# Secure System of Attack Patterns towards Application Security Metric Derivation

K Raja Sekhar
Professor
Department of CSE
KL University Vaddeswram

L S S Reddy, Ph.D
Director & Professor
LBR College of Engineering
Mylavaram

U J Kameswari
Assistant Professor
Department of ECM
KL University Vaddeswaram

## ABSTRACT
Attack pattern system exhibits a unique property of pattern sequential cascading nature which can be identified during the design phase of an application system implementing security scenarios. In this paper a mathematical framework of secure system of attack patterns is presented to verify the stated design specification property along with theoretical back ground work. The framework defines 12 definitions of secure system of attack patterns, propositional transition system, computable functions and other supported elements. The frame work establishes 15 specifications with associated lemmas and theorems to construct and build the background towards verification of proposed system. Finally the proposed attack pattern system is assessed against the number of patterns, resources and other pattern properties with the help of simple security scenario.

## General Terms
Application Security, Attack patterns, Properties of attack pattern

## Keywords
Secure system of attack patterns, Computable functions, Configuration mapping Points and cascadability of patterns

## 1. INTRODUCTION
Attack pattern system is inherently similar to design specification system and the key role in the behavior of attack pattern system is played by communication and interaction with their computing environment involving target software and machines with in predefined sometimes unknown security context. This overall scenario is specified with design specification where pattern representation is made against the computable resources it requests and verifying that this pattern is consuming the resource without any compromise in its execution of the whole pattern system. Computing system with attack patterns can be considered as design specification in the sense that pattern sequential behavior within the context of an application preferably web application can be stated as per the following specification statement.

"As the solution to one pattern is applied that solution may implicitly derive or yield another pattern behavior and implementing the next pattern in sequence again yield another pattern and this sequence of cascadability is possible within the context of an existing security scenario where patterns can be applied to the code at different levels in the application development".

The implementation of such a scenario is applied by defining the computing environment with the help of model specification. The model attempts to build and construct the external supporting elements needed to prove and verify the said specification. For example the external environment for any application begins with web itself. Then the levels of environment variables list starts from session with variable number of parameters. Activation zone containing the target machine(s) and the software is the major resource for computing system of attack patterns. This activation zone contains the target area in which the attack scenario is applied. The other parameters that produce computing system are pay load field, networks, database, client and server machines and other related fields. For example consider the brute force pattern where the basic security scenario contains algorithm, the length of key in terms of computation complexity. To break the key the target environment consists of only one computing system where as all the parameters are mere components.

The system of attack patterns and its solution is executed with the help of mathematical elements described with subjects, objects, classifications, requests, outputs or decisions, configurations, access matrices, computations or functions, propositions, sequences and attributes. Patterns including pattern structure containing the pattern solution are the main subject and subject parameters which are functionally mapped to the objects in the proposed security system are states of extended finite state machine, the scenarios of security requirements with test cases, propositions

## 2. SECURE SYSTEM OF ATTACK PATTERNS
To verify the design specification mentioned in the above section, we initially defined a secure system of attack patterns to hold the pattern input, pattern representation and pattern resources. So the Secure system of attack patterns contains three elements of Patterns, Propositions and Computable functions. The pattern demonstration is done with Proposition set called Prop set and finally Pattern requirement is exhibited with Computable Functions. The pattern consumption with resource variable during its implementation is mapped and confirmed with the help of configuration mapping points.

To facilitate the above mentioned scenario for design specification, the model designates the following definitions of 'Secure System of Attack Patterns', 'Valid Proposition Set', 'Computable Function set', 'Configuration Mapping Point' to describe the behaviour of Secure system of attack patterns in terms of representing all the properties of the attack pattern modelled with predicate and proposition logic including constants and variables describing the logic against the basic resources a pattern requires to characterize the behaviour. The objective of these four functions is to provide in depth analysis of pattern system and ensuring that pattern is consuming pre defined resources without any destruction throughout the execution.

Definition 1: Secure System of Attack Patterns – S

A Secure System *S* of Attack Patterns is defined on subjects {S1, S2… Sn} and Objects {O1, O2, O3… Om} with number of inductions using mappings, access matrices, Classifications involving attributes of subjects and objects applying propositions, computable functions based on sequences to derive the decisions or outputs. Each object includes security scenario with code snippets or code blocks or code components representing some security scenario or application development with security requirements and test cases in the form of programs containing code segments called synthetic code expressed in the form of pattern data requirements.

A secure system is defined as follows

$$\forall Si \;\&\; Oj \exists Mm, Nn, Cq, Ux$$

Such that

$$S = \Im(Pi, Fj, Ck) \Rightarrow Dt$$

Where m is mapping, n is access matrix, q is computation and x is classification

Satisfying the following conditions

1. Pi is set of all valid propositions (definition 2) associated to the attack patterns
2. Fj is the set of functions to evaluate the pattern characteristics
3. Ck is the set of Computations derived by the functions paired with proposition termed as configuration mapping points.
4. Dt is the output associated with acceptance or rejection

Definition 2: Valid Proposition Set - *Prop*

A set of Propositions is valid defined on Secure System *S* if

1. All the propositions or Predicates are derived from the attributes or properties (whether they require resources or not) of the patterns applied in the secure system S.
2. Each Proposition or Predicate Pi reflects or maps to one (or more) of the state of the extended finite state machine (may be at more than one state) represented by Proposition Transition System (PTS) and details each property of the Secure system of attack patterns
3. Pi is unique against the pattern $\forall i$ Pi $\in$ Sj where j is unique pattern and Pi is the final propositional or predicate logic expression constructed or computed from the set of patterns

   I.e. *Prop* = {Pi/ Pi $\in$ Set (Sj) and $\exists$ j such that Pi $\neq$ Pj and $\forall iPi \in Sj$ where j is subject attack pattern

Definition 3: Computable Function Set (F[j])

A Computable Function Set is set of all computable functions defined over the resource type belonging to the pattern subject in S (Secure System of Attack Patterns). In order to create the secure system the model confines and defines 9 types of full functions and 3 partial functions based on the usage of the application requirement restricted only to specific category of web applications and can be extended to other category also.

Definition 4: Configuration Mapping Points (CP)

A Configuration mapping point <*Prop*, F[j]> is a pair defined in S over *Prop* satisfying the following conditions

1. All the propositions Pi from Prop set maps against the functions defined over resource type
2. Each proposition maps one or more functions defined over resource type

I.e. $\exists i$ for Pattern i such that $\forall j \;\; \exists k$ such that *Prop* (Pj) → F {1…k} for functions

To assist the model verification of the said design specification, we construct the logical equivalence of the system 'S' with a transition system called proposition transition system (PTS). [1][2][3] Provided the background work needed to define a Transition system with propositions to understand the behavior of any design specification. Many of the contributions are available where a transition system is converted to an equivalent finite state machine supporting variations of PTS through assorted properties of transition systems. [4] Supported construction of labeled transition system which is used for SCSI-2 systems. The variations in PTS are possible in their interpretation and refinement of the model checking. [5] Presented a detailed work on the interpretation of PTS for model checking refinement.

In our work we have defined a generalized Proposition Transition System (PTS) that exhibits the anticipated behavior of the applied system to receive the preferred input, builds the process and outputs the estimated output. The PTS accepts the pattern properties in terms of propositions and relates them to the respected states with proper meaning and shows the transition paths as per the predefined behavior as described in pattern system 'S'. That is we extended the PTS with pattern actions to generalize the conduct of Pattern system 'S'.

The following definitions represent and symbolize propositional transition system under the categories of 'Fully or Totally Transition System', 'Computation Oriented Fully or Totally Transition System', 'Propositional Transition System', 'Path in Transition System', 'Reachable state of PTS', 'Computation or Trace', 'Acceptance or Path Execution', 'Pre Condition and Post condition Set' which are defined to construct and show the equivalence behavior of pattern system with the help of extended transition system to check and verify that the assumed model is acceptable and suitable to the system of attack patterns. To prove the properties and behavior of the pattern system the model builds the state machine with mapping between propositions and computable functions defined as per the definitions.

Definition 5: Fully or Totally Transition System (FTS)

A Fully or Totally Transition system is a 3 tuple system as {S, T, $\partial$} defined over Secure system of attack patterns Where

S is non empty finite set of states (3 types of states defined)
T is non empty set of transitions and
$\partial$ is output transition function associated with T

Definition 6: Computation Oriented fully or totally Transition system (CFTS)

A CFTS is FTS where the computation always begins from the initial root node. A state may have various properties to follow and each of the state and its properties are described on propositional and predicate level in parallel to the computations with predefined functions applied. A Computation Oriented fully or totally Transition system is FTS with computation starting at the root node where each computation involves with the functions defined for the system.

The machine assumes that at particular point of time (initially at the start state) all the propositions are declared true against description of each state. A transition system where every state is assigned such description will be called propositional or temporal transition system.

Definition 7: Propositional Transition System (PTS)
A Propositional Transition system is a Triple < T, L, C> where
T is fully transition system with set of transitions defined on S
L : S → $2^{Prop}$ with *Prop* (definition 2) is fixed set of atomic propositions is a state description which assigns to every state the set of atomic propositions true at S and
C is the set of configuration mapping points defined over Prop set and Functions over the resources of patterns used in the system.

PTS is transition system defined over attack pattern system with set of Propositions, Computable Functions and configuration points. PTS is specified with extended finite state machine (EFSM)

Definition 8: Pre Condition and Post Condition Set of PTS
Pre Condition Set is the set of conditions on variables used on PTS like Propositions, computable functions which are mapped using Configuration mapping points to construct the PTS required without any effect of path execution or Trace on the final PTS.

Pre Condition Set = {< Pi, Fj>} where Pi is Proposition variables and F is the function variables} and each Pi and Fj is mapped and paired against Configuration Mapping Points

Post Condition Set is the set of variables and resources remained where the output of the PTS is accepted by the final state and the final state is represented with decision variable. The decision variable is mapped with the two types of results to form the PTS.

Post Condition Set = {< Pi, Fj>} where Pi is Proposition variables and F is the function variables remained and the result of the PTS will stop within the scenario of the security context where each Pi & Fj is mapped and paired against Configuration Mapping Points

The first result is the finite state machine execution on valid path which is continued from the initial state to represent further execution in sequence where the resource available from the first computation is used as one computing resource for the next execution. The second result is the successful execution of the PTS on Valid Path without any resource as remaining for further computation with result stops and the sequence stops there with this computation.

That is post condition set of the first pattern in 'S' will be set to precondition set of the next pattern in 'S' if some propositions and functions remains in the post condition set after execution of the first pattern and the execution continues until post condition set is null.

Definition 9: Paths in Transition system
A Path in a transition system *T* is sequence of states and transitions represented in terms of propositions from the set *Prop* involving computation of functions which transform every state in to its successor satisfying the following conditions

1. S0 with start state containing valid *Prop* set
2. For each transition i relate proposition j from *Prop* set through configuration mapping points

S0 → S1 → S2… with a0, a1… actions represented with configuration mapping points and the path is said to be rooted as S0. The Path ∏ consisting of n transitions is said to have a length n denoted |∏| = n. The path is either maximal or Minimal. If the path terminates at a state t1 where further Configuration mapping point pair does not exists to make a transition from t1 then the path is said to be valid with proper termination otherwise the path is invalid. The path is either valid maximal or valid minimal. Maximal Path contains set of all states where the propositions are mapped to Functions to execute configuration mapping point pairs uniformly or equivalently with proper mapping of propositions to states against each pattern subject otherwise the path is minimal.

Definition 10: Reachable State of PTS
A state t is reachable from a state s in transition system *T* if there is a path in *T* leading from s to t with respect to one of the following two conditions

1. Either the state s satisfies a proposition from *Prop* on transition from s to t i.e. s → t on Pi
2. Or not

The set of all states in *T* reachable from s is denoted by Post $\begin{bmatrix} * \\ T \end{bmatrix}$ (s). We denote the successive states of the path by $\prod(0), \prod(1), \prod(2)...$ against each proposition maps the transition $\begin{bmatrix} p1, p2, p3... \\ \rightarrow\rightarrow\rightarrow... \end{bmatrix}$ i.e. p1, p2, p3… at $\prod(0), \prod(1), \prod(2)...$ respectively.

Definition 11: Computation or Trace with Configuration Mapping Points
A Computation or Trace in a PTS <T, L, C> is a (finite or infinite) sequence of state descriptions and respective actins along a path represented with configuration mapping points paired with computable functions and propositions

L (S0) [a0/c] → L (S1) [a1/c] → L (S2) → … Where each c is one of configuration mapping point where Proposition is mapped to the function. This Trace is represented with the help of final Transition Proposition System built along with Predicate Logic expression.

Definition 12: Acceptance or Path Execution
Acceptance or Path Execution defined over PTS *T* over some Computation or Trace is set of all states reachable from initial state S0 over S1, S2 … to t such that the path from s to t is valid (Valid Path).

## 3. SYSTEM VERIFICATION

The system is modeled to verify the desired properties of the secure system of attack patterns. The first basic rule and specification apply to the entire and whole system in specifying the security scenario as defined below

### 3.1 Specification – Main Scenario

Axiom 1: Any application scenario containing security context is mapped to relevant secure system of attack patterns. By the definition of attack pattern any pattern is defined with problem solution approach where the solution maps to a security context. For example Brute force attack results in creating the authentication scenario within any application.

Axiom 2: The solution to any pattern represents some security context

By the definition of pattern every pattern is represented by the solution which is shown by mechanism how attacker constructs the original attack and this is rearranged and restructured with mechanism of attack in the pattern definition.

Theorem: Let 'S' be a secure system of attack patterns, then applying attack patterns to security context yields a system of principles to determine desired results (metrics) with suitable values

Proof: The system is established which is treated as a simple relation on abstract set attack patterns. [Axiom 1 – every security context is mapped to pattern system]
The relation is represented as S subset of X*Y where S is the relation on the abstract sets of attack patterns, X be the inputs of attack patterns and Y be the output functions. The system establishes a functional relationship as follows
[By Axiom 2- the solution to pattern contain security context]
S: f(X) → Y with X and Y as input output pair

The system is proved by mathematical induction
Assume that the system initially contains an attack pattern with required security scenario.

Basis: The assumption is correct by the definition of attack pattern (containing Scenario graph which is equal to security scenario). [By Axiom 1]
Any attack pattern is basically represented with attack graph or scenario graph showing the solution to the attack if an attacker performs as it is [by axiom 2]
Induction: The system establishes a function with generated value for case n as follows
The general definition is defined as follows

F(X1, X2, X3, Xn) = Y1, Y2 … Ym
Where Xi and Yj are $i^{th}$ function and $j^{th}$ output value applied for the input
F (Xi) = Yj with X is sequence of applying attack patterns as per the rules so there exists a value Y for X

If a new case scenarios is applied either the solution to the pattern yields same or new solution with new value
So f (Xm) yields a value Yn, as the assumption of Attack pattern definition
SCENARIO: The assumption that the attack pattern yields a system of output values (metrics) with suitable values is represented with predefined set of variables, functions and remaining parameters defined in the system model as follows

$$\forall Si \,\&\, Oj \exists Mm, Nn, Cq, Ux$$

**Such that**

$$S = \Im(Pi, Fj, Ck) \Rightarrow Dt$$

Where Propositional transition system is constructed with extended finite state machine using pattern prerequisites supported from all parameters of the attack pattern system and the basic state machine is constructed as follows.
The initial state machine contains all possible prerequisites to execute pattern with the help of target system, software and environment which are exhibited using propositions to show the behavior of the pattern system. The resources required for the pattern system are confirmed with computable functions defined in the model. The method of attack including the context is established using Injection vector as per the definition made in the model. All the computable functions are defined to map the basic resources required by the pattern within the context defined for that pattern. The solution of executing the attack is compared with the finite state machine built from the model. The successful execution of state machine means that the pattern accomplishes the desired behavior what is supposed to exhibit in secure system or security scenario.

## 3.2 Specifications - Other Scenarios

The following specifications present the rules in applying the security scenario to the set of attack patterns. The rules are framed to show the behavior of the pattern to exhibit the sequential nature of the pattern execution. As the behavior takes execution in the form of extended finite state machine and accordingly the machine is converted to propositional Transition system.

Further to these specifications the state machine properties are comprehensive within the pattern behavior showed as propositions. Some more specifications are made against the behavior the propositions paired with computable functions used to derive Configuration mapping points. Specifications are made against Computable functions, code Components, security scenario, predicates & Propositions, Configuration mapping points, pattern solution, resource variables, extended finite state machine

Axiom 3: Attack pattern requires one or more resources in the problem solution scenario to apply the attack mechanism in the relevant scenario

Based on the properties of attack pattern, one of the main property for any pattern is resource consumption, whether the resource is directly available or not, If the resource consumption is primary the pattern requires that particular one otherwise the solution itself becomes resource for executing another pattern mechanism.

By the definition of Computable Function Set- there exists one or more resources to be available for the mechanism to be applied in the problem solution scenario. The model presents this resource consumption to map with one of the pre-defined resources as function variables and model assumes number of functions with associated value as weight-age against each of the category.

Theorem: Let 'S' be a secure system of attack patterns then any pattern from the set is mapped to one or more resource variables or computable functions

Proof:

By axiom 3 there exists a set of resources for pattern to implement solution, in the security context, if pattern consumes resources then pattern maps these resources to the defined computable function [ By axiom 3] So

$$\text{Pattern Pi} \rightarrow \bigcup iVjRk \Leftrightarrow \text{PTS <T, L, C>}$$

There exists Variable j and resource k to represent pattern using Propositional Transition system represented by <T, L, C> similarly the reverse is true as PTS is formed from pattern attributes only.

The construction of Proposition transition system includes mapping of function variables with propositions represented at each state of the PTS. As the path executes in the machine configuration mapping points are generated and functions variables are paired with propositions to verify the pattern properties.

Specification 3: Application Development is simulated as executing 'n' components either sequentially or non sequential manner by establishing propositional representation

using predefined operators on the set of Propositional Logic associated to the computable Functions defined over Secure system of attack patterns

Axiom 4: If S is secure system of attack patterns then there exists equivalent proposition set, computable function set and configuration mapping point set which results in executing proposition transition system representing some security scenario

By the definition of 'S' S contains Prop set, Function set and CP set then the result of executing 'S' results in decision variable associated to Accept or reject from an equivalent transition system.

If an equivalent PTS exists for 'S' then it is shown with an equivalent finite state machine

By axiom 1 & 2 each PTS is equal to some security context, So by the definition of 'S' there subsists proposition set, computable set and configuration mapping point set.

Theorem: Let S is secure system of attack patterns then there exists a Propositional transition system equal to some set of patterns from 'S' executing the Security Scenario with sequential execution.

Proof: Let 'S' is secure system of attack patterns then
there exists a proposition set Prop, function set F and Configuration Mapping Point CP, as the set Prop is not null it may contain some set of propositions [By axiom 4]
then let Prop = { Pi / i > 0} and each of resource variable maps the pattern characteristic that represent some function defined over F

let F = {fj / j > 0} As there exists's' for each CP from S the proposition pi is associated to some function Fj.

So for all CP's the pair of proposition and function represent some pattern from the set's'.

Let T be the PTS equivalent to 's' containing defined prop, F and CP so there is some output associated to PTS.

$$S = \Im(Pi, Fj, Ck) \Rightarrow Dt$$

If the output is yes all the propositions are equally and sufficiently mapped to associated function variables. Then all the patterns from set are executed representing the component. If the output is 'no' there are some propositions which are not mapped to any of the remaining function variables. So the sequential behavior of the component is not possible from the set. As per assumption each pattern from the set of pattern associates some code component or block as per the design specification.

SCENARIO: Proposition Representation

$$C \rightarrow C1 \vee C2 \vee C3....Cn \qquad \text{Or}$$
$$C \rightarrow C1 \wedge C2 \wedge C3....Cn$$

Where $C_i$ is component 'i' and C is super component that takes input independent of size, functionality. First one represents non sequential manner where as the second one represents sequential.

As each component is represented with pattern or set of patterns then the existence of secure system of attack patterns 's' is possible and any system can be represented as follows

Each Ci is executed with set of patterns beginning with single pattern followed by multiple patterns. The pattern(s) execution is shown with the help of Proposition Transition system containing sequence of components Ci in the application development finally facilitated with equivalent PTS.

Specification 4: When a component takes or inputs an attack pattern it is isolated completely with preferred execution until it outputs the desired one

Axiom 5: If 'S' be a secure system of attack patterns then each pattern from set s executes with its own set of propositions, computable functions associated to that pattern

By the definition of S each pattern associate to prop set, function set and configuration mapping set [By axiom 4]

Then the pattern execution yields equivalent pre condition and post condition set from Prop set. If the post condition set is not null the post condition set is equal to next pattern from set 'S' if post condition set is null then pattern execution is completed with Prop set

So the result of the state machine is Accept if they produce CP's [By the definition of S]

Scenario

$Ci \rightarrow \sum_{i=1}^{n} Pi = Dj$ And Pi $\notin$ Ck for other component k

and there exists some output decision function Dj. As each of the components combines one or more patterns depending on the scenario considered, the patterns execute in the sequence such that the first one positioned in the component takes first execution followed by the subsequent patterns exhibiting the isolation of patterns observing the property of sequential nature of the patterns given to that component.

Specification 5: Pattern solution applying to a component is represented by function called Computable function (CG), Proposition and the pair is represented using Configuration Mapping Points CP

Theorem: Let 'S' is secure system of attack patterns then a pattern execution from S is shown with the help of Configuration Mapping Points

Proof: As S is containing set of patterns each pattern is executed independently from other pattern [By axiom 5]

So each pattern execution takes propositions from Prop set, functions from Function set and finally associates pairing of each proposition with function.

As 'S' is the system then there exists output variable in terms of accept or reject from finite state machine.

If the output is yes the pattern execution is successful satisfying the PTS and if the output is reject the pattern is not executing with in 's'.

So for the derived PTS an equivalent Finite state machine exists giving the mapping of propositions and functions results in Configuration mapping points [By Axiom 1, 2 and 4].

Finally pattern execution is shown with the help of Configuration mapping points if's' exhibits desired properties [By specification 1, 2, 3 and 4]

Scenario: Pattern Pi $\rightarrow f(CP)$ for some <*Prop*, F[j]> derived from the pre defined set CP.

The application development is simulated in terms of security scenario containing components with set of patterns. As per the details in the specification 4 the execution is represented where each component is containing set of patterns. When each property of the pattern is mapped to function, all the functions which are relating to this pattern form the Computable function (CG). Subsequently the execution of state machine and transition system is built in finding the Configuration Mapping Points

Specification 6: Attack Patterns apply the problem solution paradigm of design in destructive context. This assumption uses the negation operator for all its computations to map the solution with pattern.

Axiom 6: If 'S' is the secure system of attack patterns the execution of 'S' shows the propositions mapping to the properties of pattern i.e. pattern behavior is expressed in terms of associated propositions or predicates

As's' takes one or more pattern each pattern behavior is mapped with set of propositions where each proposition map an individual property of that pattern [by the pattern definition if resource is required it is set to true for that predicate value otherwise false]

The resource or property mapping is expressed with destructive mechanism showing as negation assuming that the property is not true for the pattern.

Scenario: Pattern Pi $\rightarrow \neg(pi)$ for all propositions i. That is Pattern can be represented with proposition containing negation operator. Either all or some of the pattern characteristic can be represented using negation operator.

For example Pattern Pk contains the proposition set {p1, p2, p3… pn} then the proposition formula for the entire pattern can be represented using negation operator. The significance behind this proof is in mentioning whether the resource is available or not. For example with the proposition P1: 'Connection establishment', the remaining propositions will be set to true even though the connection is not established. So the destructive nature is applied in the context of availability of resources to implement the solution. If all the resources applied to the attack are not shown in real environment the attack scenario is constructed in destructive manner.

Specification 7: Pattern has a solution or Pattern contains problem solution approach

Axiom 7: Let S is secure system of attack patterns then any pattern from 'S' shows its problem solution approach for the equivalent PTS and exists for the scenario shown with desired solution

By the definition of 'S' for each of the attack pattern system within secure system of 'S'

$$\forall Si \;\&\; Oj \exists Mm, Nn, Cq, Ux$$

$$S = \Im(Pi, Fj, Ck) \Rightarrow Dt$$

this yields that every pattern has solution derived from the secure system.

Specification 8: Solution can be implemented with multiple scenarios.

Axiom 8: IF S is secure system the associated PTS and its execution is shown with the same path in the final finite state machine. By the definition of 'S' there exists an equivalent PTS giving desired result of accept or reject. [By axiom 1& 4]

If there exists PTS the equivalent state machine show the path [by the definition of Path Execution]

Theorem: Let 'S' is secure system, then the execution of patterns from 'S' is mapped to the execution of patterns in any order i.e. the order of patterns execution shows multiple scenarios where each scenario takes patterns from the same set 'S'

Proof: As 'S' contains pattern set each pattern is mapped with set of configuration mapping points [By specification 5]. For each Point from CP set a proposition is paired with computable function [Axiom 4, 5 and 6] there exists PTS for the pattern execution which results in 'accept' or 'reject'.

By axiom 8 the execution is shown with equivalent path from PTS. By axiom 1 each execution is mapped to a security scenario of the context.

In case if the patterns are not in order each pattern execution is isolated separating that execution in terms of Pre condition set and post condition set. Then for each path in PTS from the same set of patterns from 'S' there exists multiple paths showing the same result i.e. if the first path is accept then all the remaining paths also accept resulting different scenarios with the same result.

Scenario: From the definition of secure system of attack patterns pattern solution can be represented and defined for all possible values of Patterns, Functions and computations.

$$S = \Im(Pi, Fj, Ck) \Rightarrow Dt$$

for all values of i, j and k the system can be implemented with multiple scenarios.

Specification 9: Attack Patterns uses the resources or variables required to implement solution

Axiom 9: Let 'S' is secure system. Then each pattern is shown in the PTS execution in terms of pattern properties against Resource Variables. In order to construct secure system of attack pattern system each pattern uses the resources defined as per the model and variables of attack patterns to implement the solution [By the Definition of 'S', F, and CP's]

I.e. $\exists Rm, An$ for each resource type m and attribute n of attack pattern to implement the solution with output to accept or rejecting the system

$$S = \Im(Pi, Fj, Ck) \Rightarrow Dt$$

The pattern system is combination of patterns, pattern parameters, computable functions, resource variables and configuration mapping points. As the execution of each path starts at initial state the propositions are mapped to resource variables or computable functions to generate Configuration mapping points. So the trace or execution of the state machine is influenced by the mapping of resource variables within the solution.

Specification 10: The initial state of pattern generation contains default values of the essential resources within the context of security scenario

Axiom 10: Let S is secure system of attack patterns, then the equivalent PTS starts execution from initial state

AS per the definition of 'S' S Contains Prop set, Functions and patterns, when the first pattern from set 'S' starts execution the necessary propositions might have been already allocated to the first state [since this is the first one in 'S']

As soon the execution starts the propositions are mapped to functions with respective resource variable conditions to derive mapping points. So the PTS starts from the initial state only to support 'S'.

To implement the solution for pattern system the initial state of the pattern contains default values defined against each resource and variable that pattern contains.

I.e. for all initial values of Ri with resource type i the secure system will be developed as

$$S = \Im(Pi, Fj, Ck) \Rightarrow Dt$$

As the solution starts in the initial state of the Finite state machine in constructing the Proposition transition system, the model assumes that the pre defined set of propositions are initialized in that state in dividing the propositions as per the categories of Prop-1, Prop-2 and Prop-3 sets so that execution or Trace of the state machine starts from that configuration only.

Specification 11: Pattern Solution is valid state containing of final state of attack scenario which is of existing scenario of security context or new

Axiom 11 Let 'S' be System then the execution of PTS with 'S' results in equivalent finite state machine with accept in the final state

The final state of the pattern solution is represented with a decision variable accept or reject to derive the validity of the pattern. If result is accept the pattern is validated against all possible resources and computation functions mapped against resource identified otherwise the pattern is not validated and expecting to result in the new attack pattern for the system.

As per the model specification the Pre Condition Set and Post Condition set are set to true in the initial state to begin the execution or Trace. Once the path execution completes and reaches the acceptance state, the remaining set of propositions are mapped and compared to precondition set of next scenario. Once the Post Condition set is equaled with Pre condition set the execution of the scenario with post condition set is made equal to pre condition set and the execution continues to progress as per the state machine rules and actions. [By axiom 4, 5 and 9]

So the pattern solution is valid state containing of final state of attack scenario which is of existing scenario otherwise the next scenario in the sequence representing the next pattern in the security system of attack patterns.

Specification 12: Is every initial state starts with value of Injection Vector (IJ)

Axiom 12: If 'S' is secure System then the consumption of resource variables is done with the help of PTS only

The Injection Vector contains the values for all environment variables of the host/target area, number of machines, network size and other parameters. The initial state of State machine holds these initial values which can be used to derive necessary output values

By the definition of Computable Functions, Each variable is representing pattern property only which is taken from Injection Vector element.

$\exists Ei \Rightarrow Dj$ For all environment variables i and Output variables j

The pattern solution is associated with computable functions and resource variables as mentioned in the model. Each of the variables is mapped and derived based on the categories that are made in the system. The model defines 9 first level categories of functions further sub divided in to 45 secondary level functions. Each of these functions is defined with resource variable which are with propositions to get configuration mapping points.

Specification 13: Pattern solution constructs the target system with values for Target system, size of network and security context value

Axiom 13: If 'S' is secure system then the equivalent PTS is shown with execution containing pattern solution exhibited with the help of security context

The pattern solution can be represented by another variable called target system to construct pattern solution. Target System is formed by combining host and target machine, network type and network size and number of machines, number of resource variables and security context with predefined values.

I.e. for all initial values of set of host/target machine, network size, number of resource type and other parameter values of {i, j, k} the secure system will be represented as

$$S = \Im(Pi, Fj, Ck) \Rightarrow Dt$$

By Axiom 12 every 'S' exhibits solution containing all possible values of solution scenario. So if pattern solution contains PTS, the equivalent execution displays the properties of pattern only.

By Axiom 8, 9 1nd 10 'S' with PTS exhibits the security context only with respective values of pattern elements

As mentioned in the previous specifications the pattern solution constructs the target system with pre-defined value set with the help of computable functions, resource variables and configuration mapping points. So the security context is required for pattern execution.

Specification 14: Pattern solution can be represented with computing values of Configuration mapping points

Axiom 14: Let 'S' be the secure system then pattern solution contains configuration mapping points

By the definition of 'S' and configuration mapping points itself S' contains prop and function. So every system of pattern solution applies the Environment variables to provide pattern solution towards desired values with CP as follows <*Prop*, F[j]> where Prop is the set of propositions and F[j] is the set of computable functions.

Specification 15: Each component with respect to pattern set is independent of other components in the system

Axiom 15: Let 'S' is secure system, then the pattern execution is independent from other pattern with respective to other patterns

This specification disintegrates the components with respect to pattern set. That is pattern mapping to this component is isolated from other components. The pattern execution is applied independently irrespective of the others. The pattern execution is shown with the help of Proposition Transition system. When multiple patterns are executed the patterns are applied in sequence based on the mechanism of attack which indicates the way attack is constructed. So the next pattern is verified in terms of the path execution once the first one's execution is completed.

## 4. RESULTS AND OBSERVATIONS

As the data related to attack patterns is currently available in documentation called CAPEC list maintained by MITRE – US Home land organization which contains the complete list of attack patterns that are primarily categorized on mechanism of attack. Essentially we took most of the data elements from the CAPEC List and created a database of attack patterns using a software tool. [6] As part of the thesis work we created attack pattern database which is organized in to primary schema and secondary schema as discussed in the prototype model. The database facilitates the required data to the specification tool and is used to create the elements defined in the system. The main objective of the work is to create security metrics program based on attack patterns. As part of the work we extended the concept to generate the metrics in terms of Points and the process of generation is verified using several case studies. [7] We have defined 4 categories of Templates to hold the data required as per the secure system of attack patterns and the paper contains further details.

### 4.1 Case Study – Single Pattern

In this section we show the relationship between attack patterns and propositions for the security scenario containing individual patterns. As described in the model any resource/variable/requirement that is needed/desired/ required/ preferred/considered necessary to implement the pattern solution is considered as proposition or predicate so that the level of hypothesis to establish the theoretical base made in the model specification is correct and verified.

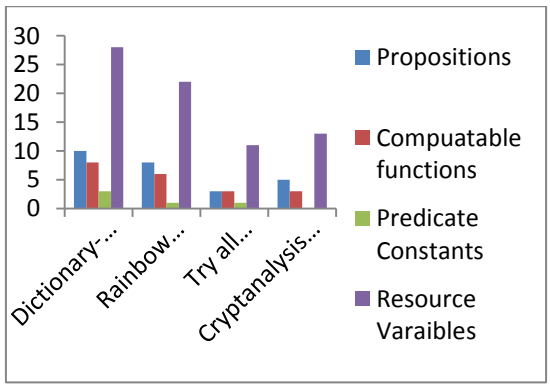


**Figure 1 Number of Predicates, Functions for 4 Patterns**

The graph shown in Figure 1 describes the number of Propositions, number of computable functions and the resource variables related to four attack patterns under Probabilistic Technique attack mechanism. Out of four types of patterns under the said mechanism, Brute force attack pattern is considered which is further divided in to Encryption and Password brute forcing. In Password brute forcing three types viz., Dictionary based, Rainbow password and Try all passwords attacks are specified and under Encryption brute forcing - Cryptanalysis attack is mentioned. As shown in the graph the pattern "Dictionary based password attack" requires more resources than the remaining three patterns. As per the document the solution to the dictionary based password attack is more complex in terms implementation view which agrees with the graph.

## 5. COCLUSION AND FUTURE WORK

In this paper we proposed a frame work containing system of attack patterns which is built along with computable functions, configuration mapping points and propositional transition system. The verification of attack pattern system is shown with respective theorems and lemmas. The results are shown against number of propositions, resource variables, proposition constants and computable functions with respect to four attack patterns namely dictionary based pattern and related patterns. In future we extend the work towards the final result to calculate the proposed security metric called Cascadability Points.

## 6. ACKNOWLEDGMENTS

Our sincere thanks to US Homeland CAPEC organization for giving us opportunity to use the available pattern catalogue and allowed us to create pattern database from the available list of patterns. Our thanks to Build- in- Security a security consortium to allow us refer the details of vulnerability database available from their web site. Lastly my sincere thanks to my guide Dr. LSS Reddy in encouraging me to complete the research work.

## 7. REFERENCES

[1] W Thomas, "Automata theory and Infinite Transition Systems", Lecture notes, University of Liege, DAAD Procope Project, May 2006, pp 1-12

[2] E M Clarke, O Grumbreg, "Research on Automatic Verification of Finite-State Concurrent Systems", Annual Revue Computing Science, vol. 2, 1987, pp 269–290

[3] E A Emerson, "Temporal and Modal Logic", Formal Models and Semantics, Hand-book of Theoretical Computer Science, Jan van Leeuwen Editor, Elsevier Publications, 1990, pp 995– 1072

[4] D Bert, F Cave, "Construction of Finite Labeled Transition systems from Abstract systems", Research Project, VERDON, 1998, pp 1-10

[5] D Dams, "Abstract Interpretation and Partition Refinement for Model Checking", PhD thesis, Technical University of Eindhoven, Netherlands, 1996

[6] K Raja Sekhar, Dr LSS Reddy, UJ Kameswari, "A Prototype Model to Generate Application Security Metric using Attack Patterns", Proceedings of IEEE Conference IACC'09, March 2009, Patiala, pp 143-147

[7] K Raja Sekhar, Dr LSS Reddy, UJ Kameswari, "Templates to derive Security Metrics based on Attack Patterns", Proceedings of 3rd International Conference on Software engineering", Indore, September 2012