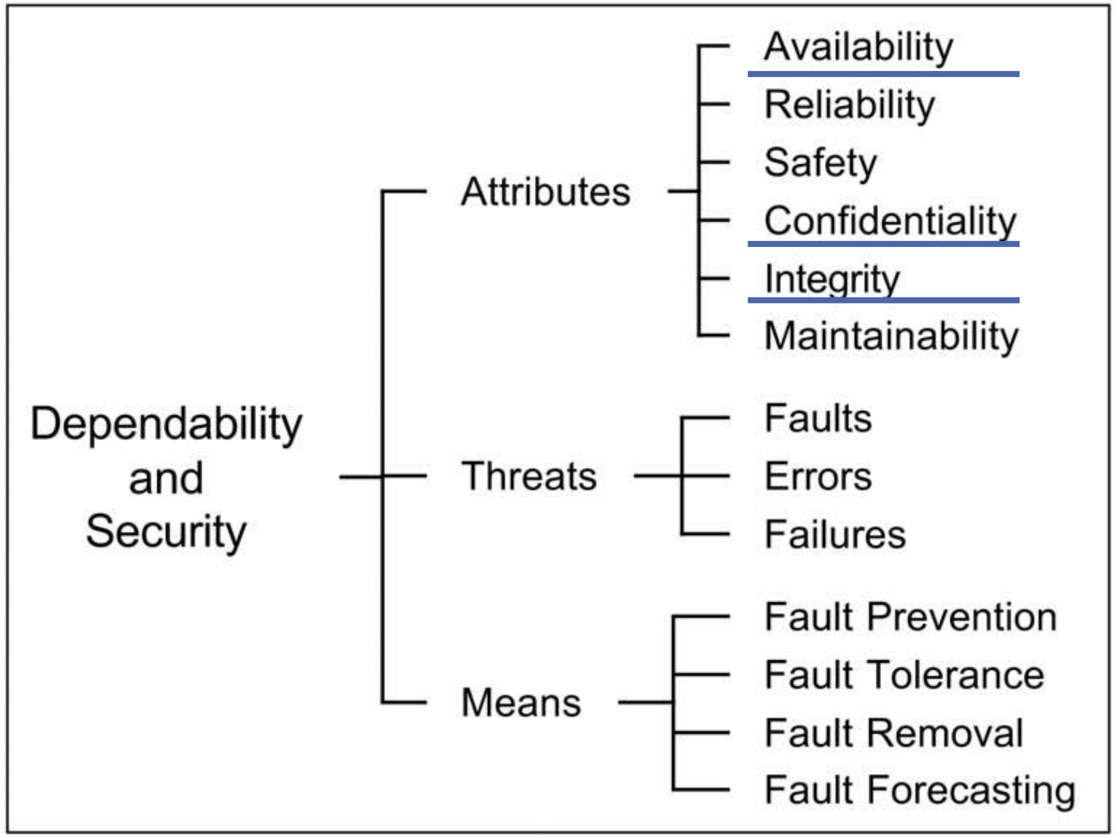
Security

Outline

- Security threats and vulnerability

- STRIDE Threat Modeling tool
- PLOVER : Preliminary List Of Vulnerability Examples for Researchers

- Quantitative evaluation of security
  - Attack trees
  - ADversary VIew Security Evaluation (ADVISE)


Dependability and Security
Attributes
Availability
Reliability
Safety
Confidentiality
Integrity
Maintainability
Threats
Faults
Errors
Failures
Means
Fault Prevention
Fault Tolerance
Fault Removal
Fault Forecasting

Security

The system generally implements an ***authorization policy***

- **Data confidentiality**
  protected data not read by unauthorized users

- **Data integrity**
  protected data not modified by unauthorized users

A violation in data confidentiality or data integrity does not imply a failure in reliability or availability. Both attributes are not related to the functionality of the system

Likewise,

- **Non-repudiability**
  “Prevents future false denial of involvement by either party in a transaction”
  “availability and integrity of the identity of the sender of the message or the receiver”
- **Authenticity**
  “The claimed identity of a party to a transaction can be independently verified”
  “ integrity of a message content, origin, time of emission, …”

Security policy

Security policy:
a set of security-motivated constraints that must be satisfied by an organization or a computer system.

e.g. policy regarding the public disclosure of company information, policy of the physical and networked access to company computers, constraints on how information may flow within a system

Methods for

- formally expressing and analysing security policies

- the enforcement of the policy

Security failures:

- Security failure of the system, or

- the security policy not adequately describes the security requirements
  (similar problem in system development life-cycle: requirements and specification)
  example: firewall/access control

Security

Fault taxonomy: attacks identified as *malicious faults*
Malicious faults can be executed with success only if there is a vulnerability in the system

Dependability in the face of system’s vulnerability and attacks

Attackers learn over time
Attackers build a strategy over time

Security

Coupling of vulnerability and security exploitation makes security failures different from traditional failures

- **Vulnerability:**

a computer system vulnerability is a flaw or weakness in a system or network that could be exploited to cause damage, or allow an attacker to manipulate the system in some way

Causes of vulnerabilities may be system components or basic flaws in an individual program or interactions of software programs, ….

- **Exploit**

Exploiting is the means through which a vulnerability can be leveraged for malicious activity (piece of software; sequence of commands, open source exploit kits, ….)

Security

One of the most important steps in preventing a security breach is *identifying security vulnerabilities before an attacker can leverage them*

Define what needs to be protected. Set the goal for the overall system security. Have an accurate list of the assets of the system: the network, the operating systems, the software, the environment ….

Having this list helps for example to identify
(i) security vulnerabilities from obsolete software and
(ii) known program bugs in specific operating systems and software.

Continuous monitoring of new and emerging threats and attack strategies.

Examples of source of vulnerability

Broken authentication

Authenticated credential or session identifier are compromised

Stolen credential or session id can be used by malicious agents to pose as original user.

Example:

a user executes login during a not authenticated session and then did not logout.
An attacker can get individual session ID that, for example, appear in an URL website and use the user account access

Admin Account Privileges

There are no strict control on user account access privileges.

Avoid that un-privileged user have admin-level accounts.
These users could create more privileged account

Examples of source of vulnerability

Countermeasure:

- limit the “access privileges” of software users. The less information/resources a user can access, the less damage that user account can do if compromised. Verifying that user account access is restricted to only what each user needs to do their job. *Policy of Least Privilege.*

- to make harder to attackers to hijack user accounts, instead of using username and password, apply multiple authentication methods (such as biometrics, one-use texted codes, and physical tokens) for giving users access to the system. *Multifactor Authentication (MFA).*

Examples of source of vulnerability

Hidden backdoor programs

A backdoor is a program (code) that allows a computer to be remotely accessed.

Such code is often installed for diagnostic, configuration, or technical support purposes by manufactures or developers. Hidden backdoor: in case the backdoor is installed without the user's knowledge. This is a software vulnerability because someone with knowledge of the backdoor can access the computer system and the networks connected to the computer.

Example: software vulnerability in the Huawei routers

From the web: https://www.bloomberg.com/news/articles/2019-04-30/vodafone-found-hidden-backdoors-in-huawei-equipment

"Europe's biggest phone company identified hidden backdoors in the software that could have given Huawei unauthorized access to the carrier's fixed-line network in Italy, a system that provides internet service to millions of homes and businesses… "

Examples of source of vulnerability

Software faults, e.g., buffer overflow.

Buffer overflow occurs when the input cannot be stored in the buffer and the memory area outside the buffer is overwritten. An attacker can overwrite areas that hold executable code, replacing it with their own code. A pointer can be overwritten for pointing to an exploit that when executed takes the control of the program.
(C, C++ are vulnerable programming languages)

Security bugs in interactive software

Issues in a single software can create security vulnerability. In case of more than one software that interface with other software the complexity increases and there may be more causes (such as unanticipated code interactions) to create vulnerability.

Examples of source of vulnerability

Misconfiguration (at any level of the application stack: network service, web server, databases, virtual machine, …).

Assume the application server's configuration allows detailed error messages, e.g. stack traces, to be returned to users. These messages can expose sensitive information or underlying flaws. For example, component versions that are known to be vulnerable.

Security Threats

Smart devices

Unknown devices represent possible vulnerabilities.

Network today is really complex and encompasses many “smart” devices, based on wireless communications. The issue with these devices is that they can be hijacked by attackers to form slaved networks of compromised devices to carry out further attacks.

Countermeasure

- have a clear figure of the architecture of the network and the total number of devices

- adds extra layers of protection between each of the individual assets on the network. If attackers bypass the outermost defenses of the network, there will still be other layers of protection between the compromised asset and the rest of the network. *Defense-in-depth approach* to network security.

Examples of source of vulnerability

Phishing Attacks

force employees to bypass security layers of the organization (Social Engineering).
The attacker attempts to trick the victim into disclosing sensitive data, disclosing account credentials, downloading malware.
Example: Link in an email (that mimics someone who has authority in the company) that will download malware in the computer, or request login and passwd.

Human factors

Many data breaches in an organization can be traced back to its employees. Employees may download the wrong file from an online site, or give the wrong person their user account credentials. Also employees may abuse their access privileges for personal gain.

Countermeasure

Provide basic knowledge needed to identify and avoid phishing attacks and accidental leakages ….
Cybersecurity awareness training

Security Threats

Malwares

Programs whose scope is to access sensitive data and copy it.

Some malwares can autonomously copy data and send it to a specific port or server that an attacker can then use to discreetly steal information.

Some malwares enter the system by using old security vulnerability not patched or they are downloaded by the web clicking on files, etc.

SQL injection

Gain access to the database via malicious code injection.

Untrusted data sent to the interpreter as part of a command or query. The interpreter executes unintended commands

e.g.: SELECT * FROM users WHERE username = 'administrator'--' AND password = ‘’

Security defences

Feature of running safe scripts

Users often wouldn’t know to disable this “feature.” For example, Safari uses the option to automatically run “trusted” or “safe” scripts.

Countermeasure

- Email Virus Detection Tools.
- Avoid to run scripts without malware/virus checks.
- Static analysis of the code.

Examples of security defences

Intrusion detection systems

An intrusion detection software monitors a network or a system for malicious activity or policy violations.

Intrusions or violations are reported. We distinguish between network intrusion detection systems and host-based intrusion detection systems. The hacker do not realised to be observed. Honepots can be used: a folder, for example, that contains information or resources of value to attackers. The software is capable of blocking or analyzing the attackers.

Penetration Testing (ethical hacking)

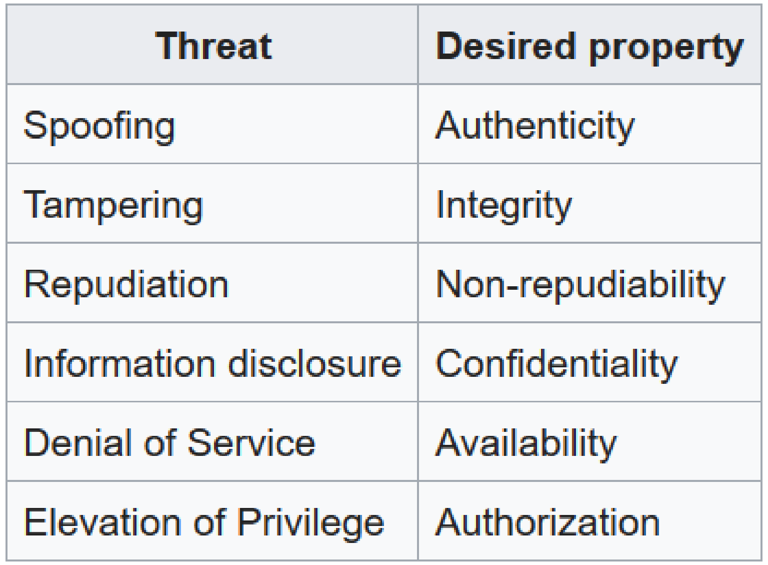
A penetration test (pen test) is an authorized simulated cyber attack on a computer system, performed to evaluate the security of the system.

The process typically identifies the target systems and a particular goal, then reviews available information and undertakes various means to attain that goal.

Microsoft Security Development Lifecycle (SDL) Threat Modeling tool

The STRIDE threat model provides a way to methodically review system designs and highlight security threats (*https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling*)

STRIDE uses six security threat categories to review system design (developed at Microsoft):



Threat	Desired property	
Spoofing	Authenticity	an adversary exploiting confusion about who is talking
Tampering	Integrity	an adversary modifying data
Repudiation	Non-repudiability	an adversary denying that something happened
Information disclosure	Confidentiality	disclosure of information to someone not authorized to see it
Denial of Service	Availability	deny or degrade service to users
Elevation of Privilege	Authorization	gain capabilities without proper authorization

Shostack (2014). Threat Modeling: Designing for Security. Wiley.

Microsoft Security Development Lifecycle (SDL) Threat Modeling tool

Taken from https://docs.microsoft.com/it-it/azure/security/develop/threat-modeling-tool-getting-started

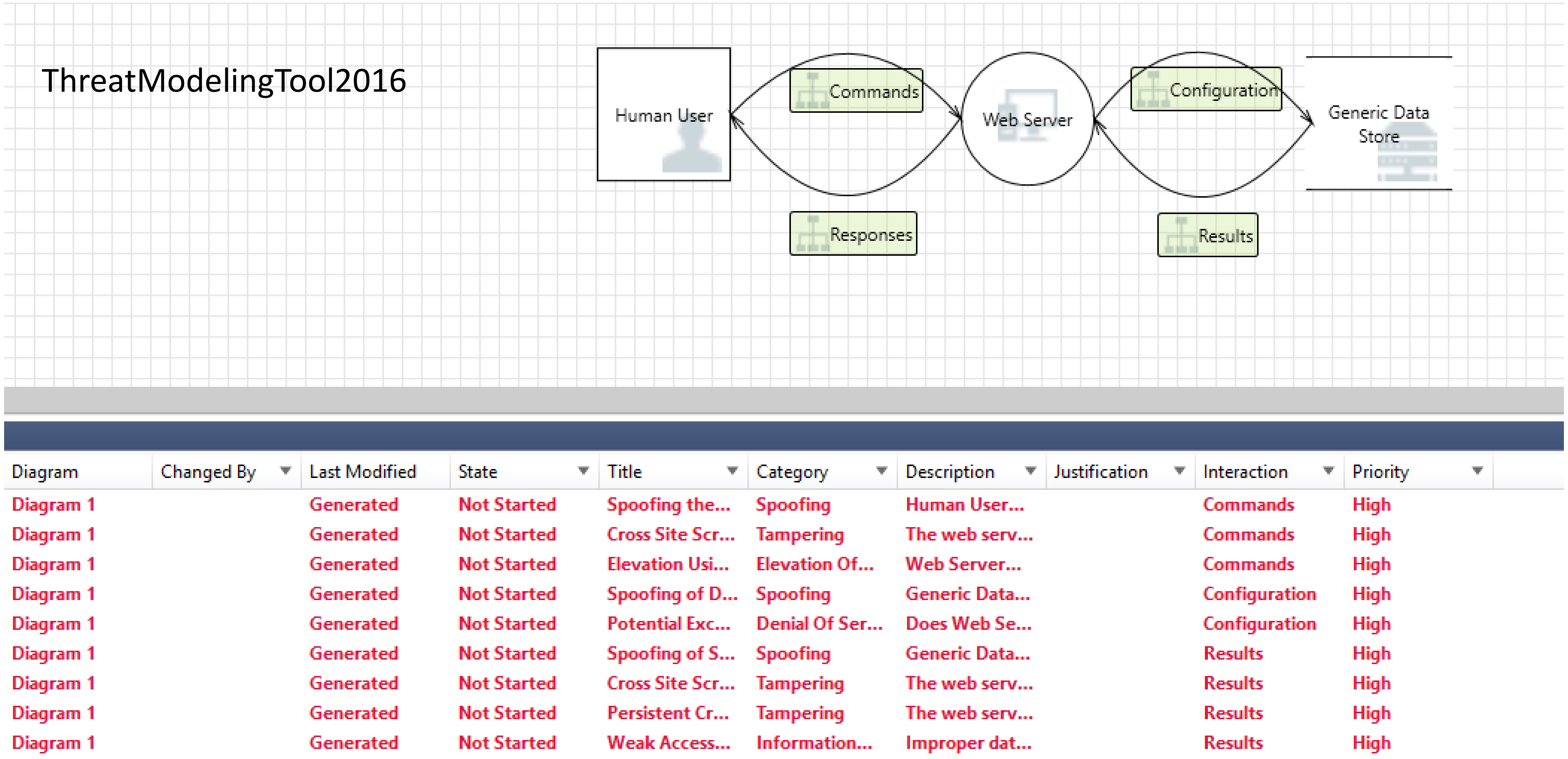


Diagram	Changed By	Last Modified	State	Title	Category	Description	Justification	Interaction	Priority
Diagram 1		Generated	Not Started	Spoofing the...	Spoofing	Human User...		Commands	High
Diagram 1		Generated	Not Started	Cross Site Scr...	Tampering	The web serv...		Commands	High
Diagram 1		Generated	Not Started	Elevation Usi...	Elevation Of...	Web Server...		Commands	High
Diagram 1		Generated	Not Started	Spoofing of D...	Spoofing	Generic Data...		Configuration	High
Diagram 1		Generated	Not Started	Potential Exc...	Denial Of Ser...	Does Web Se...		Configuration	High
Diagram 1		Generated	Not Started	Spoofing of S...	Spoofing	Generic Data...		Results	High
Diagram 1		Generated	Not Started	Cross Site Scr...	Tampering	The web serv...		Results	High
Diagram 1		Generated	Not Started	Persistent Cr...	Tampering	The web serv...		Results	High
Diagram 1		Generated	Not Started	Weak Access...	Information...	Improper dat...		Results	High

SDL report

"what can go wrong in this system we're working on?"

Vulnerability classification

Many works classify vulnerabilities ad threats that may appear in general in a computer system

Example

PLOVER : Preliminary List Of Vulnerability Examples for Researchers

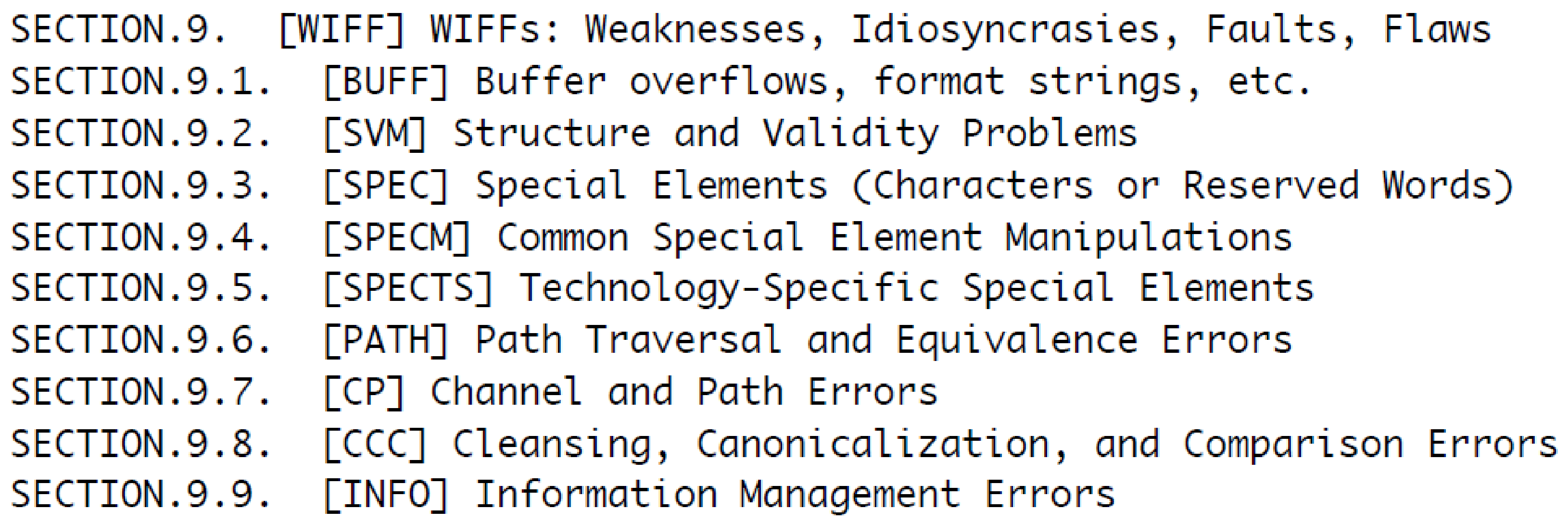
https://cwe.mitre.org/documents/sources/PLOVER.pdf

identifies 28 specific Weaknesses, Idiosyncrasies, Faults and Flaws (WIFFs)

Working document which lists over 1400 real examples of vulnerability (2006)

PLOVER : Preliminary List Of Vulnerability Examples for Researchers

Taken from https://cwe.mitre.org/documents/sources/PLOVER.pdf



SECTION.9. [WIFF] WIFFs: Weaknesses, Idiosyncrasies, Faults, Flaws
SECTION.9.1. [BUFF] Buffer overflows, format strings, etc.
SECTION.9.2. [SVM] Structure and Validity Problems
SECTION.9.3. [SPEC] Special Elements (Characters or Reserved Words)
SECTION.9.4. [SPECM] Common Special Element Manipulations
SECTION.9.5. [SPECTS] Technology-Specific Special Elements
SECTION.9.6. [PATH] Path Traversal and Equivalence Errors
SECTION.9.7. [CP] Channel and Path Errors
SECTION.9.8. [CCC] Cleansing, Canonicalization, and Comparison Errors
SECTION.9.9. [INFO] Information Management Errors

PLOVER : Preliminary List Of Vulnerability Examples for Researchers

Quantitative evaluation of Security

Combinatorial models

All basic events must be statistically independent

Do not model state - they model operational dependency of the system on the components

Reliability block diagrams: not used in security

Attack trees (similar to Fault Trees)

- Consider a security breach as a system failure
- An attack tree models all possible attacks against the system

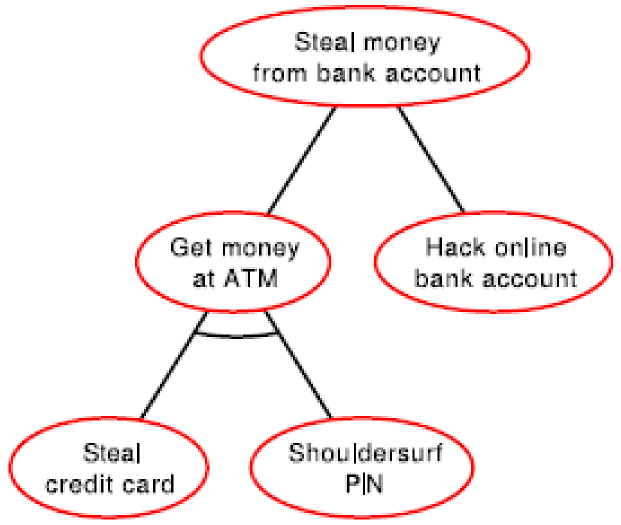
Attack Trees

The tree describes sets of events that can lead to the goal in a combinatorial way

Security of the system:

set of attack trees, where the root of each tree is the goal of a attacker that can damage the system operation

1. Root = goal of an attacker
2. Leaf nodes = different basic ways to achieve that goal (atomic attacks)
3. OR nodes = a node of which only one of its child nodes needs to be successful
4. AND nodes = a node of which all of its child nodes need to be successful



Attack tree published in [Buldas 20] Ahto Buldas et al. Attribute evaluation on attack trees with incomplete information, Computers & Security 88 (2020)

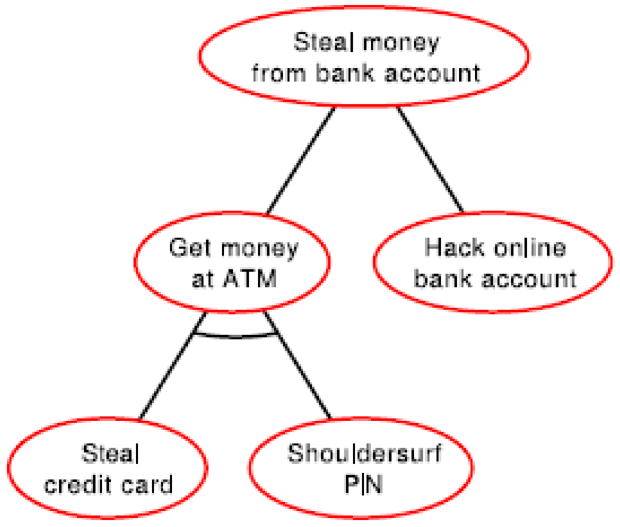
Attack Trees

Evaluation of different aspects of the system security, depending on the kind of values assigned to the leaf nodes

Since an atomic attack can have multiple values, the attack tree can be used to combine these values and help users to learn more about a system's vulnerability

Example

Possible/impossible, cost -> lowest possible cost attack



[50$, Eq=Si, …p=0.8]

[10$, Eq=No, …, p=0.2]

Example

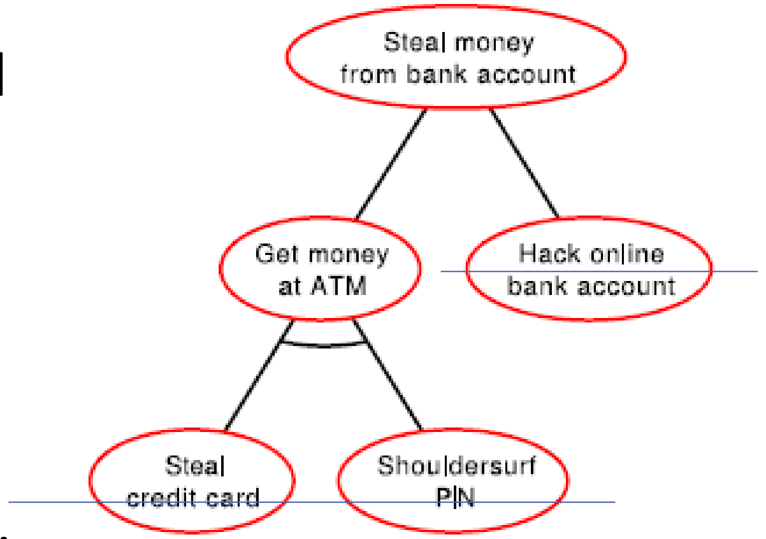
probability, special equipment value -> most probable attack with no special equipment required

assign values to leaf nodes and propagate the node value up to the root

Evaluation of Security

Minimum cut-set -> set of atomic attacks that achieve a goal

S = {{Steal credit card, Shouldersurf PIN}
{Hack online Bank acount}}

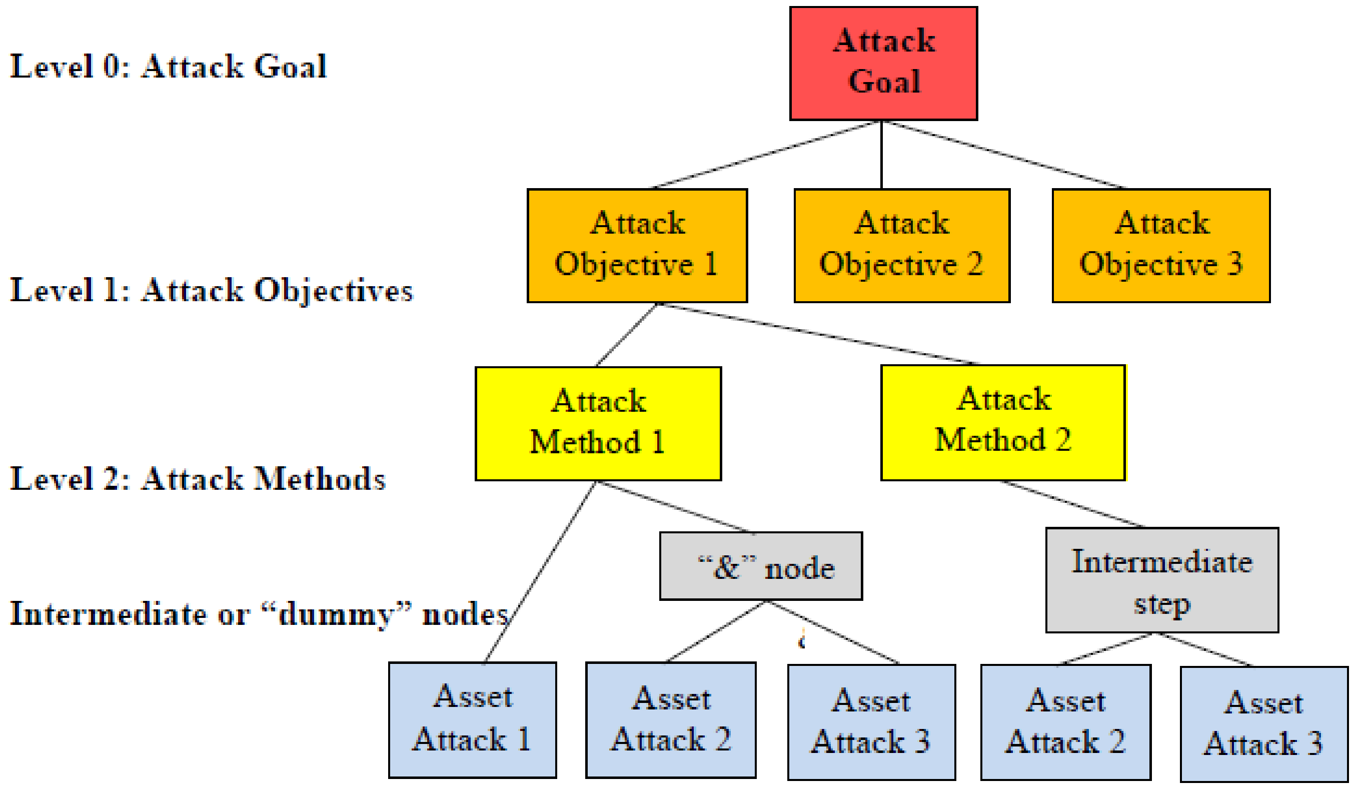


Impact of certain atomic attacks on the overall system security

Attack Trees: systematic ways to describe system vulnerability , making possible to assess risks and making security decisions

Attack trees: reusable as part of a larger attack tree for a system

EVITA: Attack tree structure

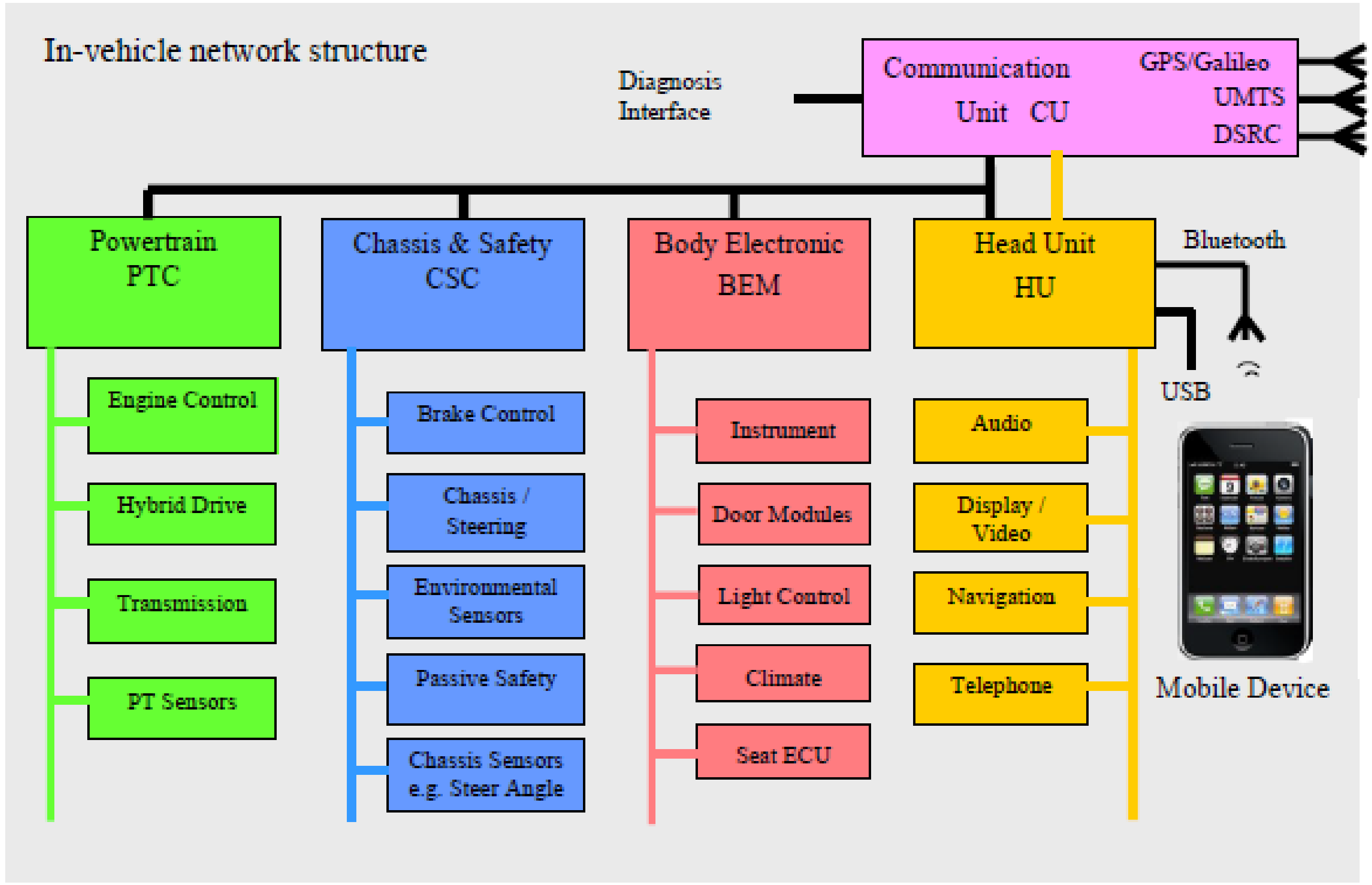


Each attack method will be based on a logical combination (AND/OR) of attacks against one or more "assets" populating the lowest levels of the attack tree.

Probability of success can be estimated for asset attacks

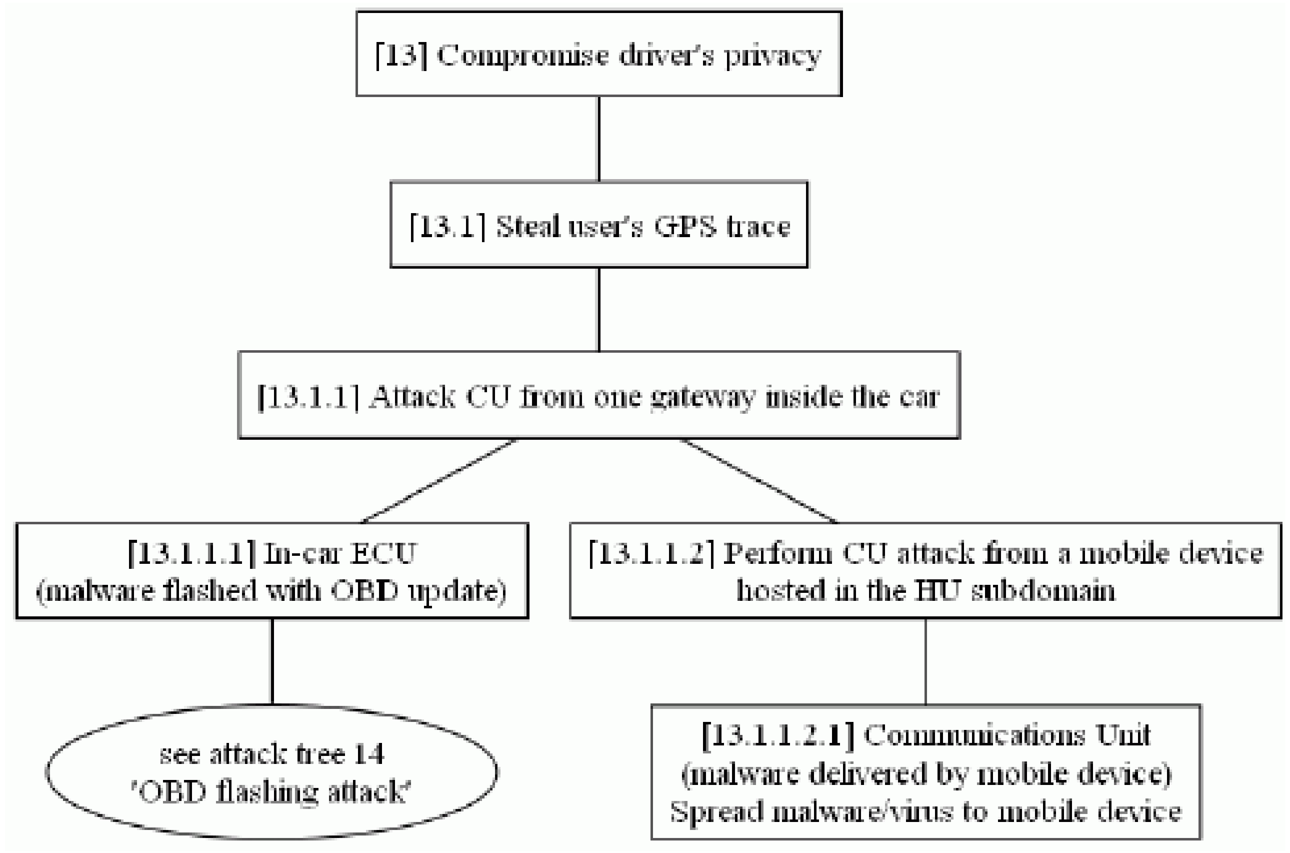
[EVITA-D2.3] E-safety vehicle intrusion protected applications (EVITA) project. Project reference: 224275 Programme: EU Seventh Reserch Framework Programme (2007–2013) Deliverable D2.3. *Security requirements for automotive on-board networks based on dark-side scenarios*.

Reference architecture



Taken from [EVITA-D2.3]

Attack tree : Compromise driver privacy


[13] Compromise driver's privacy
[13.1] Steal user's GPS trace
[13.1.1] Attack CU from one gateway inside the car
[13.1.1.1] In-car ECU (malware flashed with OBD update)
[13.1.1.2] Perform CU attack from a mobile device hosted in the HU subdomain
see attack tree 14 'OBD flashing attack'
[13.1.1.2.1] Communications Unit (malware delivered by mobile device) Spread malware/virus to mobile device

Misuse the OBD updates or manipulate the CU to gain access to personal data.

Attack trees

Attack trees:

do not capture the dependence of security vulnerability and attacks as well as sequences of attacks steps

Stochastic assumptions are needed to describe systems that have yet to be built and for systems whose set of vulnerability is unknown.

state-based stochastic methods application in security context

Models for security analysis

These models must describe

1. *How and when security attacks occur*
2. *Impact of an attack on the system when it is executed successfully*
3. *Mechanisms, effects and costs of system recovery, system maintenance and defenses*

There are differences with classical dependability

- In the nature and details of security models

Asset: information or resources that could be subject to attack

ADversary VIew Security Evaluation - ADVISE

These set of slides are based on the paper:
 E. LeMay, M. D. Ford, K. Keefe, W. H. Sanders and C. Muehrcke, "Model-based Security Metrics Using ADversary VIew Security Evaluation (ADVISE)," *2011 Eighth International Conference on Quantitative Evaluation of SysTems*, Aachen, 2011, pp. 191-200.

ADversary VIew Security Evaluation - ADVISE

Main objective:
- Compare security strenght of different system architectures
- Analyse threats by different adversaries

Executable state-based security model system
1. A system

2. An adversary view (how the adversary can attack the system)

3. Security metrics

An *attack* is specified in terms of many small attack steps.

Specification of an *Attack Execution Graph* (AEG)

Attack decision function
how the adversary selects the most attractive next attack step

Attack Execution Graph - AEG

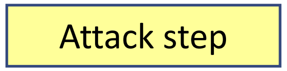
Attack execution graph (AEG)

<A, R, K, S, G>

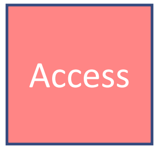
Mobius tool

https://www.mobius.illinois.edu/

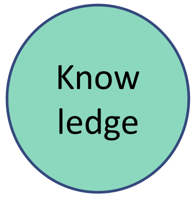
A: set of attack steps



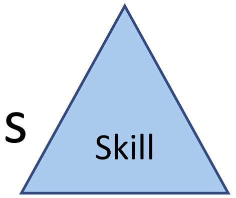
R: set of access domains in the system



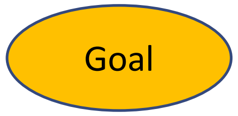
K: set of knowledge items relevant to attack the system



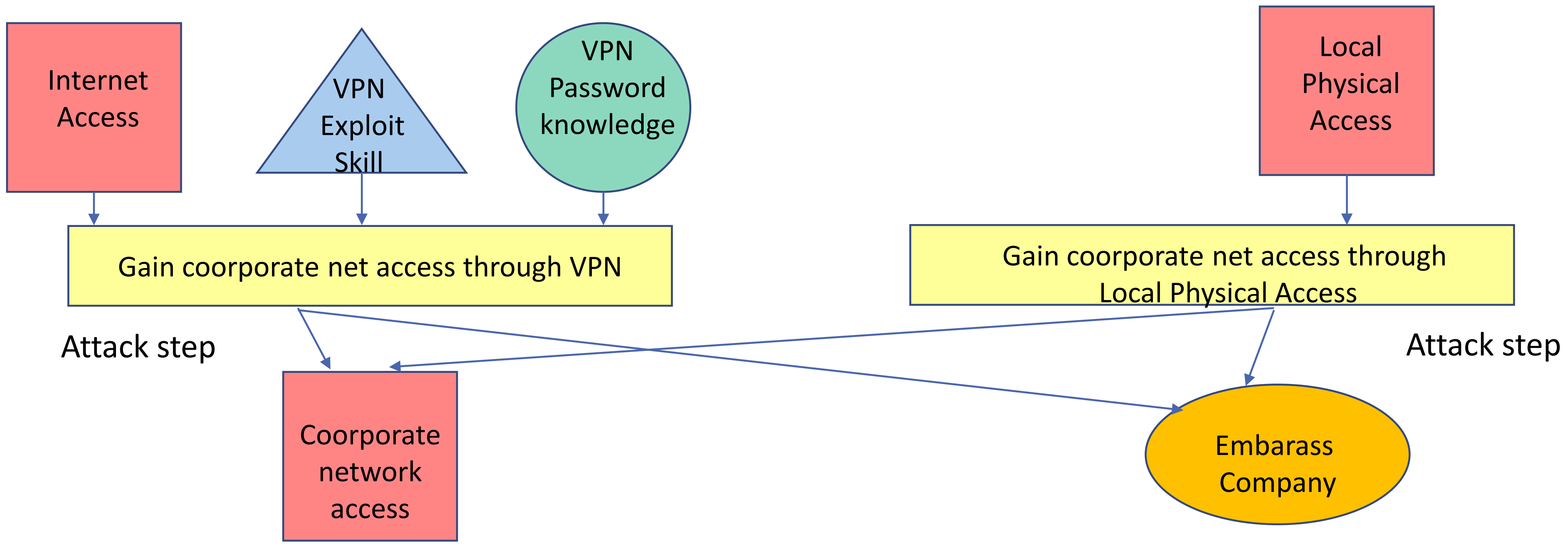
S: set of the adversary attack skills



G: set of adversary attack goals revelan to to the system



ADVISE: AEG



Internet Access
VPN Exploit Skill
VPN Password knowledge
Local Physical Access
Gain coorporate net access through VPN
Gain coorporate net access through Local Physical Access
Attack step
Attack step
Coorporate network access
Embarass Company

Example of AEG taken from paper

E. LeMay, M. D. Ford, K. Keefe, W. H. Sanders and C. Muehrcke, "Model-based Security Metrics Using ADversary VIew Security Evaluation (ADVISE)," *2011 Eighth International Conference on Quantitative Evaluation of SysTems*, Aachen, 2011, pp. 191-200.

Attack Step

$a_i = <B_i, T_i, C_i, O_i, Pr_i, D_i, E_i>$

X is defined as the set of all reachable model states

X= {s1, ..., sn}

B_i: X ->{true, false}
precondition to check if the attack is enabled
The adversary has the access, the knowledge, and/or skill needed for the attack and the adversary does not have what can be gained when the attack is executed with success

T_i: X x R+ -> [0,1]
time required to execute the attack.
T_i (s) is a random variable defined over a prob. distribution function

C_i: X -> $R^{>=0}$
cost of attempting the attack

O_i:
finite set of outcomes (e.g., success and failure)

Pr_i: X x O -> [0, 1]
prob. of outcome o after the attack
(Σ_o Pr (s, o) = 1)

D_i: X x O -> [0, 1]
probability that the attack is detected when outcome o occurs

E_i: X x O -> X
next state when the outcome o occurs

Attack Step do-nothing

a_{DN} = do-nothing

B_{DN}
precondition is always true

T_{DN}
time between two occurrences of do nothing

C_{DN}
cost is zero

D_{DN}
detectability is zero

$E_{DN}(s,o) = s$
the next state is the same of the current state

$Pr_{DN}(s, o) = 1$
there is only one outcome, with probability 1

Every AEG contains the a_{DN} attack step

⇩

there is always at least one attack step in the AEG whose precondition is satisfied

Model state s

A state s in X reflects the progress of the adversary in attacking the system

$s = < R_s, K_s, G_s >$

R_s : set of domains that the adversay can access

K_s : set of knowledge of the adversary

G_s : set of attack goals achieved by the adversary

Adversary Profile definition

Adversay Profile = < s_0, L, V, w_C, w_P, w_D, U_C, U_P, U_D, N>

s_0: initial state of the model

L: attack skill level function

V: attack goal value function

w_C, w_P, w_D : weights for preferences: weight for for cost, payoff, detection probability

U_C, U_P, U_D: utility functions for cost, payoff, detection probability

N: planning horizon

Adversary Profile definition

Adversay Profile = < s_0, L, V, w_C, w_P, w_D, U_C, U_P, U_D, N>

s_0: starting point of the adversary attack
different for insider (more access and knowledge) and outsider adversary

L is the attack skill level function
L : S -> [0, 1] maps each attack skill to a value in [0, 1] (proficiency of the adversary)

V is the attack goal value function
V: G -> $R^{>=0}$, monetary value of each attack goal in the AEG from the adversary viewpoint , more valuable -> larger value

Payoff value P(s) of a state s is a function of the value of all goals V(g) achieved in the model state P(s)= f(V(g))

Adversary Profile definition

Adversay Profile = $< s_0, L, V, w_C, w_P, w_D, U_C, U_P, U_D, N>$

Attack preference weight: attactiveness in each of the three criteria when deciding an attack. They are a value in [0,1]

W_C: relative attactiveness of decreasing the cost in attemping the attack step

W_P: relative attactiveness of increasing the payoff for successfully executing the attack step

W_D: relative attactiveness of decreasing the probability of being detected during or after the attack

Adversary Profile definition

Adversay Profile = $< s_0, L, V, w_C, w_P, w_D, U_C, U_P, U_D, N>$

Utility functions: map the native value of each attractiveness criterion to a [0, 1] utility scale (higher utility values represent more desirable values)

U_C: $R^{>=0}$ -> [0, 1] map the monetary value of the attack step cost to a [0, 1]
lower cost - higher utility value

U_P: $R^{>=0}$ -> [0, 1] map the monetary value of the attack step payoff to a [0, 1]
higher payoff - higher utility value

U_D: [0, 1] -> [0, 1] map the probability of attack step detection to a [0, 1]
lower detection probability - higher utility value

ADVISE model: execution

A_s is the set of available attack steps a_i in state s:

the attack steps whose precondition is satisfied ($B_i(s)$=True)

The attractiveness of the all available attack steps is evaluated from the viewpoint of the adversary with the criteria

- Cost
- Detectability
- Expected payoff in the next state

The *attack decision function* chooses the next attack step

The attack step outcome determines the next state (the outcome is stochastic)

The process is repeated

ADVISE model: attack decision function

Short sighted adversary attack decision function

$$attr(a_i, s) = w_C\, C_i(s) + w_P\, P_i(s) + w_D\, D_i(s)$$

linear combination of adversary preferences weights with the data about attack step

$$\underbrace{P_i(s)}_{\text{expected payoff}} = \Sigma_o \left(P(\underbrace{E_i(s,o)}_{\text{Payoff in the next state reached by outcome o (Ei(s,o))}}) \;.\; Pr_i(s, o)\right) \qquad D_i(s)= \Sigma_o \left(D_i(s,o) \;.\; Pr_i(s, o)\right)$$

$\beta(s)$ best next attack step

$$\{a^* \text{ in } A_s \mid attr(a^*, s) = \max \{ attr(a_i, s) \text{ forall } a_i \text{ in } A_s \} \}$$

one of the maximally attractive attack steps is chosen uniformely

ADVISE model: execution

Utility function U_C U_P U_D are not shown in attr(a_i, s) for semplicity
They should be applied to move towards a common unit of utility.

$C_i(s)$ ---- U_C ($C_i(s)$)
$P_i(s)$ ---- U_P ($P_i(s)$)
$D_i(s)$ ---- U_D ($D_i(s)$)

Ci(s) =2.01 million
Ci(s')=2.05 million
Better mapped -> same utility value

Ci(s) =10.000
Ci(s')=50.000
better mapped -> two distinct utility values

An attack step outcome is randomly generated using the probabilities distributions

The attack step outcomes determine the sequence of state transitions

ADVISE execution algorithm

ADVISE model execution algorithm

```
Time <- 0
State <- $s_0$
while Time < $\tau$ do
    Attack$_i$ <- $\beta$(State)
    Outcome <- o,                        ------ o, Prob$_i$(State)
    Time <- Time +t,                     ------  t, T$_i$(State)
    State <- E$_i$ (State, Outcome)      ------- E$_i$, next state
function
end while
```

ADVISE metrics specification

State metrics $< \tau, \lambda, \sigma>$

τ is the end time $[0, \tau]$

λ is the type of state metrics :

EndProb: probability of being in state s at time τ with $\sigma(s)$=True

AvgTime : average amount of time spent in state s such that $\sigma(s)$= True in the interval $[0, \tau]$

σ is the state indicator function: s= <R, K, G>

$\sigma(s)$ returns True, for states of interest

e.g., $\sigma(s)$ = true if goal g1 has been achieved

ADVISE metrics specification

Event metrics $< \tau, \delta, \varepsilon>$

τ is the end time $[0, \tau]$

δ is the type of event metrics : let ε a set of events

Freq: number of occurrences of events in ε in the interval $[0, \tau]$

ProbOcc : prob. that all the events in ε occur at least once in the interval $[0, \tau]$

ε is a set of events in the model
(attack steps, attack step outcomes, access domains, knowledge or goals)

Example
Frequency of attack step a_i in the interval $[0, \tau]$
ε is equal to $\{a_i\}$

ADVISE model

In the paper:

- more sophisticated adversary decision with a long range planning attack decision function are shown (State Look-Ahead Tree)

- A case study on a SCADA (Supervisory Control and Data Acquisition) architecture is analysed: 2 variants of the architecture and 4 different profiles of adversaries.