
- p_3: found target that trusts the trusted host
- p_4: forged SYN packet created
- p_5: able to predict TCP sequence numbers of target host
- p_6: saturated state of network connections of trusted host
- p_7: final (compromised) state

Transitions

- t_0: attacker scanning system (splits into 3 transitions)
- t_1: attacker creating forged SYN packet
- t_2: attacker launching SYN flood against trusted host
- t_3: attacker figuring out how to predict victim's TCP sequence numbers
- t_4: forged SYN packet created
- t_5: attacker modifying trusted host file on victim
  - Attacker can now get *root* access on victim

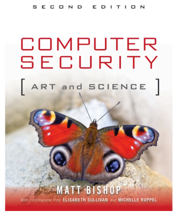


Attack Graph and *rsh* Attack

- Attack starts at p_0
- t_0 splits into 3 transitions, as on success, 3 states of interest
- Need to instantiate all 3 states:
  - p_1: find unused address on target
  - p_2: find trusted host
  - p_3: find target that trusts trusted host
- t_1 is creating forged SYN packet
  - Transition from p_1 to p_4
- t_2 is attacker launching SYN flood (DoS) against trusted host
  - Transition from p_4 to p_6

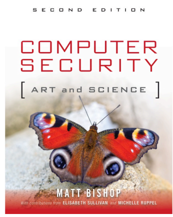
Attack Graph and *rsh* Attack

- t_3: attacker figuring out how to predict victim's TCP sequence numbers
  - Transition from p_2 to p_4
- t_4: attacker launches attack using entities above
  - Transition from p_3, p_5, and p_6 to p_7
- t_5: attacker executes command
  - Example: modifying trusted hosts file to be able to get *root*



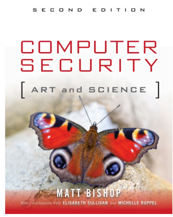
Intrusion Response

- Incident prevention
- Intrusion handling
  - Containment phase
  - Eradication phase
  - Follow-up phase
- Incident response groups



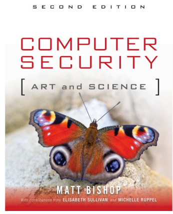
Incident Prevention

- Identify attack *before* it completes
- Prevent it from completing
- Jails useful for this
- IDS-based methods detect beginning of incidents and block their completion
- Diversity increases difficulty of attacks succeeding



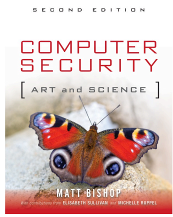
Jailing

- Attacker placed in a confined environment that looks like a full, unrestricted environment
- Attacker may download files, but gets bogus ones
- Can imitate a slow system, or an unreliable one
- Useful to figure out what attacker wants
- MLS systems provide natural jails



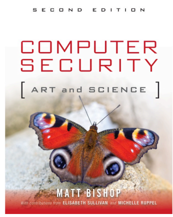
Example Jail

- Cheswick recorded a break-in attempt using the SMTP server
- He created a very restrictive account, put the attacker in it
  - Monitored actions, including who the intruder was attacked
    - None succeeded and Cheswick notified the sysadmins of those systems
  - File system visible to attacker resembled UNIX file system
    - Lacked some programs that provided system information, or could reveal deception
    - Access times to critical files masked
- At request of management, finally shut down jail



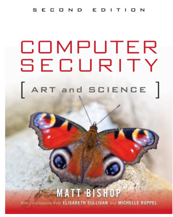
IDS-Based Method

- Based on IDS that monitored system calls
- IDS records anomalous system calls in locality frame buffer
  - When number of calls in buffer exceeded user-defined threshold, system delayed evaluation of system calls
  - If second threshold exceeded, process cannot spawn child
- Performance impact should be minimal on legitimate programs
  - System calls small part of runtime of most programs



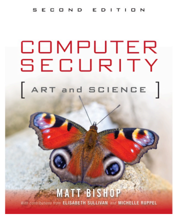
Example Implementation

- Implemented in kernel of Linux system
- Test #1: *ssh* daemon
  - Detected attempt to use global password installed as back door in daemon
  - Connection slowed down significantly
  - When second threshold set to 1, attacker could not obtain login shell
- Test #2: *sendmail* daemon
  - Detected attempts to break in
  - Delays grew quickly to 2 hours per system call



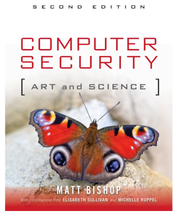
Diversity

- Monoculture: an attack that works against one system works against all
- Diverse culture: one attack will not compromise all systems
  - Many different types of systems
  - Also can vary system configurations



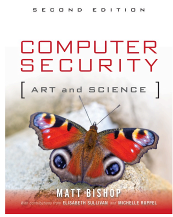
Attack Surface and Moving Target Defense

- *Attack surface*: set of entry points, data that attackers can use to compromise system
- Usual approach: harden system to reduce attack surface, so more difficult for attackers to succeed
- *Defender's dilemma*: asymmetry between attacker, defender introduced by attack surface being non-empty
- *Moving target defense* (*MTD*): *change* attack surface while system runs
  - Attacks that work one time may not work another time
  - Reconnaissance data gathered as a prelude to attack no longer accurate after changes



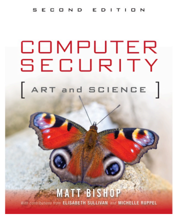
Example: IP Address Hopping

- Client needs to contact server
- Component maps destination IP address, port number to different IP address, port number
  - These are chosen (pseudo)randomly
- When packet reaches network, another component remaps IP destination IP address, port number to real IP address, port number
  - If client, server on different networks, changed IP address must be on the same network as server
  - Mapping changes frequently (e.g., every minute)
- Attacker monitoring network cannot obtain real IP address, port number of server



Example: Mapping for Port Hopping

1. Divide time into discrete intervals of length τ at times $t_0, \ldots, t_i, \ldots$
   - At time k, port $p_k = f(k, s)$, where s is seed and f a pseudorandom number generator
   - Ports overlap at interval boundaries
   - So if L amount of overlap, p_k valid over interval $[t_k - L_\tau, t_k + L_\tau]$
2. Use encryption algorithm for mapping
   - Low-order octet of IP address and port number enciphered
   - High octet of result is low-order octet of IP address, rest is port number
   - Remapping just reverses encryption to get real IP address, port number

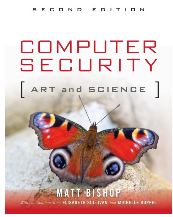


Notes on Moving Target Defenses

- Network-based MTDs
  - Must rely on randomness to prevent attacker from predicting changes to attack surface
  - Defender must distinguish between clients authorized to connect and clients not authorized to connect
- Host-based MTDs
  - Also must rely on randomness to prevent attacker from predicting changes to attack surface
  - Here, attacker is typically authorized to have access to some account in some way
  - Attack surface is within host

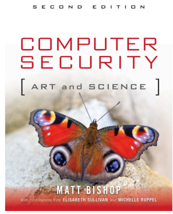
Address Space Layout Randomization

- Executables have several segments
  - Exact number, layout depends on compiler and systems
- When loaded into memory, segments arranged in particular order
  - That way, positions of variables, functions fixed in virtual memory
  - Attack tools exploit knowing where these are
- *Address space layout randomization* (ASLR) perturb the placement of segments, variables, functions
  - Then attack tools exploiting knowing where segments, variables, functions won’t work



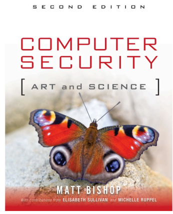
Address Space Layout Randomization

- Key question: how is perturbation done?
- Simplest: randomize placement of segments in virtual memory
- Others
    - Randomize order and/or locations of variables, functions within segments
    - Add rando amount of space between variables, between functions
- Effectiveness depends on entropy introduced into address space
    - 32-bit Linux: uncertainty of segment base typically 16 bits, so easy to use brute force attack
    - 64-bit Linux: uncertainty of segment base typically 40 bits, so a search takes long enough that it is likely to be detected



Intrusion Handling

- Restoring system to satisfy site security policy
- Six phases
  - *Preparation* for attack (before attack detected)
  - *Identification* of attack
  - *Containment* of attack (confinement)
  - *Eradication* of attack (stop attack)
  - *Recovery* from attack (restore system to secure state)
  - *Follow-up* to attack (analysis and other actions)
- Discussed in what follows

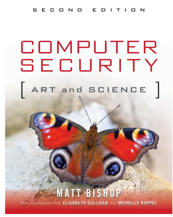


Containment Phase

- Goal: limit access of attacker to system resources
- Two methods
    - Passive monitoring
    - Constraining access

Passive Monitoring

- Records attacker’s actions; does *not* interfere with attack
  - Idea is to find out what the attacker is after and/or methods the attacker is using
- Problem: attacked system is vulnerable throughout
  - Attacker can also attack other systems
- Example: type of operating system can be derived from settings of TCP and IP packets of incoming connections
  - Analyst draws conclusions about source of attack



Constraining Actions

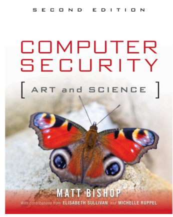
- Reduce protection domain of attacker
- Problem: if defenders do not know what attacker is after, reduced protection domain may contain what the attacker is after
  - Stoll created document that attacker downloaded
  - Download took several hours, during which the phone call was traced to Germany

Example: Honeypots

- Entities designed to entice attacker to do something
- *Honeyfiles, honeydocuments*: designed to entice attackers to read or download it
  - Stoll used this to keep intruder on line long enough to be traced (internationally)
- *Honeypots, decoy servers*: servers offering many targets for attackers
  - Idea is attackers will take actions on them that reveal goals
  - These are instrumented, monitored closely
- *Honeynets*: like honeypots, but a full network
  - Treated like honeypots

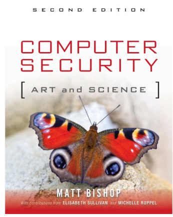
Deception

- Cohen’s Deception Tool Kit
  - Creates false network interface
  - Can present any network configuration to attackers
  - When probed, can return wide range of vulnerabilities
  - Attacker wastes time attacking non-existent systems while analyst collects and analyzes attacks to determine goals and abilities of attacker
  - Experiments showed deception is effective response to keep attackers from targeting real systems



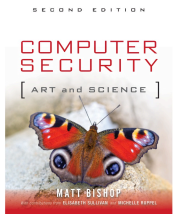
Example: Honeypot Project

- International project created to learn about attacker community
- Phase 1: identify common threats against specific OSes, configurations
  - Gen-I honeypots crude but very effective
- Phase 2: collect data more efficiently
  - Gen-II honeypots easier to deploy and harder to detect
- Used to gather attack signatures, enable defenders to handle attacks without endangering production systems



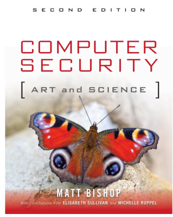
Eradication Phase

- Usual approach: deny or remove access to system, or terminate processes involved in attack
- Use wrappers to implement access control
  - Example: wrap system calls
    - On invocation, wrapper takes control of process
    - Wrapper can log call, deny access, do intrusion detection
    - Experiments focusing on intrusion detection used multiple wrappers to terminate suspicious processes
  - Example: network connections
    - Wrapper around servers log, do access control on, incoming connections and control access to Web-based databases



Firewalls

- Mediate access to organization’s network
  - Also mediate access out to the Internet
- Example: Java applets filtered at firewall
  - Use proxy server to rewrite them
    - Change “<applet>” to something else
  - Discard incoming web files with hex sequence CA FE BA BE
    - All Java class files begin with this
  - Block all files with name ending in “.class” or “.zip”
    - Lots of false positives



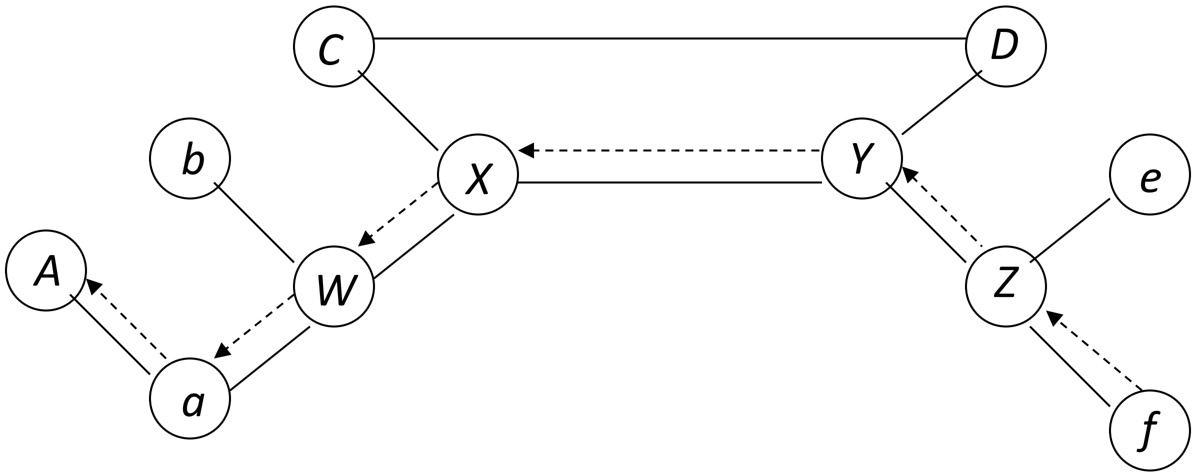
Intrusion Detection and Isolation Protocol

- Coordinates reponse to attacks
- *Boundary controller* is system that can block connection from entering perimeter
  - Typically firewalls or routers
- *Neighbor* is system directly connected
- *IDIP domain* is set of systems that can send messages to one another without messages passing through boundary controller

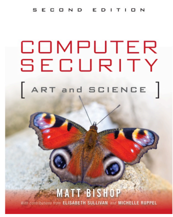
Protocol

- IDIP protocol engine monitors connection passing through members of IDIP domains
  - If intrusion observed, engine reports it to neighbors
  - Neighbors propagate information about attack
  - Trace connection, datagrams to boundary controllers
  - Boundary controllers coordinate responses
    - Usually, block attack, notify other controllers to block relevant communications

Example

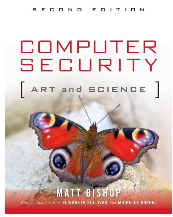


- *C*, *D*, *W*, *X*, *Y*, *Z* boundary controllers
- *f* launches flooding attack on *A*
- Note after *X* suppresses traffic intended for *A*, *W* begins accepting it and *A*, *b*, *a*, and *W* can freely communicate again



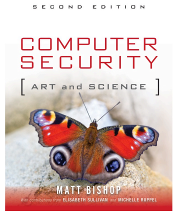
Follow-Up Phase

- Take action external to system against attacker
  - Thumbprinting: traceback at the connection level
  - IP header marking: traceback at the packet level
  - Counterattacking



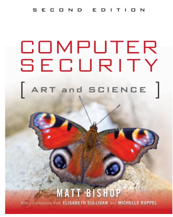
Thumbprinting

- Compares contents of connections to determine which are in a chain of connections
- Characteristic of a good thumbprint
  1. Takes as little space as possible
  2. Low probability of collisions (connections with different contents having same thumbprint)
  3. Minimally affected by common transmission errors
  4. Additive, so two thumbprints over successive intervals can be combined
  5. Cost little to compute, compare



Example: Foxhound

- Thumbprints are linear combinations of character frequencies
  - Experiment used *telnet*, *rlogin* connections
- Computed over normal network traffic
- Control experiment
  - Out of 4000 pairings, 1 match reported
    - So thumbprints unlikely to match if connections paired randomly
    - Matched pair had identical contents

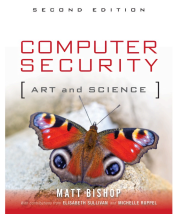


Experiments

- Compute thumbprints from connections passing through multiple hosts
  - One thumbprint per host
- Injected into a collection of thumbprints made at same time
  - Comparison immediately identified the related ones
- Then experimented on long haul networks
  - Comparison procedure readily found connections correctly

IP Header Marking

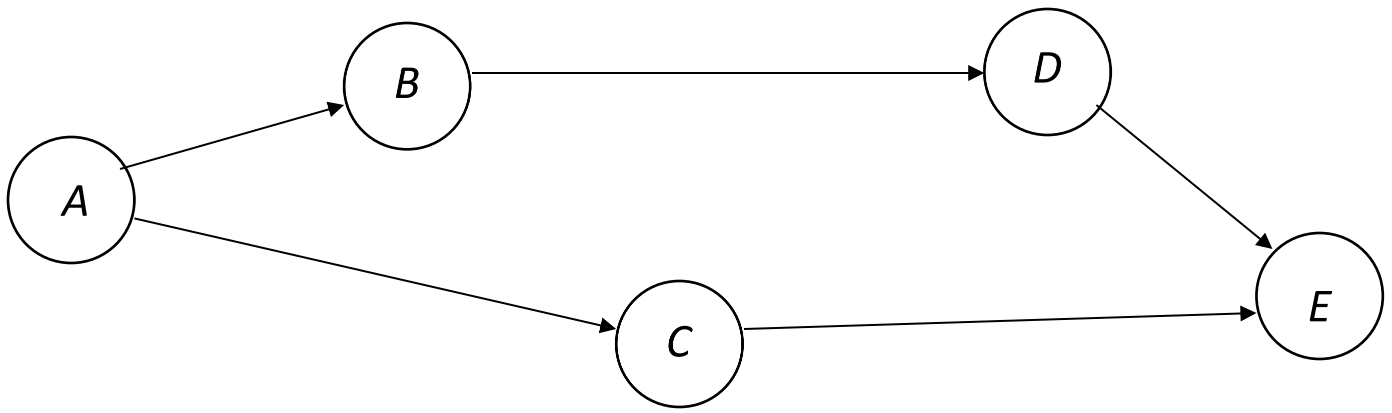
- Router places data into each header indicating path taken
- When do you mark it?
  - Deterministic: always marked
  - Probabilistic: marked with some probability
- How do you mark it?
  - Internal: marking placed in existing header
  - Expansive: header expanded to include extra space for marking



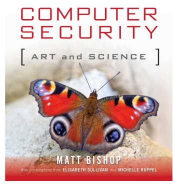
Example: Probabilistic Scheme

- Expand header to have *n* slots for router addresses
- Router address placed in slot *s* with probability *sp*
- Use: suppose SYN flood occurs in network

Use

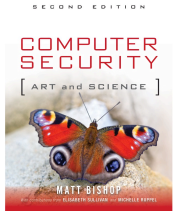


- *E* SYN flooded; 3150 packets could be result of flood
- 600 (*A*, *B*, *D*); 200 (*A*, *D*); 150 (*B*, *D*); 1500 (*D*); 400 (*A*, *C*); 300 (*C*)
  - *A*: 1200; *B*: 750; *C*: 700; *D*: 2450
- Note traffic increases between *B* and *D*
  - *B* probable culprit



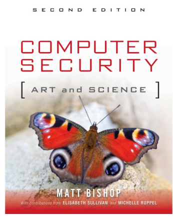
Algebraic Technique

- Packets from *A* to *B* along path *P*
- First router labels *j*th packet with x_j
- Routers on *P* have IP addresses $a_0, \ldots, a_n$
- Each router a_i computes $Rx_j + a_i$, R being current mark $a_0x_j^i + \ldots + a_{i-1}$ (Horner's rule)
  - At *B*, marking is $a_0x^n + \ldots + a_n$, evaluated at x_j
- After n+1 packets arrive, can determine route



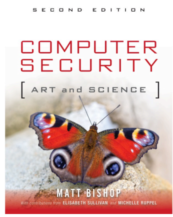
Alternative

- Alternate approach: at most *l* routers mark packet this way
- *l* set by first router
- Marking routers decrement it
- Experiment analyzed 20,000 packets marked by this scheme; recovered paths of length 25 about 98% of time



Problem

- Who assigns x_j?
  - Infeasible for a router to know it is first on path
  - Can use weighting scheme to determine if router is first
- Attacker can place arbitrary information into marking
  - If router does not select packet for marking, bogus information passed on
  - Destination cannot tell if packet has had bogus information put in it

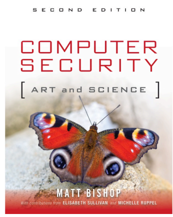


Counterattacking

- Use legal procedures
  - Collect chain of evidence so legal authorities can establish attack was real
  - Check with lawyers for this
    - Rules of evidence very specific and detailed
    - If you don’t follow them, expect case to be dropped
- Technical attack
  - Goal is to damage attacker seriously enough to stop current attack and deter future attacks

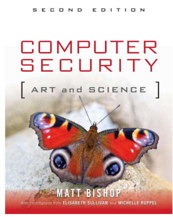
Consequences

1. May harm innocent party
   - Attacker may have broken into source of attack or may be impersonating innocent party
2. May have side effects
   - If counterattack is flooding, may block legitimate use of network
3. Antithetical to shared use of network
   - Counterattack absorbs network resources and makes threats more immediate
4. May be legally actionable



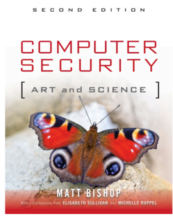
Example: Counterworm

- Counterworm given signature of real worm
  - Counterworm spreads rapidly, deleting all occurrences of original worm
- Some issues
  - How can counterworm be set up to delete *only* targeted worm?
  - What if infected system is gathering worms for research?
  - How do originators of counterworm know it will not cause problems for any system?
    - And are they legally liable if it does?



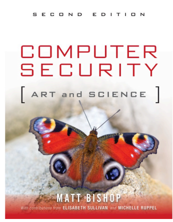
Incident Response Groups

- *Computer security incident response team* (CSIRT): team established to assist and coordinate responses to a security incident among a defined constituency
  - "Constituency" defined broadly; may be vendor, company, sector such as financial or academic, nation, etc.
- Mission depends in large part on constituency
  - Critical part: keep constituency informed of services CSIRT provides, how to communicate with CSIRT



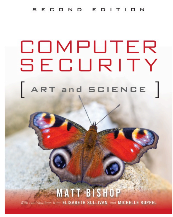
Example: CERT/CC

- Grew out of Internet worm, when many groups dealt with it and had to communicate with one another
  - In some cases, they did not know about other groups, what they are doing
  - Sometimes trusted third party did introduction
- Raised concerns of how to communicate and coordinate responses to future events
- Led to development of Computer Emergency Response Team (CERT, later CERT/CC)



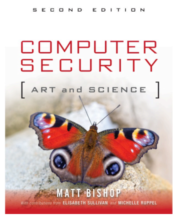
CSIRT Missions

1. *Publication*: publish policies, procedures about what it can do, how it will communicate with constituency, how constituency can communicate it
2. *Collaboration*: collaborate with other CSIRTs to gather, disseminate information about attacks, respond to attacks
3. *Secure communication*: preserve credibility; ensure constituency they are communicating with CSIRT and not masquerader; and CSIRT must be sure it is dealing with affected members of constituency and other CSIRTs, not masqueraders



How a CSIRT Functions

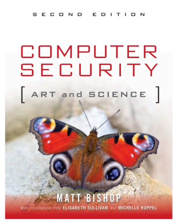
- Policy defines what it will, will not do
- Plan how to respond to incidents, driven by needs and constraints of constituents
  - Avoid solely technical approach
  - Couple that with strategic analysis to find organizational issues contributing to attack or hindering appropriate responses
  - Understanding incident involves non-technical aspects of organization such as people, resources, economics, laws and regulations
- Disseminate information to prevent, limit attacks
  - Include vulnerability reports



Digital Forensics

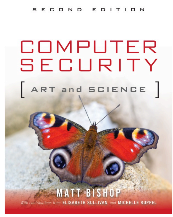
The science of identifying and analyzing entities, states, state transitions of events that have occurred or are occurring

- Also called *computer forensics*
- Usually done to figure out what caused an anomaly or understand nature of attack: how did attackers (try to) enter system, what they did, and how defenses failed
- *Legal forensics* may include digital forensics
  - Here, analysts must acquire information and perform analysis in such a way that what is uncovered can be admitted into a legal proceeding



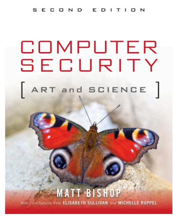
Goals of Forensics Principles

- *Locard's Exchange Principle*: every contact leaves a trace
- Forensics principles create environment in which Locard's Exchange Principle holds
- Must consider entire system
  - Attack on one component may affect other components
  - Multistage attacks leverage compromise of a component to compromise another
  - Attack may have effects that analyst does not expect



Principle 1: Consider the Entire System

- Analyst needs access to information the intruder had before, after attack
  - Includes changes to memory, kernel, file systems, files
- Rarely recorded continuously, so information incomplete
- Logs also often omit useful information
  - Record connections, states of connections, services, programs executed
  - Omit directories searched to find dynamically loaded libraries, or which ones are loaded; also omit memory contents during program execution
  - Application logging also may not log security-relevant events



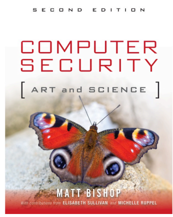
Principle 2: Assumptions Should Not Control What Is Logged

- Analysts work from logs capturing information before, during, after incident being analyzed
  - If assumptions guide what is being logged, information may be incomplete
- Record enough information to reconstruct system state at any time
  - Virtual machine introspection great for this

Example: ExecRecorder

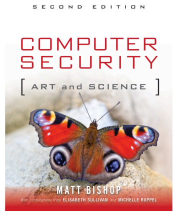
Architecture to enable replay of events with minimal overhead and no changes to operating system

- Hypervisor Bochs contains checkpoint, logging, replay mechanisms
  - These are invisible to operating system running in Bochs
- Checkpoint component takes snapshots of system state
- Logging component records nondeterministic events to enable them to be reproduced *exactly*
- Replay component reconstructs and restores state of system, and system activity occurs from that point on



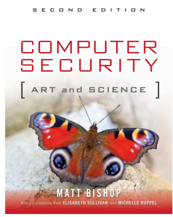
Principle 3: Consider the Effects of Actions As Well As the Actions

- Aim is to establish what system did as well as what attacker did
- Logs record actions, sometimes effects, but almost never causes allowing actions to occur
- Example: remote attacker gains enough access to execute commands on other systems
  - Logs show which server she went to, commands issued
  - Logs do not show vulnerability that enabled attacker to succeed, so others may exploit the same vulnerability



Principle 4: Context Assists in Understanding Meaning

- Same action may cause 2 different effects when executed in 2 different contexts
- Example: LINUX command typed at keyboard (not full path name of command)
  - What gets executed depends on search path, contents of file system
- Example: file system monitoring tool logging access to files by file name
  - The same name may refer to 2 different files (refers to file X, then file X deleted and a new file X created)



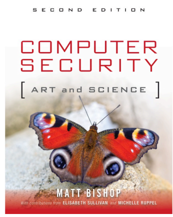
Principle 5: Information Must Be Processed, Presented in an Understandable Way

- Those who need to understand the forensic analysis can do so
- First audience: analysts
  - Interfaces to forensic tools must be designed with usability in mind, and indicate where gaps in data, analysis are
  - Presentation of results must also be clear to a technical audience
- Second audience: non-technical audience
  - Provide information in a way that the audience can understand what happened, how it happened, what the effects of the attack were, the level of assurance that the data, analysis is correct
  - May need to present evidence in a way appropriate to a particular audience, such as legal audiences

Practice

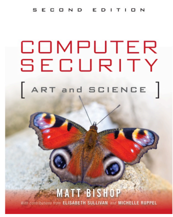
Typically 4 steps to reconstruct state of system and sequence of actions of interest

1. Capture, preserve current state of system, network data
2. Extract information about that state and prior states
    - Reverse these steps if system is active; in this case, state will be approximate as gathering data takes time and state may change during that process
3. Analyze data to determine sequence of actions, objects affected, and how they are affected
4. Prepare, report results of analysis to intended audience



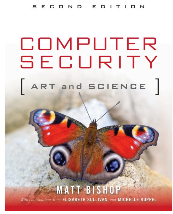
Gathering Data

- Get a complete image of all components
- If infeasible (because compromise discovered after it is done, or system is active), get as complete an image as possible
  - May include disk images, backups, stored network or IDS data
- Be sure to make cryptographic hash of all data
  - That way, you and others can verify data is unaltered after being checksummed



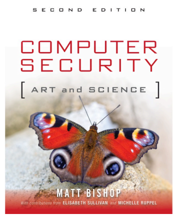
Example: Gathering Data

- Disk is full, but space used by files much less than size of disk
- Sysadmin removes dick, mounts it read-only on another system
- Sysadmin creates image of it on some other media
  - On a second, previously wiped, disk
- Sysadmin creates cryptographic checksum of image
  - Can be used to show image was not changed since its creation
- Sysadmin uses a different program to recompute checksum and verifies it matches previously computed checksum
  - Used to ensure cryptographic checksum is correct



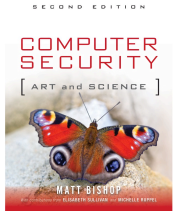
Persistent vs. Volatile Data

- Persistent data: remains when system or data storage is powered off
  - Data on hard drive or secondary storage
- Volatile data: transient, disappearing at some point in time (like when system is powered off)
  - Data in memory
  - More difficult to capture than persistent data



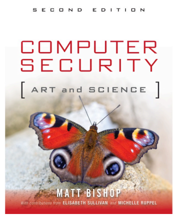
Capturing Volatile Data

- Problem: using software to capture memory contents alters memory
- One approach: use specialized hardware
  - Carrier and Grand built custom PCI card; attached to bus
    - When computer boots, card configures itself, disables its controller so it is invisible to programs scanning PCI bus
    - Throw switch, card re-enables controller, suspends CPU, dumps memory to a non-volatile storage medium
    - When done, disables its controller and restart CPU



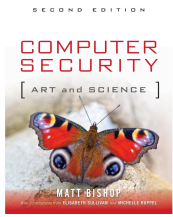
Capturing Volatile Data

- Second approach: store memory-reading software in trusted location
  - Attacker cannot alter it
  - Software freezes operating system and all associated processes, captures and dumps memory contents, unfreezes operating system and all associated processes
  - Intel IA-32 platforms have System Management Mode to provide such an area
    - SMM has software drivers for standard network PCI card
    - SMM grabs contents of CPU registers, and PCI grabs contents of memory; these transmitted to waiting server
    - Using SMM suspends operating system so memory contents in consistent state



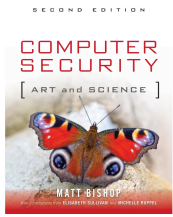
Capturing Volatile Data

- Third approach: put acquisition software between operating system, hardware
  - Virtual machine introspection does this; to capture memory contents, virtual machine monitor stops VM, copies contents of memory
- Fourth approach: remanence effect
  - Memory retains contents for very short time after power lost
  - Cooling memory increases this time significantly
  - This used for forensics on Android phones



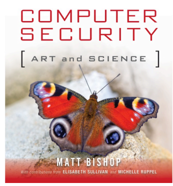
Extracting Information

- Analyze to produce a timeline
- Example for the disk mentioned earlier; work done from disk image
  1. Analysts obtain list of files on disk
  2. They check for deleted files; find several corresponding to undeleted files
  3. They examine free space; find large number of files there



Analyze the Data

- Goal is to answer specific questions that depend on nature of attack, resources involved, and the data
- Example for the disk mentioned earlier; information gathered from disk image
- Analysts examine files stored in free space as they are hidden; turn out to be copies of recently released movies
- Key question: how did they get there?
- Analysts extract log files of network server, user actions; find a login name with control characters in it, and no corresponding logout; possible buffer overflow
    - Validation: run login program, give it user name of 1000 characters; it crashes



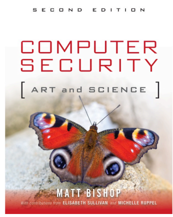
Analyze the Data

How did attackers gain access to system (to run login program)?

- Analysts examine server logs, server configuration files; nothing suspicious
- Analysts look through other network log files, find an entry made by a program starting the *telnet* service
  - This is a remote terminal interface and should never run
  - Find the program in a sysadmin's directory
- Analysts look at network logs
  - IDS captures packets, stored for 30 days
  - After that, deletes packet bodies and saves headers for 5 months

Analyze the Data

- Analysts look for *telnet* packets; find several, including one containing the user name matching the one with control characters
- Analysts copy these packets to separate file, create a textual representation in another file
  - And these are checksummed and saved on read-only media
- How did movies get put into free space?
  - Obvious answer: attackers simply deleted them or wrote them directly to free space
    - But then disk would not have been full as deleted blocks would simply be overwritten
  - More probable answer: attacker created file, opened it, deleted file from file system
    - Program checking disk space by traversing file hierarchy will miss it; looking at disk map won’t; this also explains discrepancy

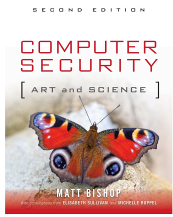


Report the Findings

Must take into account the audience (principle of presenting information in an understandable way)

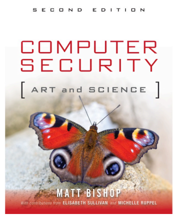
- If non-technical audience, report should say movie files stored in unused disk space, and give data on number of movies found, titles, and so forth
- If technical audience, also describe how movies stored, how they were found

This suggests preparing a detailed technical report for reference, then use that as basis for writing other reports as needed



Anti-Forensics

- *Anti-forensics*: the attempt to compromise the availability or usefulness of evidence to forensics process
- Goals:
  - Interfere with forensic analysis tools gathering information, by hiding data or obscuring type, sequence of evidence
  - Hinder the validation of authenticity of digital image
  - Exploit weaknesses in forensic analysis tools
  - Attacking users of forensic analysis tools, for example by crashing analyst’s system or increasing time needed to analyze data
  - Cast doubt on results of forensic analysis; will diminish its credibility in court, for example



Examples

- *timestomp*: enables user to change file access times
- *event_manager*: enables user to delete entries from log files
- JPEG image data compresses digital representation of image into multiple bands of transform coefficients, which generally follow a smooth distribution; altering image perturbs coefficients, so distribution different; anti-forensic tools add dithering to change coefficients back to approximate original one
- Forensic tool determines if Windows files are executable by looking at file extension (".exe") and first 2 bytes of file ("MZ"), so anti-forensics tools can just change the extension

Key Points

- Goal of modeling is to understand attacks
  - Attack trees, graphs, requires/provides models represent how attacks proceed
- Intrusion response occurs before, during, after attack
  - If before attack successful, system tries to prevent attack from succeeding
  - If during or after, intrusion must be handled
    - Confinement, eradication, follow-up
- Digital forensics analyzes detritus of attack to determine its effects, how it was carried out
  - Anti-forensics try to thwart this