The ETSI TVRA Security-Measurement Methodology by means of TTCN-3 Notation

Jan deMeer, smartspacelab.eu GmbH
Axel Rennoch, Fraunhofer FOKUS Institute

A Contribution to the 10th TTCN-3 User Conference June 2011

1. Introduction to the approach:

The European Telecommunication Standardization Institute ETSI has provided a practical Evaluation Methodology, called the TVRA Methodology, with respect to three kinds of system menaces to be **A**nalyzed:

1. System **T**hreats,
2. System **V**ulnerabilities,
3. System Implementation **R**isks.

Threats, Vulnerabilities and Risks (TVR) of a system to be analyzed (thus being identified) by executing seven (basic version 2009) respectively 10 steps (advanced version 2010) according to recent ETSI TS 102 165-1 V4.2.x (2010) TISPAN [10] specification.

ETSI's Evaluation Philosophy behind the TVR-Analysis Methodology is that any security-sensitive system or module must be evaluated and tested against the security perimeter by which a module fortifies her assets. An example of fortification is the so-called Cryptographic Module (CM, s. figure 1) according the specification of the NIST standard FIPS PUB 140-2.

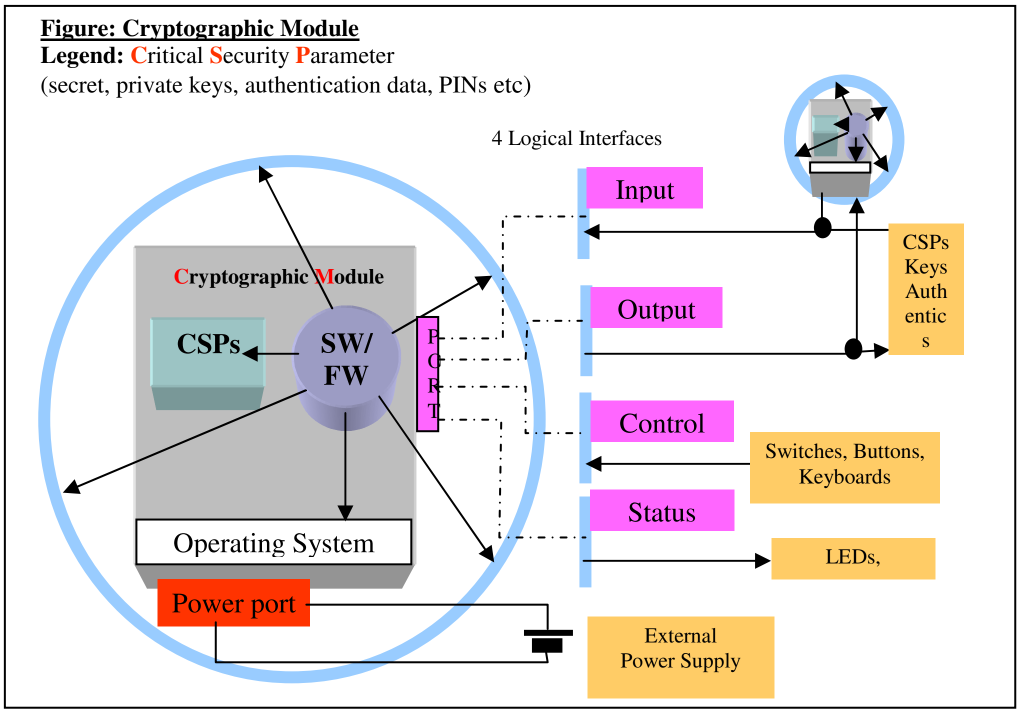

Figure 1: Crypto Module

.

By this contribution we demonstrate ETSI's TVRA security evaluation approach by applying model-based testing techniques and, where appropriate - implementations by applying TTCN-3 notation – to systems being subject of vulnerabilities and threats in a hostile environment.

Security Countermeasure shall fortify the system assets against menaces from unauthorized third parties "testing" systems for vulnerabilities fortified by weak countermeasures. Identification of weak countermeasures is a process of testing activities with the purpose of earning enough information on how to break the countermeasure and having thus access to the system assets.

In order to get an idea on the risks of implemented countermeasures we need a way to evaluate the processes attackers might apply to find a system's vulnerabilities.

Thus to test the quality of a countermeasure we require a continuous evaluation (testing) scheme which is achieved by improving the basic TVRA evaluation approach that operates sequentially by performing certain steps of evaluation towards a process-based approach that applies the same evaluation steps but recursively.

In order to achieve a process-oriented security evaluation scheme we need to invent a couple of tools, respectively the following components:

- A ToE Observer tool at the output at reference point ***y*** of the ToE/SUT operating as the "measurement feed-back channel";
- A Compliance Checker that makes continuously decisions on the difference between the given security objectives from the requirements specification and the current measurements of the security objective from ToE Asset at reference point ***y***.
- An Asset Safeguarding tool, that is able to control access to system assets during system operation, observable at reference point u.
- An Advocatus Diaboli Tool that simulates disturbances from unknown third parties (attackers being part of hostile environment) and which affects reference point ***u*** in order to allow quality evaluations of countermeasures taken by the safeguard.

Notice the 3 components compliance checker, ToE safeguard and ToE observer together form the well-known test system which interacts with the System Under Test (SUT) i.e. the Target of Evaluation (TOE) via the interfaces ***u, y*** and ***r***; whereas ***u*** is ToE input, ***y*** the ToE output interface and ***r*** is the test requirements interface to the testing system.

Hence this contribution is structured as follows: starting with a short introduction into TTCN-3 Model-based Testing Methodology – continuing with State of the TVRA-art of security analysis methodology – followed by an introduction to the evaluation approaches of *linear*[12] and *recursive* TVRA models for analysis of the quality of countermeasures fortifying system assets – next is an evaluation use case taken from authentic communication model based on use of Cryptographic Modules (CM) – Finally the paper concludes with Biography and Conclusions from the work presented and overview of anticipated further studies of security evaluation.

a. The Basic Elements of Testing with TTCN-3 Notation

We need to distinguish between continuous and discrete systems. Continuous systems use signals for communication and discrete systems use messages for communication. Since TTCN-3 is a message-based system we cannot handle continuous systems by TTCN-3. We restrict further our considerations to discrete arbitrary behavior, thus exclude stochastic behavior as well.

The basic elements of testing comprise

a. The test purpose

b. Modules, data types and messages

c. Components and ports

The test purpose just gives advice to the test people in prose or graphic notation using natural language and so-called Message Sequence Charts (MSC). Thus test purpose is part of the test documentation and is thus not executable.

The graphic notation MSC describes test basically by vertical process time axis. States on these time axis are indicated by rectangles. Processes communicate by exchanging messages which are represented in MSC by horizontal arrows originating on one time axis and ending at another one, representing the receiving process.

For example, a minimum "test purpose" comprise the two processes of a Test System (TS) and a *System-Under-Test (SUT)* together with the set of messages which are exchanged among the 2 processes.

[12] The ultimate goal of the authors is to apply the presented approach to non-linear (stochastic) system behavior. However due to its complexity and due to the intention of the authors to use (functional) TTCN-3 Test notation for security evaluation purposes this contribution is reduced to linear system security evaluation.

Modules are TTCN-3 named language containers containing test definitions and test execution control descriptions including required data types and messages. Messages usually structured as a record which is a list of other (simpler) data types and in most cases are defined as part of a standardized protocol. A protocol has a purpose and so the test system has, namely to check whether observable behavior of the protocol, i.e. certain message sequences, fulfill an identified purpose, at least do not contradict with. (We remember, by testing we cannot demonstrate the absence but the existence of errors. The latter would be a contradiction to the test/protocol purpose!) Thus we use the notion of test purpose in the sense of the objective of the tested protocol.

b. Behavioral Models for Security Analysis

Basically TVRA is a security analysis methodology developed by TISPAN WG7 (NGN Security - cp. TS 102 165-1, TR 187 002) for the purpose of analyzing and of evaluating complex system behaviour with respect to the likelihoods of possible Threats, Vulnerabilities and Risks.

Current TVRA analysis methods focus on system behavior (SUT) fortified by countermeasures which are able to withstand smart, i.e. self-modifying, attacks. In case of smart attacks you cannot wait for the firewalls' breaking. Instead you must "fix holes or heighten the wall itself" in short time. In other words countermeasures of a Target of Evaluation (ToE) must be able to be adaptive to the effort of attackers of having in their cross-fade vulnerabilities to break into the system.

According to the security objective to achieve certain levels of security we have to consider the behavioral variables of the identified assets, i.e. the asset state vector ***y*** of the ToE, in terms of testing of the SUT. More formally the behavior of a certain asset "i" can be defined as the output-input [y/u]i behavior relation of the ToE (see reference points in figure 2).

Behavior(state vector ***Y)*** of the ToE can be described by continuous, stochastic or, arbitrary signals.

a) Continuous behavior is described by linear or non-linear differential variables: [Y| dy/dt]

b) Discrete stochastic behavior by random variables: [Y| yu<Y<yo]

c) and discrete arbitrary behavior by variables obeying finite value sets: [Y|{a,b,c,…d}]

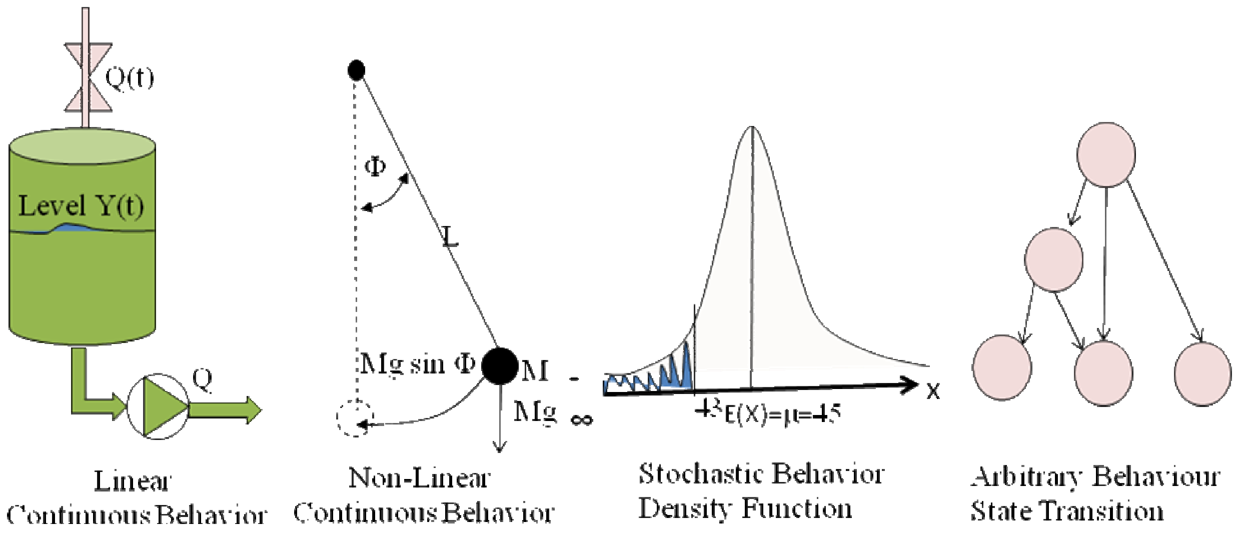

Figure 3: ToE Behavioral Models

From Figure 3 we can derive that communication is based either on continuous signals or discrete messages. E.g. by messages you may request for a service at reference point ***u*** (see figure 2) and expect an answer on the availability of this service and thus give rise to (discrete) state transitions observable at reference point ***y*** (see figure 2).

In continuous systems even we do not have such discrete state transitions we have the signals ***u(t)*** and ***y(t)***. In contrast to discrete systems we can relate output and input signals to each other, i.e. to build the differential quotient ***dy/du***, which allows us to construct a system from components. Notice that signals can be captured by discrete messages as well. In the case of which the signal is replaced by a stream of messages, each carrying a certain value of the signal.

Security Analysis - in our opinion - must be applied to complete systems consisting of components that influence each other. Hence the below invented TVRA process model benefits from this fact and takes a system view taking all dependencies into account that influence the behavior of considered assets.

The TVRA methodology however must thus fit with all three types of behavioral models. It is obvious for continuous or stochastic behavior that adaptation to a target value (EAL) makes sense. In case of the discrete behavior it is not the case, since the behavior is neither random nor continuous but arbitrary.

Arbitrary behavior can easily be tested: you send an arbitrary discrete probe of defined value sets and get back the current value of the probed variable. This way of testing is state of the art. If the defined value sets are not too large you can approach it by a kind of completeness test.

Stochastic behavior can be tested by collecting a sufficient amount of discrete probes that allow you to derive a density function of the collected values of the variable. The density function can be considered as a model to derive appropriate probes for system testing of stochastic behavior.

Linear[13] differential behavior allows to fix a certain target point of working and to keep track of this target. By testing you can probe the delta between current and target point of working or, in other words you can measure the existing *current security gap* to the target security working point.

The architecture of the TVRA process model of the next section is arranged such that all three behavioural evaluation models (see figure 3) can be implemented.

Threats, vulnerabilities and risks model the logical security-aspects of a system considered whereas the implementation effort to fortify the assets of the system obeys real aspects of programming. The correctness relationship requires a compliant overlay between the underlying security model and the implemented system constraints and vice versa.

c. **TVRA model-based testing**

The TVRA security model contains 2 relationships, i.e. Compliance and Implementation Relationships; whereas Compliance is a “horizontal” and implementation a “vertical” relationship with respect to the V-Model of System Engineering. Horizontal Relationship describes a comparison between the requirements specified in one or more standards and an implementation that must be shown to meet those requirements or not.

Vertical relationship describes refinements (i.e. UML aggregation, generalization, specialization) or extension (i.e. UML composition, dependency). Relationships are required to implement respectively to prove the correct behavior of a given aspect or a requirement, e.g. given a security requirement it *consists* usually *of a composition* of communication aspects *and* security aspects; the same security requirement *depends* from certain security objectives. Thus the security requirement is to be extended with communication aspects and with their dependencies for other objectives.

The same security requirement might be refined by enumerating all aggregates it addresses, e.g. the number of assets of a system considered or, requirements requiring specialized communication aspects etc.

Thus the compliance relationship (CR) is used to check, e.g. by applying testing techniques, up to which extend an implementation meets its requirements and the implementation relationship (IR) is used to prove design steps by more sophisticated “model checking” tools. A compliance check is executed on both on the model and on the related implementation. In case a test shows the model and the implementation behave in a similar expected way; it can be stated that the modeled requirement is compliant to the behavior implemented.

Contrary a model check works with a behavioral hypothesis. If the hypothetical behavior is a requirement it must be derivable from the model and thus from the implementation since it has been proven to be compliant to the model.

Model-checking techniques are more powerful than testing techniques since model-checker are able to prove non-existence of a property, in case it is not derivable. This is not possible by performing test cases since only existing properties can be triggered from outside. If there is no reaction on your trigger, you don’t know whether the tested feature does not react or does not exist.

By applying the TTCN-3 testing technique, model checking is not (yet) achievable but model (based) testing is. Whereas by model checking we achieve valid system designs, by model testing we achieve confidence in an implementation. In the first case a hypothetical design step before taking is model-checked, i.e. it gets proven as

[13] We do not consider non-linear systems that behave as sine or cosine functions since its complexity goes far beyond the intention of this contribution. Even linear systems are not considered by all its complexity, i.e. for simplicity purpose we just invent time-dependent state variables X(t).

a possible valid or invalid model. Whereas in the latter case the model simply helps in deriving tests which shall be able to run against an implementation. If they don't run you have no idea what is wrong, the test case, the model, the tested system or, the system's hostile environment? In case the test runs as expected, you just get known something exactly about that property tested which in turn may improve my trustworthiness in the system, e.g. as it may show (security) behavior as being successfully tested, i.e. certain countermeasures against threats, vulnerabilities or risks do exist but in the tested way.

If you want to get confidence into countermeasure against threats which may break into your implementation you may derive tests from a compliant model and to apply them onto the implementation in order to find out the trustworthiness of implemented countermeasures.

How can such an evaluation of the trustworthiness of a countermeasure be structured?

Let us make an example of how to test vulnerability of a fire wall. Firewalls usually operate with filtering rules on OSI level 3 (network) and level 4 (transport). Level 3 allows identification of hosts by using IP addresses whereas level 4 allows identification of process domains by using port numbers. Assume that there is a Trojan process in the tested host which will be scheduled at certain intervals during a considered period of time, e.g a day, at unknown ports of the host and if met, "opens the door" to the security module and steels some Security-critical Parameters. We can now construct the test to measure the response behavior of the system at a certain range of ports and at statistically distributed time periods. This is a stochastic scenario by which we get a density measure of certain test runs that e.g. to succeed more or less probable to interconnect to the Trojan.

Another example is of how to test vulnerability of implemented procedure of authentic communication. This example however is based on linear (functional) system behavior

2. ETSI TVRA Art of Security Analysis

ETSI's Security Analysis Methodology is based on the generic Security Analysis Model (SAM) TVRA[14] [10]. The anticipated generic security goal of the TVRA is the minimization of the probability of any instantiation of an object from class "unwanted incidents". Thus minimization is a generic adaptation process of the behavior of a set of system variables (see also section "Behavioral Analysis Models") captured by the Target of Evaluation (ToE) towards risk minimization.

TVRA provides relationships between security design classes to be necessarily applied in order to achieve a secure target of evaluation. The SAM further describes the security data types (SDT) of both threats and of security objectives. Whereas the threat SDT defines the basic assets, i.e. communicating parties, communication protocols, services and messages and, the objective SDT defines the basic countermeasures such as session key exchange, cryptographic activities, activities to identify DoS, authentication protocols and accounting.

The security design relationships comprise

1. the Secure System Design Class as an aggregation of 1 or more "Asset" (System Module ToE) Classes;
2. the specialization of an asset class into an appropriate countermeasure class;
3. Usually (but not necessarily) assets have vulnerability classes to which countermeasure classes must fit;
4. Vulnerability class is composed of weaknesses and threats;
5. Certain Threats are always enacted by some threat agents;

Basic System Assets are communicating parties, communication protocols, available services and messages exchanged which are targets of threats, i.e.:

1. Interception: CommParties Interceptor -> CommParties
2. Manipulation: Message SetOfFaults -> Message
3. DoS: Service DoSActivity -> Service
4. RepSend: SendProtocol RepActivity -> SendPrtotocol
5. RepReceive: ReceiveProtocol RepActivity -> ReceiveProtocol

The security objectives to be maintained through all attacks from threat agents comprise so-to-say the countermeasure assets, e.g. Session Key Exchange, Cryptographic Actions, Load Management, Authentication Protocol, Accounting which are all powerful tools to withstand the threatening of the former assets:

1. Confidentiality: CommParties SessionKeyExchnge -> CommParties

[14] TVRA = Trust – Vulnerability – Risk Analysis Model

2. Integrity: Message CryptoActivity -> Message
3. Availability: Service DoSIdentification -> Service
4. Authentication: CommParties AuthenticationProtocol -> CommParties
5. Accountability: User AccountingActivity -> CommParties

3. TVRA Process Model

The analytical TVRA process model is called to be recursive, for short rTVRA, because it can be used to track a certain security countermeasure for its quality, i.e. Evaluation Assurance Levels (EAL), to withstand attacks or jamming from a system's environment.

The approach is based on the observation of the system state variables ***y*** (reference point) and of feeding the system state measurements back to the security objectives that form the reference point r. The latter are used as the referenced security objective values that must be achieved by the countermeasures implemented in the system to defend or minimize threats, vulnerabilities and risks.

Notice that there is another reference point ***u*** that works as the input (control) interface to the Target of Evaluation (ToE). Input interfaces are vulnerable to threats and attacks from third parties such that unwanted incidents become possible and may hinder the control input from the countermeasures to affect security of the ToE in an unwanted way.

In general, security adaptation should be recurred until a satisfactory result has been achieved. What a satisfactory result constitutes should be defined by specifying reference goals and purposes for. Reference Goal and purpose shall also include statements related to targeted EAL if applicable. In addition, assumptions frame the analysis by stating what is included and what are assumed in the ToE and the ToE environment. The assumptions should be used to determine the validity and relevance of the result. Similar ToEs will not necessary result in the same result if their assumptions are different.

ToE describes the target of analysis. There might be security objectives formed as statement of one of properties CIAAA[15]. These objectives are refined into functional security requirements according to ISO 15408-2. This is the link to Common Criteria. This means that the set of functional security requirements can be tested against ISO 15408-2 according to the targeted EAL. Security objectives are used to guide the identification of assets. Assets are contained in the ToE. Assets may have weaknesses.

These weaknesses lead to vulnerabilities resulting in a risk, with an associated risk level. The risk level is a decision point. The goal of an analysis is to contain the risk (reduce it or remove it). If the risk level is acceptable, there is no need to introduce countermeasures and the analysis terminates. If the risk level is not acceptable, countermeasures are introduced in step 8.

Countermeasures reduce or remove risks. The actual effect of a countermeasure should be verified or tested. Introduction of a countermeasure iterates TVRA. Countermeasures are added until the risk level is acceptable. Countermeasures are tested to ensure that they are sufficient and correct. The way in which this is done may vary depending on many factors, amongst others, the targeted EAL.

One path is countermeasure cost-benefit analysis, another is CC-based security testing, a third path is security verification. We have tried to illustrate the process in the figure on the next page. The existing TVRA provides by its 10 (in older versions 7) steps only a linear approach. Linearity means the steps are linearly executed. The 9th step included in the analysis component does not really decide on a functional optimum state but it provides for some chosen countermeasures a simple cost/benefit relationship. This occurs at reference point ***r*** (cp. Figure 2 below).

However Complex System Analysis requires a recursive approach which can be achieved if we re-arrange the 10 steps of the linear to a more appropriate recursive approach. More appropriateness means that the decision-taking step shall move towards the head of recursion in order to determine sufficiency of the goals/purpose of security.

Decision is taken on the measured effect of countermeasures and calculated risk likelihood. It is then fed into the embedded ToE, (i.e. ToE is embedded into its environment), which is the reference point ***u***, the input interface to ToE. Since the embedded ToE comprises the inventory of assets which is matters of attacks. Thus ToE needs to be a constituent of one or more of CIAAA Categories, provides well-known objectives (2) and is compliant to the ISO/IEC standard 1408-2 due to functional requirements (3)

[15] CIAAA = Confidentiality, Integrity, Availability, Authenticity, Accountability the basic Security Categories

In order to achieve recursion the decision-making component (5,7) must be fed with measurements of the security characteristics of the embedded ToE from reference point ***y***, which include assets, vulnerabilities, risks and certain information about likelihoods of attacks. This list is so-to-say the heart of TVRA which is formally composed as a compound security vector of named values.

The measured values of the security state vector ***y*** are thus fed-back to the decision-making component that decides step-wise on cost, benefits, and on current counter measures to be applied to the embedded ToE at reference point ***u***, which is the begin of a new cycle.

The recursive TVRA (figure 2) from the figure above re-combines the 10 steps from the linear TVRA into the following 6 recursive analysis phases structured by 3 reference points, in order to compute recursively optimum countermeasures and thus minimize risks, avoid vulnerabilities and prevent threats. (Notice that the bracketed numbers contained in the list of recursive phases below, indicate the related standard TVRA steps.)

1. Determine the functional objectives(2) and the functional security requirements (3) by means of the goal specification tool;
2. At reference point ***r*** the objectives and requirements from phase 1, generated by the goal specification tool - are compared with the measurements (cost-benefit analysis) taken from the observed embedded ToE. Notice that this is an additional activity, namely test and measurement (10) of the ToE state vector ***y*** which constitutes the resultant behavior of the embedded ToE constrained by certain current countermeasures of the ToE Safeguard Tool.
3. The objective-measurement delta derived by computation (***r-y***) is fed into the compliance checking tool that makes vulnerability(5) and risk(7) analysis in order to reason for countermeasures-cost-benefit decision(9). From these computations new sets of countermeasures on vulnerabilities are determined in order to minimize ToE risks.

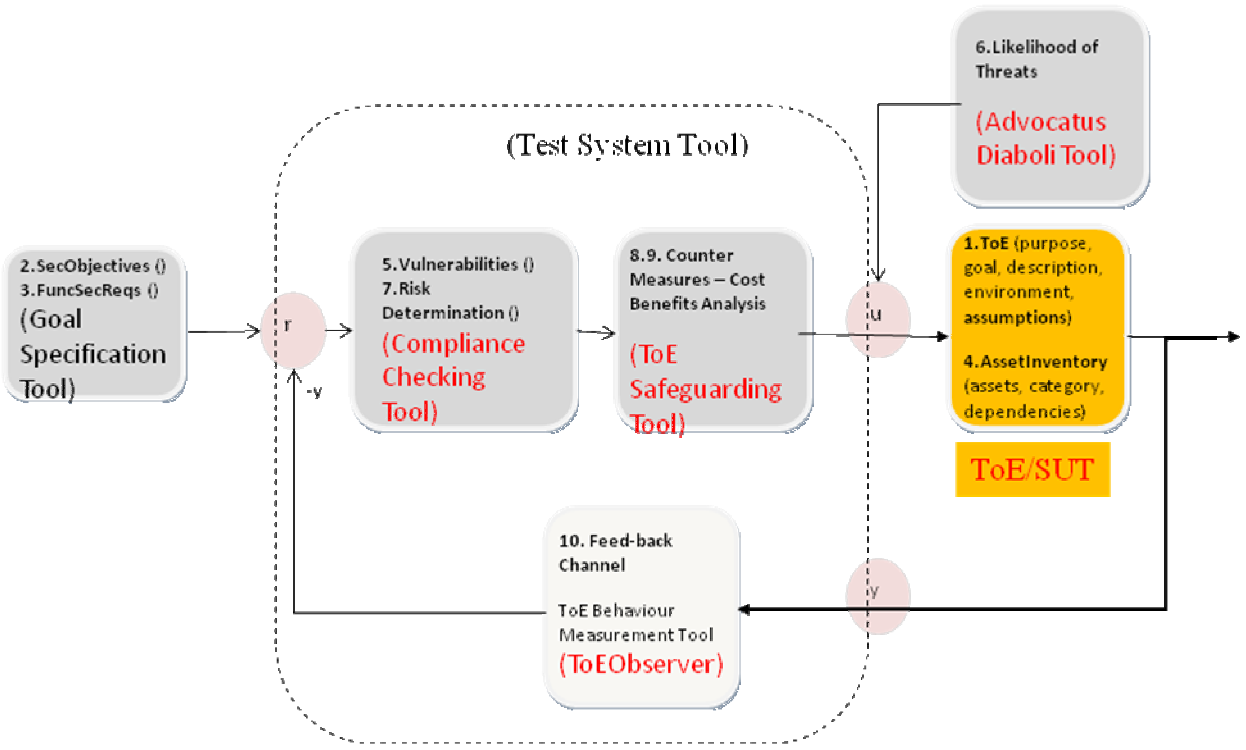
(Test System Tool)
2.SecObjectives ()
3.FuncSecReqs ()
(Goal Specification Tool)
r
-y
5.Vulnerabilities ()
7.Risk Determination ()
(Compliance Checking Tool)
8.9. Counter Measures – Cost Benefits Analysis
(ToE Safeguarding Tool)
u
6.Likelihood of Threats
(Advocatus Diaboli Tool)
1.ToE (purpose, goal, description, environment, assumptions)
4.AssetInventory (assets, category, dependencies)
ToE/SUT
10. Feed-back Channel
ToE Behaviour Measurement Tool
(ToEObserver)
y

Figure 2: recursive TVRA Process Model

4. At reference point ***u*** these sets of countermeasure are fed to ToE that is embedded into a certain environment which has access to interface u too. Thus Reference point ***u*** can be matters of attacks. The

"Advocatus Diabili" tool simulates the likelihoods of threats (6) and thus disturbs Safeguard – ToE interaction at interface ***u***.

5. The embedded ToE maintains identified assets (4) and thus tries to adopt the countermeasures from interface ***u*** of the compliance-decision-taking tool and subsequently behaves as it is currently observable at reference point ***y*** which can be thought of as the state vector of embedded ToE describing the effect of current countermeasures.
6. The values of state vector ***y*** are measured/tested and fed back compliance-decision-taking tool, where they are in turn be optimized with respect to the given objectives and requirement goals.

Thus several recursion cycles may be done in order to fit more precisely the countermeasures at reference point ***u*** to identified vulnerabilities, threats and risks. In other words their likelihoods are reduced adaptively.

4. The Authentication Model

In detail we want to specify the Authentication Asset from above. As it can be derived from the definition above there are two types the *CommunicatingParties* and the *AuthenticationProtocol*. The operational semantics of the transformation arrow -> means that the authentication protocol is bound to the communicating parties for execution.

The communication parties comprise the user called *Principal* (*Prl*), another party to which the principal relays in communication, the *Relaying Party* (*RlP*) and an independent *Identity Provider* (*IdP*). The latter party *IdP* keeps the identities of the principal and all relaying parties. *IdP* is also allowed to generate keys and to distribute them among all identified parties.

a) Specification of the Authentication Protocol

The authentication protocol is performed by 6 steps that are specified by the same operational semantics as above and look as follows:

1. Authentication Request: reqAuth: principal namelist -> identityProvider
2. Authentication Grant: grantAuth: identityProvider ticket -> principal
3. Authentication Resolution: resolveAuth: principal CM -> CM
4. Pose Authenticated User: pose: principal CM -> relayingParty
5. Authentication Resolution: resolveAuth: relayingParty CM -> CM
6. CommunicateAuthenticated: commAuth: [CM CM service] -> [CM CM service]

The authentication protocol which is executed by 6 steps involving all communicating parties. Furthermore the authentication protocol uses so-called *tickets* that are generated by the IdP and that are encrypted messages that contain the session key, called nonce, requested by the principal. A ticket is at first sent to the principal which is able to decrypt it because the principal is legally registered at the IdP and thus both use same symmetric key for confidential ticket exchange.

The IdP adds to the principal's ticket also the ticket to the relaying party but does not send it to. Thus the principal receives together with his own ticket the encrypted ticket of his relaying party too. Since the principal can only open his own ticket but not tickets for relaying parties the principal simply packs it out and forwards it the receiver obeying the key allowing him to open it.

Opening of a ticket means authorization resolution since each party registered at the identity provider has received its own symmetric key for communicating with the identity provider by means of encrypted tickets.

The resolved tickets get saved into cryptographic modules (CM) hosted by each communicating party.

Finally the prinicipal and a relaying party can now communicate by using the authentically and confidentially exchanged nonce.

b) Definition of the Authentication Security Objectives

The security objectives of category authentication are defined in the document "ETSI TISPAN NGN SECurity Requirements" [TS187001v4009]. In Annex E the security objectives are assorted according to the 7 security categories of CIAAANR, i.e. Confidentiality(Secrecy), Integrity, Availability, Authentication, Accountability, Non-Repudiation, Reliability.

The security objectives of category Authentication are as follows:

1. OBJ-AUTH-01: “It should not be possible for an *unauthorized user to pose as an authorized user* when communicating with an application or other user of an NGN[16].”
2. OBJ-AUTH-02: “It should not be possible for an NGN *to receive and process management and configuration information from an unauthorized user.*”
3. OBJ-AUTH-03: “An *NGN user* should be able to *authenticate the source of an incoming media stream.*”
4. OBJ-AUTH-04: “ An *NGN entity* should be able *to authenticate the source of an incoming signal.*”
5. OBJ-AUTH-05: “An *NGN entity* should be able to *authenticate the identity of any other NGN entity that it communicates with.*”

In the sequel and for simplification purposes we will demonstrate the TVRA analysis methods and the TTCN-3 testing testing techniques just by the only security objective of OBJ-AUTH-01.

c) TVRA analysis methods demonstration

Here we will follow the more compact demonstration of the recursive TVRA process model; but trying to answer the major questions of the basic TVRA analysis technique on the Authentication Model as it is presented above.

1. What are the functional and architectural risks of the Target of Evaluation (ToE) of an NGN that copes with the above Authentication Model?

   In our case of Authentication the ToE means the “Absence of Anonymity” in the NGN, i.e. the 5 authentication security objectives must be tested positively such that only authorized users can communicate and all sources of data can be authenticated.

   The ToE function is to establish a trusted channel between all authorized communication participants. The ToE architecture is the identification of reference points (interfaces) at which risks occur. In case of the authentication model 3 reference points can be identified, i.e. the communication for authorization requesting between Prinicipal and Identity Provider(1); between Relying Party and Identity Provider (2) and finally the trusted communication between Principal and Relaying Party (3). The risks are that authorization tickets and data sources can be spoofed, unauthorized users get access to trusted channels etc.

2. What is the risks of an embedded ToE to be manipulated by process or configuration information issued by unauthorized users?

   The authentication model copes only with users being identified by the authorization identity provider; hence it is a closed user group. However the communication network serves for all users, authorized and unauthorized ones. Thus network users might generate maliciously or by intention management information that provide access or some information from inside of the authorized closed user group.

   Thus a test case should provide some confidence that this will happen very unlikely. At the inferace u not only countermeasure information get exchanged but also management and configuration information. The latter could affect the security policy of the ToE for example.

3. What is the risk of a ToE (authorized NGN user group) being not able to identify any source of information expected at its interface u?

   Another test case should give confidence that sources of data streams or signals can very badly be spoofed.

4. What is the risk of any unauthorized NGN user (ToE) that he is recognized as an authorized one or vice versa?

[16] NGN = „Next Generation Network“ is the name of the ETSI TISPAN NGN Working Group 7 on Network Security Requirements such as Security Policy, Authentication, Access Control, Accountability, Identity, Security Registration, Communication, Data Security Requirements and Privacy.

A test must provide confidence that authorized and unauthorized users that are served by the NGN get always correctly identified.

d) TTCN-3 testing techniques demonstration of Authentication Test Case

The test cases are implemented by TTCN-3. As an example the test case of *obj-auth-01* is implemented according to the model presented in section 3:

```
type component Principal {
port idpPort idp; port rlpPort rlp}

type port idpPort message{
in ReqAuth, CommAuth, ResolveAuth;
out grantAuth}

type port rlpPort message{
in Pose, CommAuth }
testcase TC_OBJ_AUTH_01a runs on Principal {
…;
idp.send(ReqAuth:m_req1);
idp.receive(GrantAuth:mw_resp1) -> value v_ticket;
…
rlp.send(Pose:v_ticket);
rlp.send(CommAuth:m_getService);
alt{
[] rlp.receive(CommAuth:m_accept) {setverdict(pass);}
[] rlp.receive(CommAuth:m_reject) {setverdict(fail);stop}
}
…
```

5. Result Analysis and Conclusions

By the four test case categories of authentic communication (absence of anonymity) – impossibility of manipulation by management information (configuration spoofing) – data source identification – authorization spoofing – the feasibility of the feasibility of the improved TVRA process model has been applied and successfully demonstrated.

Furthermore it has been demonstrated that countermeasure must be flexible in order to “learn” from attacks performed at the u- (management and data) interface of the ToE. Since both signals, the attacker signal and the countermeasure signal interfere at u, the result of the interference is immediately be observable from the ToE output behavior.

In order to keep the learning curve of the system under test steeper than the learning curve of the attacker we must prevent the attacker from having easy access to the ***y***-output interface of the ToE - at least to have later access than the built-in measurement tool. This is because the next adaptive learning step to compute fitter countermeasures must be executed before the attacker can apply his own fitness function to his manipulation trial.

We think the suggested improvement of recursive TVRA process fits best with the adaptation problems not considered with linear TVRA, provided that attackers are skilled such that they apply learning curves respectively fitness functions to their own attacking trials. It means that there is a high risk of too slow adaptation of countermeasures to smart attackers.

This risk can be reduced by preventing non-authorized users from the observation of the ToE behavior and (2) by inventing a measurement channel that connects the ToE behavior directly to the anticipated security requirements which are as well be hidden to any unauthorized NGN user, and thus being able to quickly react on critical ToE behavior, timely enough before an attacker gets aware of.

6. Short selected bibliography of relevant standards and testing documents:

[1] ISO/IEC 15408 (1.27.16.) Part 1-3 IT ST Evaluation Criteria for IT Security
[2] ISO/IEC 18045 (1.27.36) IT ST Methodology for IT Security Evaluation (Common Evaluation Methodology)
[3] ISO/IEC 24759 (1.27.51) Test Requirements for Cryptographic Modules
[5]ETSI ES 201 873 v4.2.1 (2010-07): MTS – The Testing and Test Control Notation Version 4.2.1; part1-part5
[6] Life Cycle Cost Analysis Handbook, Edition 2005, T. Maerig, AIA, N.Coffee, AIA, M.Morgan PMP, Alaska
[7] SAE International, www.sae.org, G11 Reliability Committee: ARP5580 – Recommended Failure Modes and Effects Analysis, Practices for non-automobile Applications
[8] Peter Liggesmeyer, HPI Potsdam: Formale und stochastische Methoden zur Qualitätssicherung technischer Software.
[9] Marvin Rausand, Norwegian University of Science and Technology: The Book of System Analysis, Chapter 3: Failure Modes, Effects and Critical Analysis, (2005)
[10] ETSI TS 102 165-1 v4.2.3 (2010-12) TISPAN Methods and Protocols; Part 1 Method and Proforma for Threat, Vulnerability Analysis