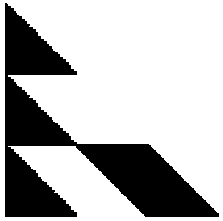
Attack Trees

Bruce Schneier

Counterpane Systems
101 East Minnehaha Parkway, Minneapolis, MN 55419
Phone: (612) 823 1098; Fax: (612) 823-1590
schneier@counterpane.com
http://www.counterpane.com

SANS Network Security 99
New Orleans, LA
8 October 1999



COUNTERPANE

Needs for Threat Modeling

- Understand what the attack goals are.
- Understand who the attackers are.
- Understand what attacks are likely to occur.
- Understand the security assumptions of a system.
- Understand where to best spend a security budget.

The Obvious Attacks Aren't Always the Actual Attacks

- Credit card fraud.
- Cell phone fraud.
- DVD / CD / software copy protection.
- Japanese pachinko stored-value card system.

COUNTERPANE

Characteristics of Attackers

- Motivation
  - Opportunity crime versus a targeted attack.
  - Financial motivation versus political motivation.
  - "Bored graduate student attack."
- Access
  - What kind of access does the attacker have?

COUNTERPANE

Characteristics of Attackers (cont.)

- Skill
  - Different attacks require different skills.
  - Some attacks require multiple skills.
- Risk aversion
  - Is the attacker willing to risk publicity?
  - Is the attacker willing to risk jail time?
  - Is the attacker willing to risk death?

Characteristics of Attackers (cont.)

- Funding
  - Money can purchase many of the other characteristics. For example, money can purchase skill or access.

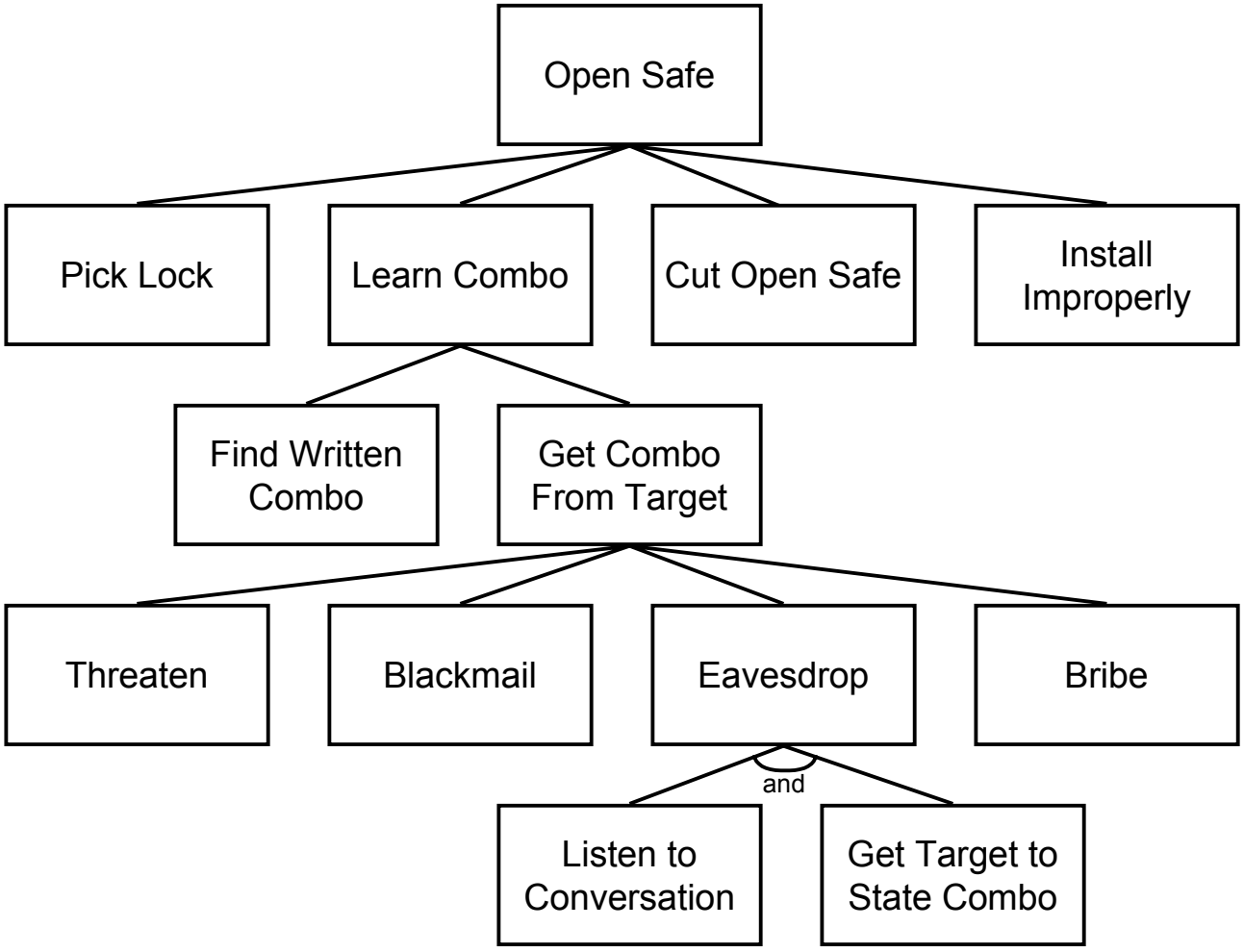
Attack Trees: What Are They?

- A way of thinking and describing security of systems and subsystems.
- A way of building an automatic database that describes the security of a system.
- A way of capturing expertise, and reusing it.
- A way of making decisions about how to improve security, or the effects of a new attack on security.

Attack Trees: How Do They Work?

- Represent the attacks and countermeasures as a tree structure.
- Root node is the goal of the attack.
  - In any complex system, there are several root nodes, each representing a different goal.
- Leaf nodes are attacks.

Basic Attack Tree



COUNTERPANE

"And" Nodes and "Or" Nodes

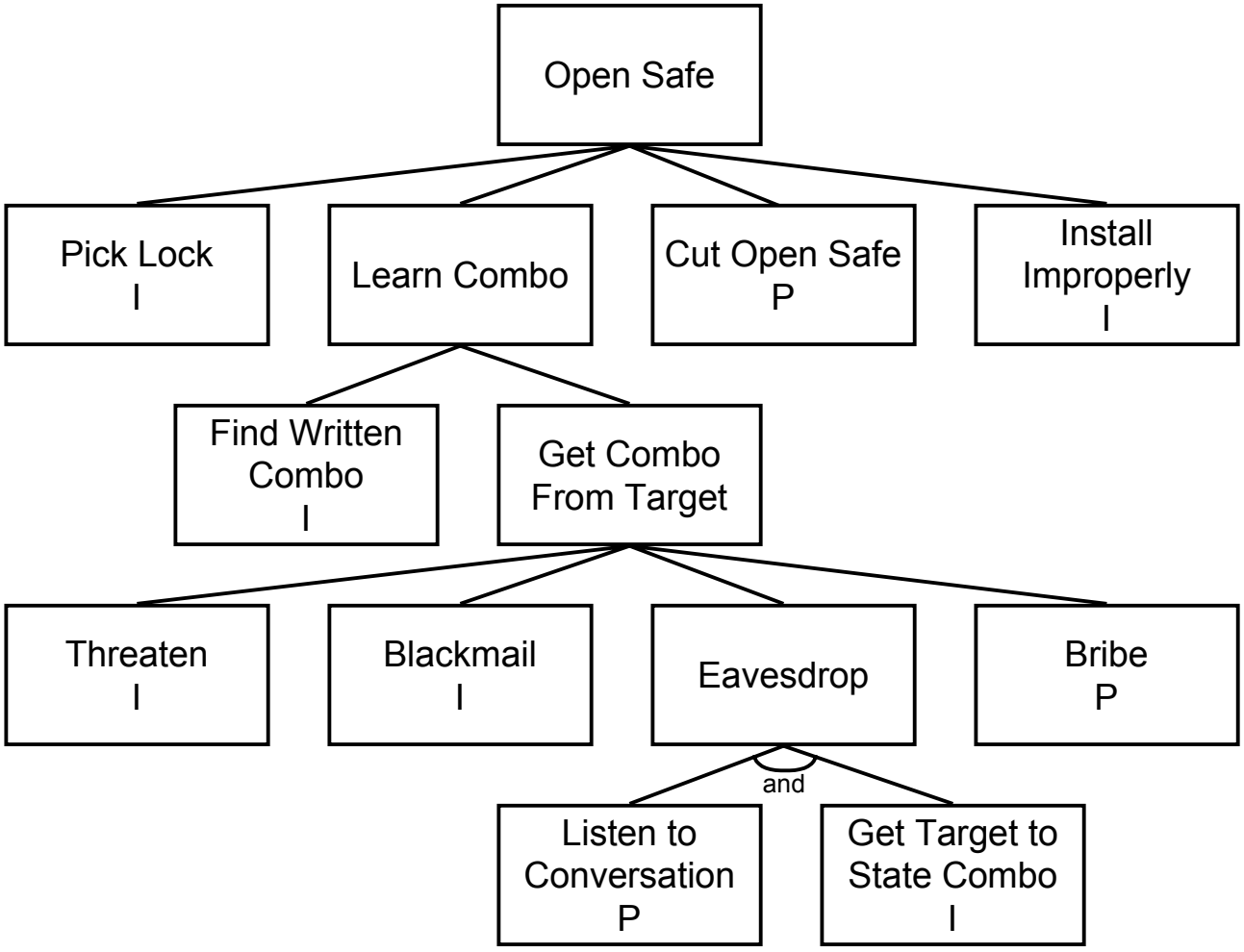
- "Or" nodes represent different ways to achieving the same goal.
  - For example, to break into a house you can either pick the door lock OR break a window.
- "And" nodes represent different steps in achieving a goal.
  - For example, to enter through a window you need to break the window AND climb through the opening.

COUNTERPANE

Boolean Node Values

- Once a tree is created, different values can be assigned to the leaf nodes.
- The simplest of these values are boolean: possible vs. impossible, for example.

Possible and Impossible Nodes



P = Possible
I = Impossible

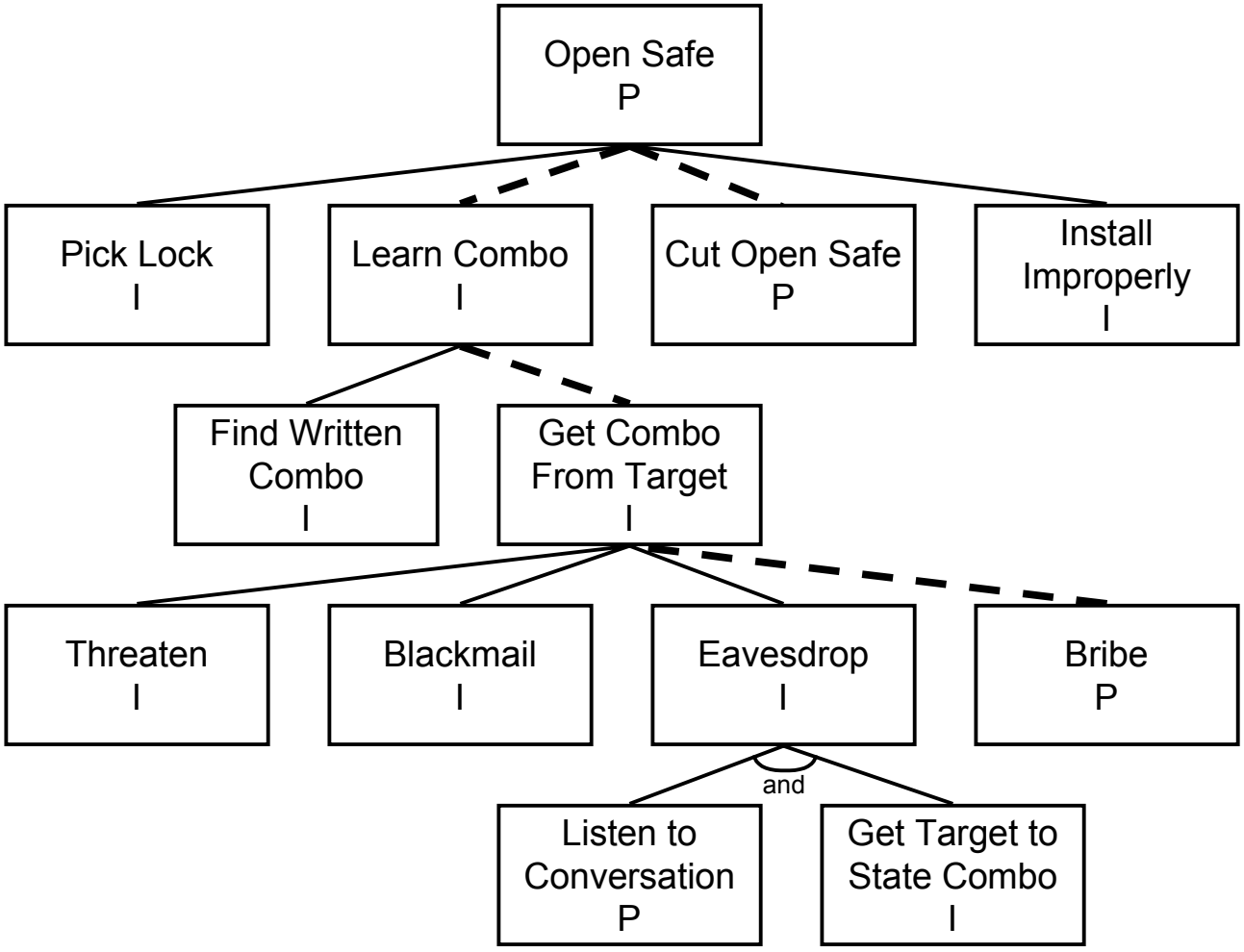
COUNTERPANE

Propagating Node Values Up the Tree

- A node's value is a function of its children's.
- Different calculation rules for AND nodes and OR nodes.
- Start at the leaf nodes and calculate up to the root.

COUNTERPANE

Possible Attacks



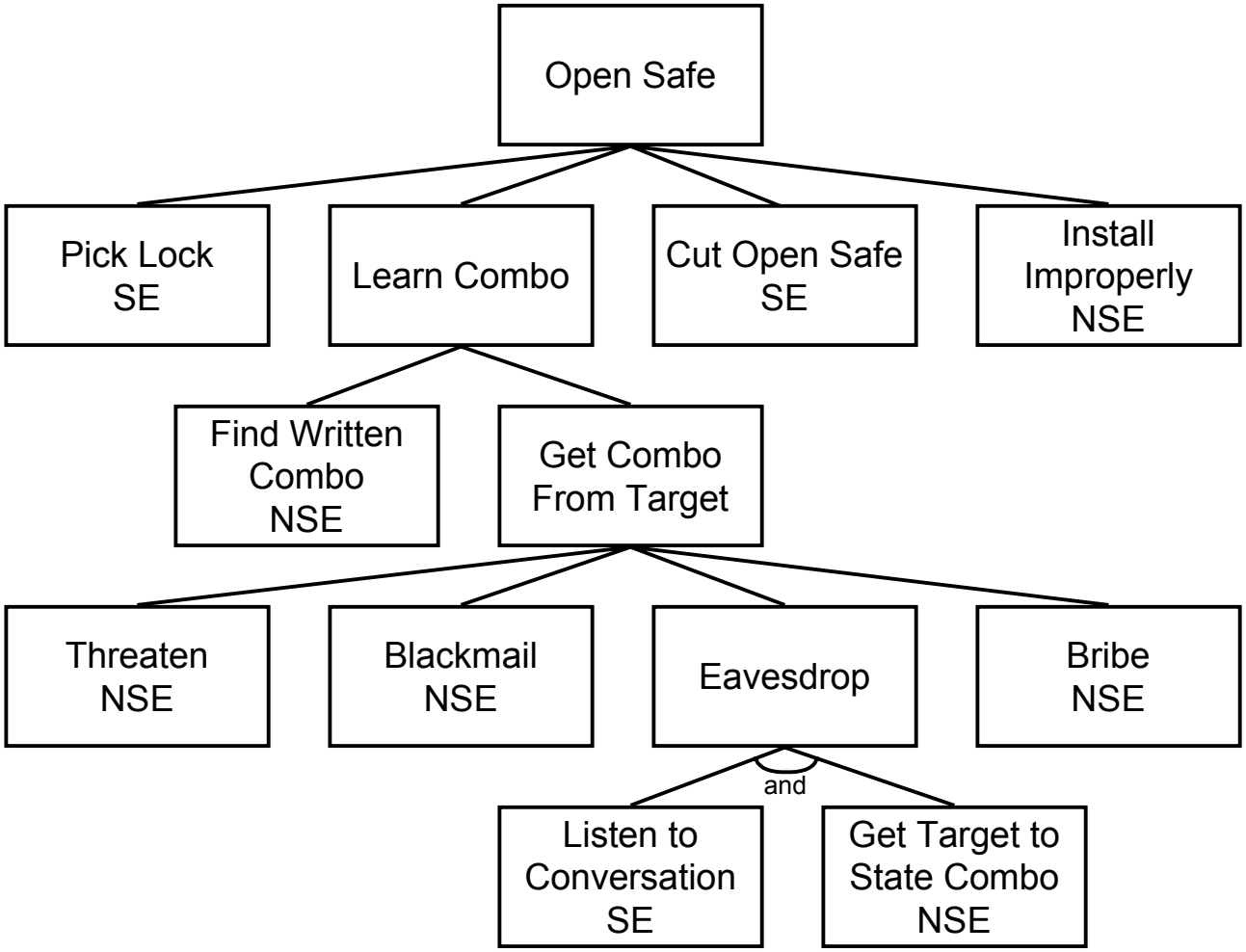
P = Possible
I = Impossible

COUNTERPANE

Other Boolean Node Values

- Any Boolean value can be codified in the leaf nodes and then used to prune the tree.
  - Easy and not easy.
  - Expensive and not expensive.
  - Intrusive and non-intrusive.
  - Legal and illegal.
  - Special equipment required and not required.

Special Equipment Required



NSE = No special equipment

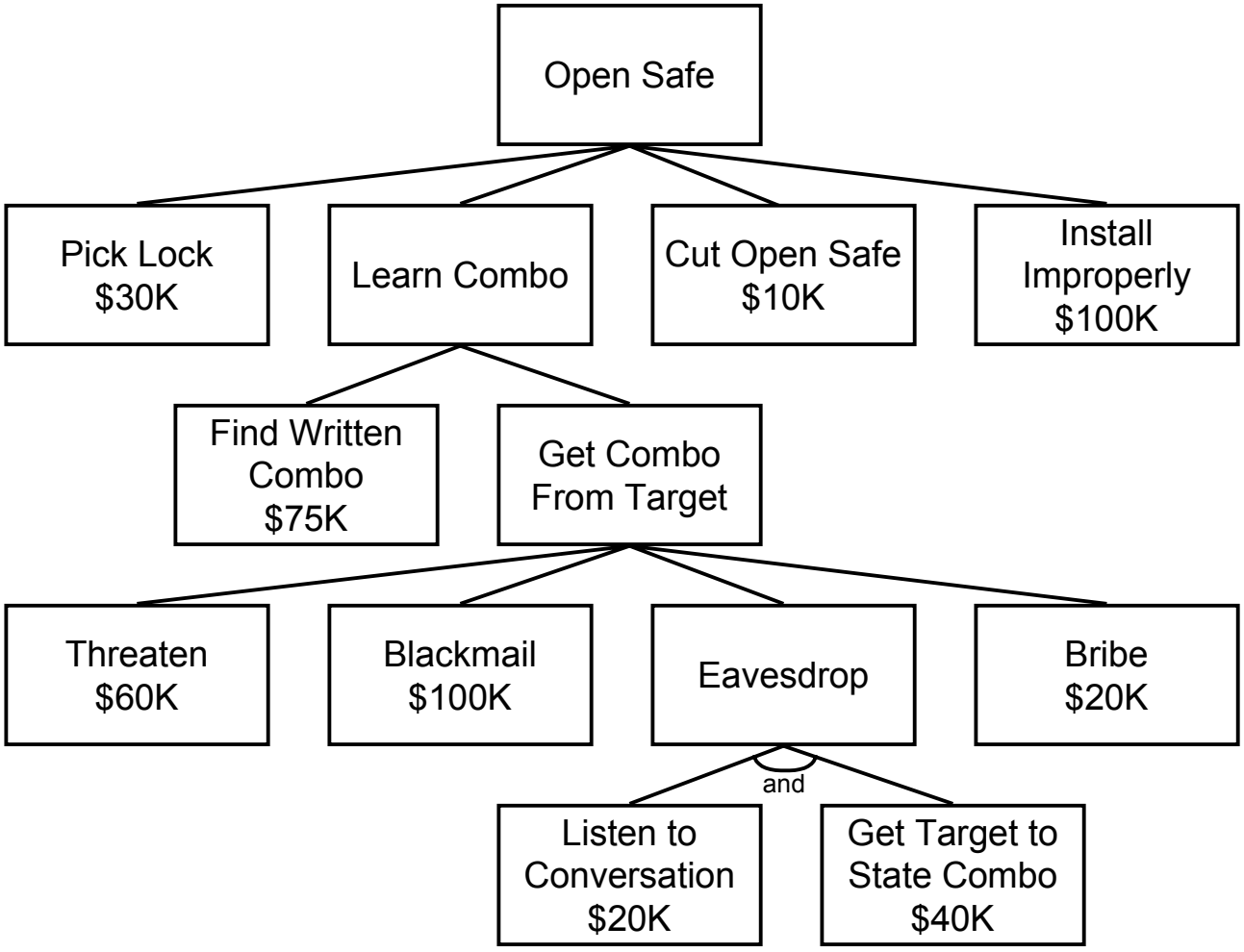
SE = Special equipment required

COUNTERPANE

Continuous Node Values

- Continuous node values can also be codified into the leaf nodes.
  - Cost in dollars to attack / defend.
  - Time to achieve / repulse.
  - Cost in resources to attack / defend.

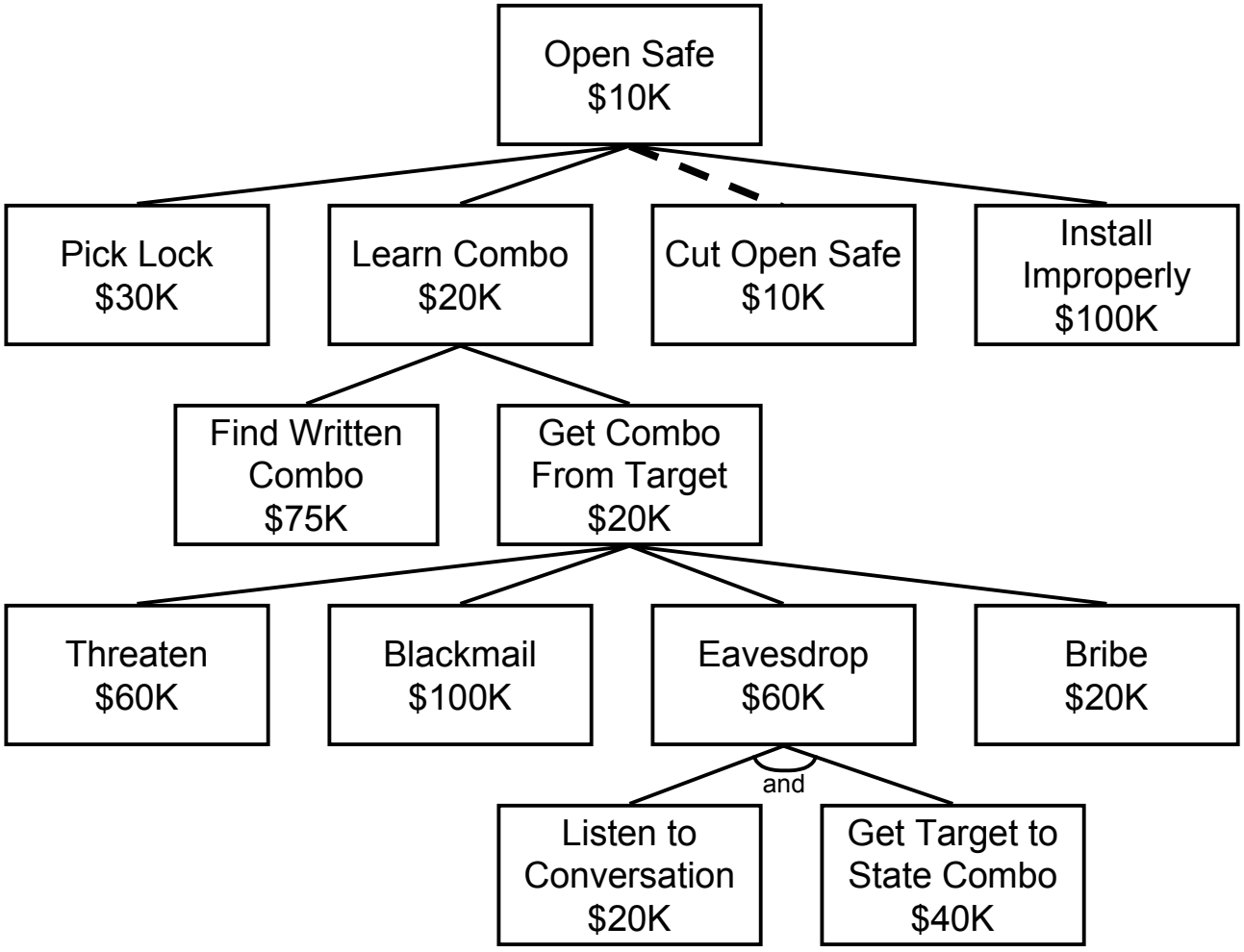
Cost of Attack



$ = Cost of attack

COUNTERPANE

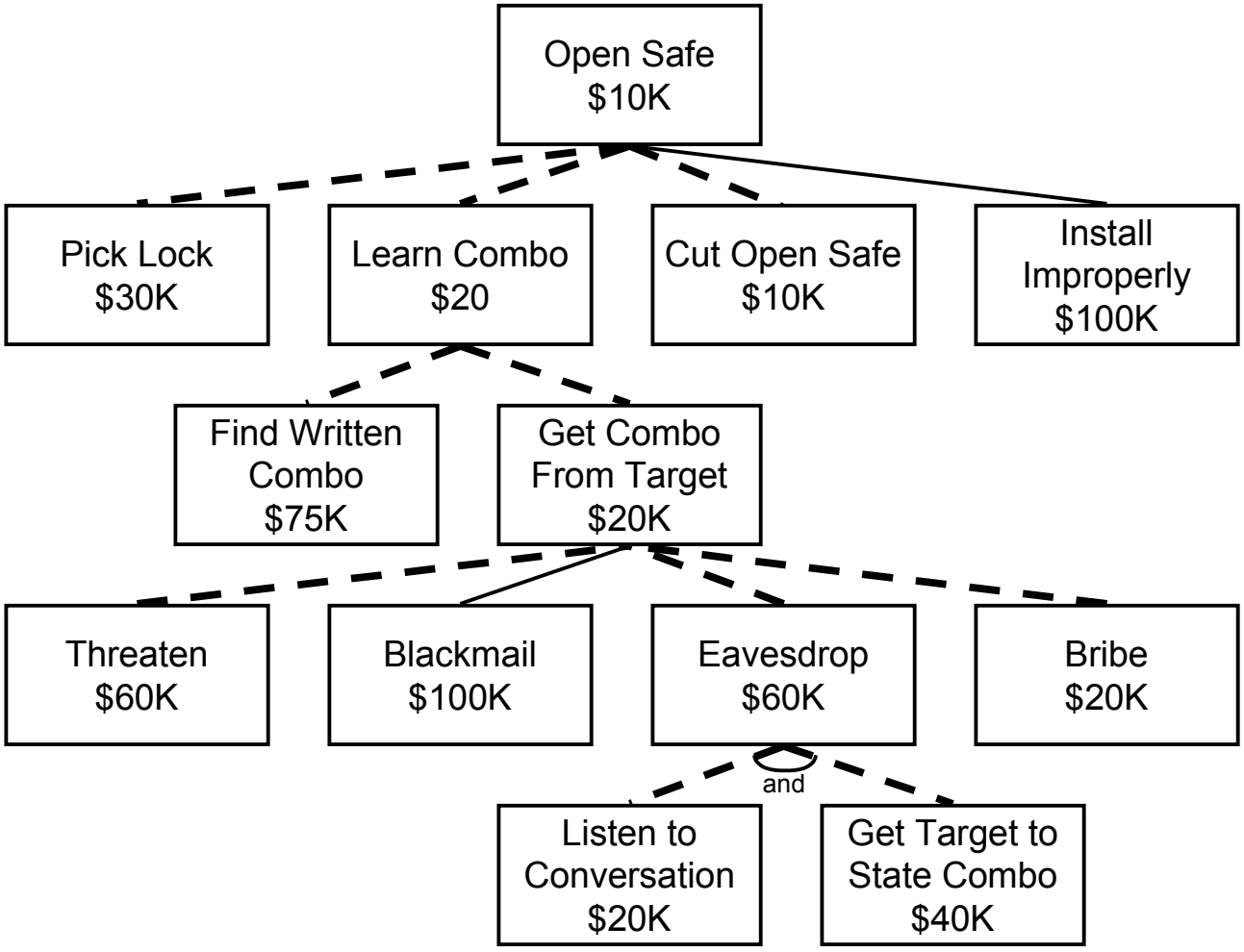
Cheapest Attack



$ = Cost of attack

COUNTERPANE

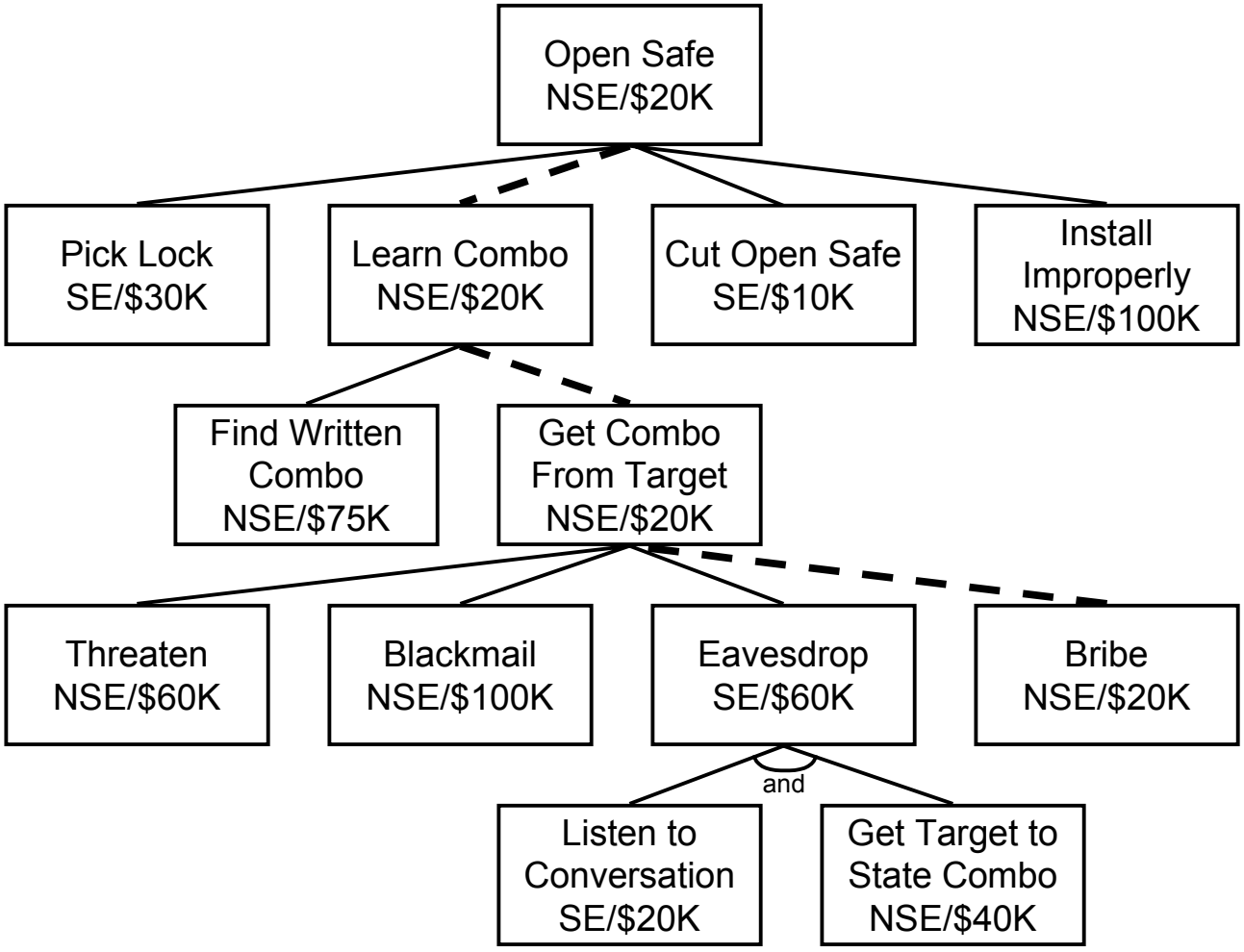
All Attacks Less than $100K



$ = Cost of attack

COUNTERPANE

Cheapest Attack Requiring No Special Equipment



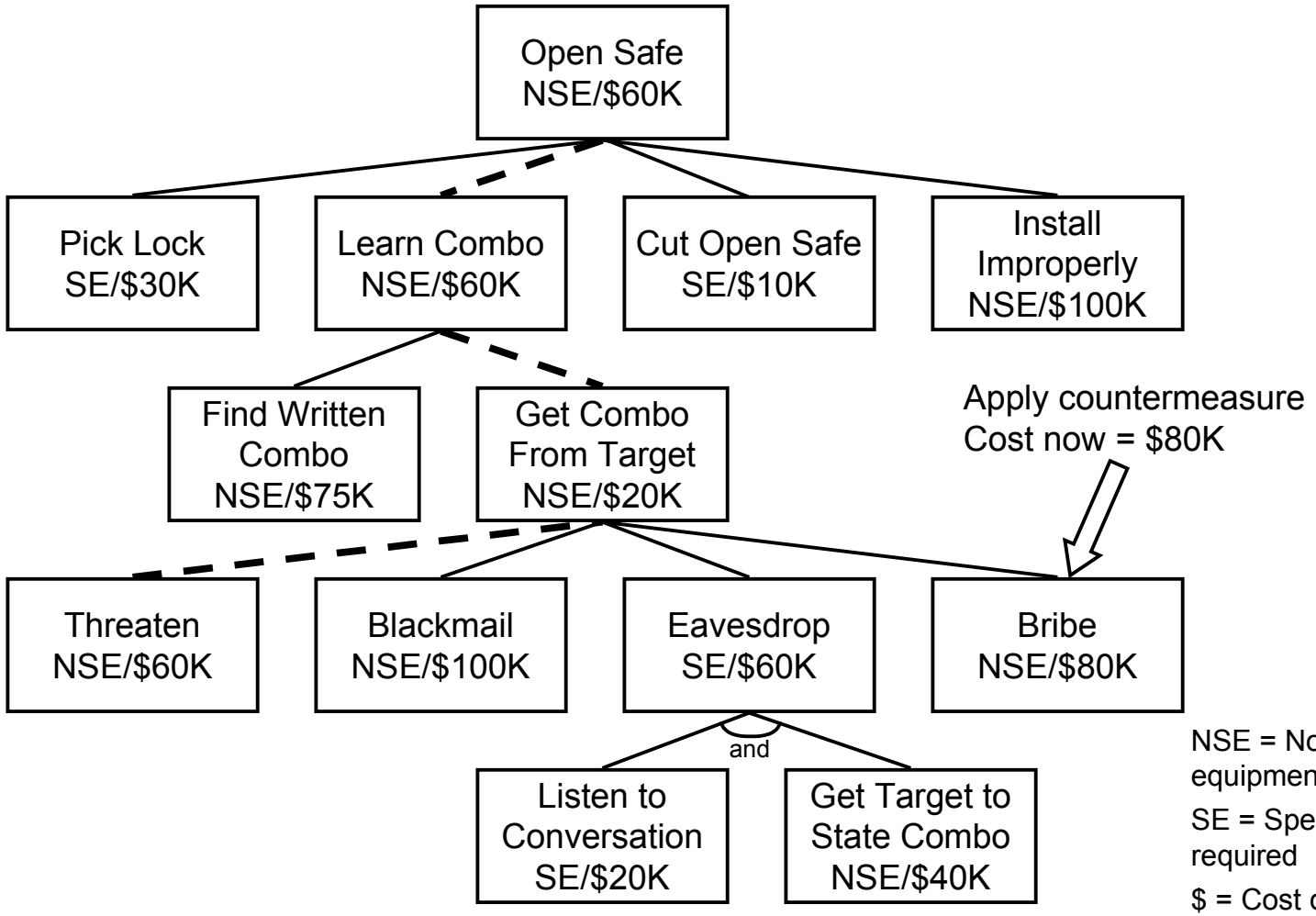
NSE = No special equipment

SE = Special equipment required

$ = Cost of attack

COUNTERPANE

Applying a Countermeasure– Cheapest NSE Attack Now $60K



NSE = No special equipment

SE = Special equipment required

$ = Cost of attack

COUNTERPANE

Other Continuous Node Values

- Probability of success of a given attack.
- Likelihood that an attacker will try a given attack.

COUNTERPANE

Combining Node Values

- Each node can have several values: Boolean and continuous.
- Can be used to make statements about attacks.
- For example:
  - Cheapest low-risk attack
  - Most likely non-intrusive attack
  - Best low-skilled attack
  - Cheapest attack with the highest probability of success

Tree Construction

- Step 0) Identify goals. Each goal is a separate attack tree.
- Step 1) Identify attack against goals; repeat as necessary.
- Step 2) Existing attack trees can be plugged in as appropriate.
- In general, once you have a library of general attack trees, you can create a specific tree out of these reusable components after the first couple of levels.

Attack Tree Against a Payment System

Goal: System-wide Attack on the Payment System (OR)

1. Replay valid authorizations. (OR)

1.1. Replay whole valid transactions. (AND)

1.1.1. Get replay to work in real time. (OR)

1.1.1.1. Force message key equality. (OR)

1.1.1.1.1. Chosen tag challenge attack. (AND)

1.1.1.1.1.1. Reprogram or emulate TAGs. (* 2.1.2.1)

1.1.1.1.1.2. Choose tag challenge to force desired message key. (OR)

1.1.1.1.1.2.1. Hack app. RNG values in high 16 bits. (OR)

1.1.1.1.1.2.1.1. Find app. boxes with flawed PRNGs. Countermeasure: Design PRNG properly. Cost to defeat: None.

1.1.1.1.1.2.1.2. Actively attack app. box PRNG. Countermeasure: Design PRNG properly. Cost to defeat: Moderate.

COUNTERPANE

Attack Tree Against a Payment System (cont.)

1.1.1.1.1.2.2. Wait out right 16 bits. Cost to defeat: Low, but not practical.

1.1.1.2. Find collision-MACs for diff. challenges. Countermeasure: Design MAC and protocol properly. Cost to Defeat: Unknown, est. medium-high.

1.1.2. Keep TAG off hotlist. (OR)

1.1.2.1. Subvert site long-term (keeper of hotlist). Cost: High; Risk: High.

1.1.2.2. Alter hotlist on-site at application box. Countermeasure: Don't allow tampering with application box's memory. Cost to defeat: Moderate.

1.1.2.3. Alter hotlist in transit. Countermeasure: Digitally sign hotlist. Cost to Defeat: Very high.

1.1.2.4. Alter hotlist on-site at center. Countermeasure: Physical and policy security at system. Cost to Defeat: Moderate; Risk: High.

1.1.2.5. Hack applications to bypass hotlist check.

1.1.2.6. Alter messages back to center from application boxes in transit. Countermeasure: Digitally sign transmissions. Cost to defeat: Very high.

Attack Tree Against a Payment System (cont.)

2. Replay partial valid transactions. (AND)

2.1. Get partial replay to work in real time. (OR)

2.1.1. Replay recharges into tag.

2.1.2. Tag key-recovery attack. (AND)

2.1.2.1. Reprogram or emulate tags. (OR)

2.1.2.1.1. Learn to reprogram tags.

2.1.2.1.2. Learn to emulate tags. (AND)

2.1.2.1.2.1. Learn details of how tags work. (OR)

2.1.2.1.2.1.1. Bribe someone in the know. Cost: Moderate; Risk: High

2.1.2.1.2.1.3. Reverse-engineer tags. Cost: Medium-high; Risk: None.

2.1.2.1.2.2. Find a cost-effective substitute for the tags. Cost: Unknown, probably moderate.

2.1.2.2. Recover internal tag keys. (OR)

Attack Tree Against a Payment System (cont.)

2.1.2.2.1. Cryptanalytic attack on MAC and protocol. Cost: Unknown, est. medium-high.

2.1.2.2.2. Reverse-engineer tag. Cost: Medium.

2.1.3. Protocol patching attack. (AND)

2.1.3.1. Learn communications protocol. Cost: Low

2.1.3.2. Find a cost-effective way to patch new messages into the protocol. Cost: Unknown, est. medium-high.

2.1.3.3. Get the application box to accept the patched protocol message.

2.1.3.3.1. Find a way to patch the MAC of one message to be equal to that of another, without knowledge of the message key. Cost: Unknown, est. high.

2.1.3.3.2. Known key MAC collision attack. (AND)

2.1.3.3.2.1. Find a way to patch the MAC of one message to be the same as another. Cost: Unknown, est. high.

Attack Tree Against a Payment System (cont.)

2.1.3.3.2.2. Learn tag internal keys for all serial numbers used. (OR)
2.1.3.3.2.2.1. Attack 2.1.3.3.2.2.1. (AND)
2.1.3.3.2.2.1.1. Recover tag key from tag.
2.1.3.3.2.2.1.2. Learn key generation mechanism.
2.1.3.3.2.2.1.3. Only use one serial number.
2.1.3.3.2.2.2.2. Capture many serial numbers.
2.1.3.3.2.2.2.3. Use many serial numbers.
2.1.3.3.2.2.2.4. Learn key generation mechanism.
2.1.3.4. Avoid hotlist problems. (OR)
2.1.3.4.1. Keep the tag off hotlist. (* 1.1.2)
2.1.3.4.2. Patch the serial number, too.
2.2. Keep tag off hotlist. (* 1.1.2).

Attack Tree Against a Payment System (Explanation)

- Countermeasures are included in the nodes.
- An asterisk (*) means that the node was previously expanded elsewhere.
- Node 2.1.3.4.1 is the same as node 1.1.2. This node has six children, but instead of writing them in both places, they are only listed on the tree once, after node 1.1.2.
- Alternatively, the nodes 1.1.2.1, 1.1.2.2, 1.1.2.3, 1.1.2.4, 1.1.2.5, and 1.1.2.6 can be viewed as having multiple parent nodes: 1.1.2, 2.1.3.4.1, and 2.2.

Using an Attack Tree to Determine the Vulnerability of a System Against an Attack

- After building an attack tree, an analyst can look at the value of the root node to see if the system goal is vulnerable to attack.
- For example, the presence of a possible Boolean value or an attacker's cost below a certain threshold.
- The analyst can also determine if the system is vulnerable to a particular type of attack.
  - Password guessing attacks, legal attacks, unskilled attacks, etc.

Using an Attack Tree to List the Security Assumptions of a System

- The attack tree can also be used to provide a comprehensive list of the assumptions of a security system.
  - For example, the security of this system assumes that no one can successfully bribe the president of our corporation.

Using an Attack Tree to Determine the Impact of a System Modification

1) Calculate the value of the root node of an attack tree using the procedures defined previously.
2) Incorporate the new attack by adding new leaf nodes to the attack tree, or revise the values of existing leaf nodes.
3) Modify the logic nodes, if necessary.
4) Calculate the new value of the root node using these new and revised leaf and logic nodes.
5) Compare the value of the root node obtained in step (1) with the value obtained in step (4). The difference in values represents the impact of the modification.

Using an Attack Tree to Compare and Rank Attacks

- In a similar manner, different attack (or defense) scenarios can be compared and ranked (based on their respective root node values) in order of most likely to succeed.

COUNTERPANE

Attack Tree Against PGP

Goal: Read a message encrypted with PGP (OR)

1. Read a message encrypted with PGP

1.1. Decrypt the message itself (OR)

1.1.1. Break asymmetric encryption (OR)

1.1.1.1. Brute-force break asymmetric encryption (OR)

It is possible to encrypt all possible keys with the recipient's (known) public key, until a match is found. The effectiveness of this attack is greatly reduced by the random padding introduced in the encryption of the symmetric key.

Attack Tree Against PGP (cont.)

1.1.1.2. Mathematically break asymmetric encryption (OR)

1.1.1.2.1 Break RSA (OR)

It is not currently known whether breaking RSA is equivalent to factoring the modulus.

1.1.1.2.2 Factor RSA modulus/calculate ElGamal discrete log

Either of these would require solving number theoretic problems currently conjectured to be very difficult.

1.1.1.3. Cryptanalyze asymmetric encryption

1.1.1.3.1. General cryptanalysis of RSA/ElGamal (OR)

No techniques are currently known for general cryptanalysis of RSA or ElGamal. Cryptanalysis of one ciphertext would imply a general method to break RSA/ElGamal.

Attack Tree Against PGP (cont.)

1.1.1.3.2. Exploiting weaknesses in RSA/ElGamal (OR)

There are a few weaknesses known to exist in RSA; however, PGP implementation has mostly eliminated these threats.

1.1.1.3.3. Timing attacks on RSA/ElGamal

Timing attacks have been reported on RSA; they should also be feasible on ElGamal. Such an attack, however, requires low-level monitoring of the recipient's computer while they are decrypting the message.

Attack Tree Against PGP (cont.)

1.1.2. Break symmetric-key encryption

1.1.2.1. Brute-force break symmetric-key encryption (OR)

All symmetric-key algorithms supported for use by PGP have key sizes of at least 128 bits. This is currently infeasible for brute-force searching.

Brute-force searching is made somewhat easier by the redundancy included at the beginning of all encrypted messages. See the OpenPGP RFC.

1.1.2.2. Cryptanalysis of symmetric-key encryption

The symmetric-key algorithms supported by PGP 5.x are: IDEA, 3-DES, CAST-5, Blowfish, and SAFER-SK128. No efficient methods are currently known for general cryptanalysis of these algorithms.

Attack Tree Against PGP (cont.)

1.2. Determine symmetric key used to encrypt the message via other means

1.2.1. Fool sender into encrypting message using public key whose private key is known (OR)

1.2.1.1. Convince sender that a fake key (with known private key) is the key of the intended recipient

1.2.1.2. Convince sender to encrypt using more than one key—the real key of the recipient, one a key whose private key is known

1.2.1.3. Have the message encrypted with a different public key in the background, unknown to the sender

This could be done by running a program which fools the user into believing that the correct key is being used, while actually encrypting with a different key.

Attack Tree Against PGP (cont.)

1.2.2. Have the recipient sign the encrypted symmetric key (OR)

If the recipient blindly signs the encrypted key, he unwittingly reveals the unencrypted key. The key is short enough so that hashing should not be necessary before signing. Or, if a message can be found which hashes to the value of the encrypted key, the recipient can be asked to sign the (hash of the) message.

1.2.3. Monitor sender's computer memory. (OR)

1.2.4. Monitor receiver's computer memory. (OR)

The (unencrypted) symmetric key must be stored somewhere in memory at some point during the encryption and decryption. If memory can be accessed, this gives a way to capture the key and get at the message.

Attack Tree Against PGP (cont.)

1.2.5. Determine key from pseudo-random number generator (OR)

1.2.5.1. Determine state of randseed.bin when message was encrypted (OR)

1.2.5.2. Implant software (virus) which deterministically alters the state of randseed.bin (OR)

1.2.5.3. Implant software which directly affects the choice of symmetric key

1.2.6. Implant virus which exposes the symmetric key

Attack Tree Against PGP (cont.)

1.3. Get recipient to (help) decrypt message (OR) 1.3.1. Chosen ciphertext attack on symmetric key (OR)

The CFB mode used by PGP is completely insecure under a chosen ciphertext attack. By sending the (encryption of the) same key to the recipient, along with a modified body of the message, the entire contents of the message can be obtained.

Attack Tree Against PGP (cont.)

1.3.2. Chosen ciphertext attack on public key (OR)

Since RSA and ElGamal are malleable, known changes can be made to the symmetric key which is encrypted. This modified (encrypted) key can then be sent along with the original message. This opens up the possibility of related-key attacks on the symmetric algorithms. Or, a weak ciphertext can be found whose decryption under the symmetric key algorithm reveals information about the modified key, which then leads directly to information about the original key.

1.3.3. Send the original message to the recipient (OR)

If the recipient decrypts and replies to this message automatically, the plaintext message is immediately revealed.

Attack Tree Against PGP (cont.)

1.3.4. Monitor outgoing mail of recipient (OR)

If the receiver replies to the original message in a non-encrypted manner, information about the original message may be gleaned

1.3.5. Spoof reply-to: or from: field of original message (OR)

In this case, the receiver may reply directly to the forged email address, and even if the reply is encrypted, it will be with a key whose private key is known.

1.3.6. Read message after it has been decrypted by recipient

1.3.6.1. Copy message off user's hard drive or virtual memory (OR)

Attack Tree Against PGP (cont.)

1.3.6.2. Copy message off backup tapes (OR)

1.3.6.3. Monitor network traffic (OR)

1.3.6.4. Use electromagnetic snooping techniques to read message as it is displayed on the screen (OR)

1.3.6.5. Recover message from printout

This may work if the recipient prints out a copy of the message. In that case, the line to the printer can be tapped, the printer's memory can be examined, or the hardcopy itself may be left in an insecure area.

Attack Tree Against PGP (cont.)

1.4. Obtain private key of recipient

1.4.1. Factor RSA modulus/calculate ElGamal discrete log (OR)

Either of these would require solving number theoretic problems currently conjectured to be very difficult.

1.4.2. Get private key from recipient's key ring (OR)

1.4.2.1. Obtain encrypted private key ring (AND)

1.4.2.1.1. Copy it from user's hard drive (OR)

1.4.2.1.2. Copy it from disk backups (OR)

1.4.2.1.3. Monitor network traffic (OR)

Attack Tree Against PGP (cont.)

1.4.2.1.4. Implant virus/worm to expose copy of the encrypted private key

Given the recent Melissa virus incident, something like this is very feasible. Other options include making the file publicly readable, or posting it to the web.

1.4.2.2. Decrypt private key

1.4.2.2.1. Break IDEA encryption (OR)

1.4.2.2.2.1.1. Brute-force break IDEA (OR)

IDEA uses 128-bit keys. This is currently infeasible for brute-force searching.

1.4.2.2.2.1.2. Cryptanalysis of IDEA

No efficient methods are currently known for general cryptanalysis of IDEA.

Attack Tree Against PGP (cont.)

1.4.2.2.2. Learn passphrase

1.4.2.2.2.1. Monitor keyboard when user types passphrase (OR)

1.4.2.2.2.2. Convince user to reveal passphrase (OR)

1.4.2.2.2.3. Use keyboard-logging software to record passphrase when typed by user (OR)

1.4.2.2.2.4. Guess passphrase

1.4.3. Monitor recipient's memory (OR)

The private key must be stored somewhere in memory when the user decrypts any messages sent to him

COUNTERPANE

Attack Tree Against PGP (cont.)

1.4.4. Implant virus to expose private key

Really a more sophisticated version of 1.4.2.1.4. in which the virus waits to the private key to be decrypted before exposing it.

1.4.5. Generate insecure public/private key pair for recipient

If software can be modified (Trojan horse) or corrupted (virus), it can be used to have PGP generate an insecure public/private key pair (e.g. with a modulus whose factorization is known to the attacker)

General Attack Tree Against a Computer System

Goal: Read a specific message that has been sent from one Windows 95 computer to another.

1. Convince sender to reveal message. (OR)

1.1. Bribe user.

1.2. Blackmail user.

1.3. Threaten user.

1.4. Fool user.

2. Read message when it is being entered into the computer. (OR)

2.1. Monitor electromagnetic emanations from computer screen. (Countermeasure: use a TEMPEST computer.)

2.2. Visually monitor computer screen.

General Attack Tree Against a Computer System

3. Read message when it is being stored on sender's disk. (Countermeasure: use SFS to encrypt hard drive.) (AND)

3.1. Get access to hard drive. (Countermeasure: Put physical locks on all doors and windows.)

3.2. Read a file protected with SFS.

4. Read message when it is being sent from sender to recipient. (Countermeasure: use PGP.) (AND)

4.1. Intercept message in transit. (Countermeasure: Use transport-layer encryption program.)

4.2. Read message encrypted with PGP.

General Attack Tree Against a Computer System (cont)

5. Convince recipient to reveal message. (OR)

5.1. Bribe user.

5.2. Blackmail user.

5.3. Threaten user.

5.4. Fool user.

6. Read message when it is being read. (OR)

6.1. Monitor electromagnetic emanations from computer screen. (Countermeasure: use a TEMPEST computer.)

6.2. Visually monitor computer screen.

General Attack Tree Against a Computer System (cont)

7. Read message when it is being stored on receiver's disk. (OR)

7.1. Get stored message from user's hard drive after decryption. (Countermeasure: use SFS to encrypt hard drive.) (AND)

7.1.1. Get access to hard drive. (Countermeasure: Put physical locks on all doors and windows.)

7.1.2. Read a file protected with SFS.

7.2. Get stored message from backup tapes after decryption.

8. Get paper printout of message. (Countermeasure: store paper copies in safe.) (AND)

8.1. Get physical access to safe.

8.2. Open the safe.

What Else?

- Attack trees can show:
  - Intrusive vs. non-intrusive attacks.
  - Legal vs. illegal attacks.
  - Budget, skills, and/or access required of an attacker.
  - Probabilities of success for various attacks.
  - Likelihood of different attacks.
  - Value of different attacks.

What Else? (cont.)

- Attack trees can compare:
  - Effects of various countermeasures.
  - Security of different products.
- Attack trees can show:
  - What assumptions security is based on.
  - What happens when those assumptions are broken.
  - How to best use a security budget.

Scalability

- Attack trees become part of larger attack trees.
  - Attack tree against safe is part of a larger attack tree, whose goal is to read a document.
  - Attack tree against PGP is part of a larger attack tree, whose goal is to read a particular file.
- You can read the results of an attack tree without understanding its details.

Scalability (cont.)

- Changes at lower levels automatically propagate.
  - A new attack against PGP automatically affects the security of any tree that has PGP as a component.
  - A new attack against an encryption algorithm likewise propagates up.
- Subtrees are reusable components.
  - The PGP tree works everywhere PGP is used.

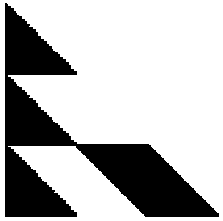
Conclusions

- In many systems, applying security measures is like sticking a tall spike in the ground and hoping that the enemy runs right into it.
- Attack trees are a methodology to ensure that security is a broad palisade.
- Attack trees are a rigorous way to think about security.
- Attack trees work.

YOU ARE ALL INVITED TO SUBSCRIBE TO MY FREE MONTHLY E-MAIL NEWSLETTER:

CRYPTO-GRAM

See http://www.counterpane.com for details.



COUNTERPANE