International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2017 IJSRCSEIT | Volume 2 | Issue 6 | ISSN : 2456-3307

A Self-Executing Study of Arranging Scribble for Security Principle

Dr. G. Nagalakshmi[1],Vidadhala Kartheek[2]

[1]Head of the Department Computer Science and Engineering, siddartha Institute of Science and Technology, Puttur, Karnataka, India

[2]Student of Software Engineering, siddartha Institute of Science and Technology, puttur, Karnataka, India

ABSTRACT

We gift Script worker (SITAR), a technique to automatically repair unusable low-level check scripts. instrument uses reverse engineering techniques to create Associate in Nursing abstract check for each script, maps it to Associate in Nursing annotated event-flow graph (EFG), uses repairing transformations and human input to repair the check, and synthesizes a replacement "repaired" check script. throughout this technique, instrument together repairs the relation to the user interface objects utilised within the checkpoints yielding a final check script which will be dead automatically to validate the revised computer code package. instrument amortizes the worth of human intervention across multiple scripts by accumulating the human info as annotations on the EFG. to increase computer code package responsibility and security. New cost-effective tools for computer code package quality assurance unit of measurement needed thus this, paper presents associate degree automatic check generation technique, referred to as Model-based Integration and System check Automation (MISTA), for integrated sensible and security testing of computer code package systems. Given a Model-Implementation Description (MID) specification, MISTA generates check code which will be dead instantly with the implementation beneath check. the center specification uses a high-level Petri internet to capture every control- and data-related wants for sensible testing, access management testing, or penetration testing with threat models. once generating check cases from the check model in line with a given criterion, MISTA converts the check cases into practicable check code by mapping model- level elements into implementation-level constructs. MISTA has enforced check generators for diverse check coverage criteria of check models, code generators for diverse programming and scripting languages, and check execution environments like Java, C, C++, C#, HTML-Selenium IDE, and golem Framework. MISTA has been applied to the sensible and security testing of various real-world computer code package systems.

Keyword: Functional Testing, Model-Based Testing, Petri Nets, Security Testing, Computer Code Assurance.

I. INTRODUCTION

The widespread application of net and mobile computing has significantly increased our dependence on software- enabled systems. This dependence raises very important problems regarding coding system reliability and security as a results of a coding system failure can end in fatal consequences. However, coding system testing could also be a labor-intensive activity, that often accounts for 5 hundredth or further of the pc code development costs. to reinforce testing productivity and reduce costs, it's extraordinarily fascinating to automatize check generation and execution. Automation permits further check cycles due to repeatable checks and extra frequent check runs. It in addition facilitates quick, economical verification of demand changes and bug fixes, and minimizes human errors. In this, we've got a bent to gift a tool-supported technique mentioned as Model-based Integration and System check Automation (MISTA),1 for integrated testing of system functions, access management policies, and security threats. It uses Predicate-Transition (PrT) nets as academic degree expansive formalism for building helpful and security check models. PrT nets unit of measurement high-level Petri nets, a well-studied formal methodology for

modeling and verification of coding system systems [3]-[7]. previous work has in addition incontestable that PrT nets unit of measurement capable of specifying access management policies and security threats [8]-[10]. as a results of check models such by PrT nets can capture every data and management flows of check wants, MISTA can generate complete model-based check cases, yet as specific check inputs and check oracles (expected results). Note that model-based check cases do not appear to be nonetheless possible with the SUT as a results of check models unit of measurement abstract descriptions of SUT's behaviors. MISTA provides academic degree expansive technique for describing the relations between the model-level elements so|and so} the implementation-level constructs at intervals the target language or check surroundings thus on automatically work on the model-level tests into possible code.

II. RELATED WORK

An excellent treatment of the realities shut test-suite evolution and maintenence [28], they discuss varied realistic use cases at intervals that take a glance at cases unit of measurement extra, removed, and refactored in follow. They to boot means, wholly completely different from previous cases, take a glance at repair may be a heap of advanced and hard-to-automate and existing test-repair techniques that concentrate on assertions is additionally unsuitable in follow. This motivates North yank nation to repair real take a glance at scripts that involves differing types of changes and wishes domain knowledge to repair. we tend to tend to reinforce the wide used EFG model by storing human actions as new nodes/edges/labels at intervals the model to accelerate the semi-automatic repair methodology.

III. MODEL-IMPLEMENTATION

A. PrT Nets for Test Modeling

Multiple initial markings (states) area unit typically associated with identical net structure. Suppose is AN initial marking, and Mk0(p) is that the set of tokens residing in place P. A token in p is also a tuple of ground terms<X1,…..,Xn>. we have a tendency to tend to together denote it as p(X1,…..,Xn). . For a zero-argument token < >in p , we have a tendency to tend to simply denote it as p. The tokens in AN initial marking represent take a glance at info or system settings (e.g., decisions and preferences) or every. in AN extremely go-cart system, as AN example, token product (VGN-Z17) and token quantity (3) represent the merchandise VGN-Z17 and thus the number 3. A transition might even be associated with a list of variables as formal parameters. These variables sometimes appear inside the connected arc labels. Fig. one shows an easy PrT net, where holding, clear, on, and handempty unit places (circles); and stack(x, y) is also a transition (a rectangle). The guard condition of stack(x, y) is x!=y (it is encircled in brackets in Fig. 1). AN arrow (e.g., from holding to stack) represents a regular arc; a line part with atiny low circle (e.g., from handempty to stack) represents AN matter arc.

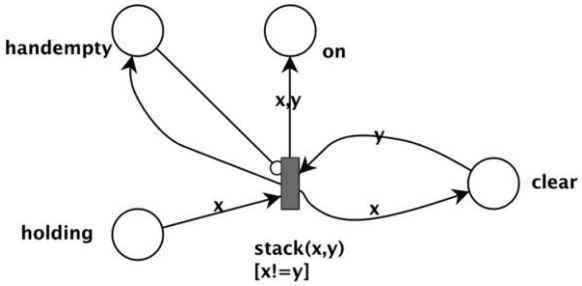

Figure 1. A simple PrT net.

B. Model-Implementation Mapping

A MIM specification could also be a 7-tuple , where the weather area unit as follows.

1) ID is that the identity of the SUT take a glance ated against the take a look at model.

2)f0:OM OI OI is that the article operate that maps the objects at intervals the take a glance at model to the objects at intervals the SUT. Given Associate in Nursing object x at intervals the take a glance at model f0(x), is Associate in Nursing object at intervals the SUT.

3) fc:T CODEI is that the part (or method) mapping operate that maps transitions (component calls) at intervals the PrT internet to code blocks (test operations) at intervals the SUT.

4) fa:P CODEI is that the accessor operate that maps the places at intervals the PrT internet to code blocks (called accessor) at intervals the SUT. Associate in Nursing accessor is sometimes a sequence of assertions that scan and check system states.

5) fm:P CODEI is that the mutator operate that maps the places at intervals the PrT internet to code blocks (called mutators) at intervals the SUT. A mutator could also be a chunk of code which is able to modification system states.

6) could also be an inventory of places at intervals the PrT internet that area unit implemented as system settings at intervals the SUT. These places area unit stated as setting predicates.
7) h is that the helper code operate that defines user-provided code to be clathrate at intervals the take a glance at code.

IV. GENERATING MODEL-BASED TESTS

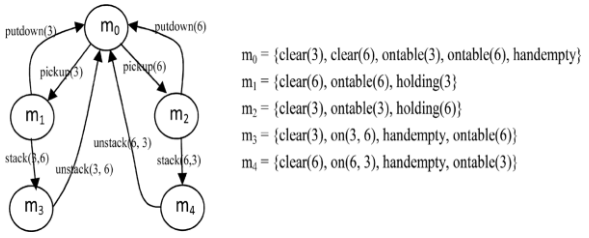

Figure 2. A reachability graph.

Algorithm 1 Generate tests for reachability coverage with robustness tests.

Input: PrT net
Output: transition tree with strength tests
Declare: root, newNode, currentNode area unit nodes queue could also be a queue of nodescleanSubstitutionsandrobustnessSubsequences
area unit lists of substitutions newMarking could also be a marking
1. begin
2. initialization: queue ø ; root root manufacture a latest node
3. for each initial marking mkoϵ mo , do
4. manufacture the initial state node as a baby of the premise
5. add the node into queue
6.end for
7. whereas queue ≠ ø do
8. currentNode initial node in queue
9. for each transition t ϵT, do
10. cleanSubstitutions all substitutions that make t firable beneath currentNode.marking
11. for every Θ ϵ cleanSubstitutions, do
12. newMarking the marking of firing t with Θ under currentNode.marking
13. newNode.parent currentNode
14. newNode.marking newMarking
15. newNode.transitiont
16. newNode.substitution Θ
17. newNode.isRobustness false
18. add newNode to currentNode.children
19. if newMarking has not occurred within the tree
20. add newNode to queue
21. end if
22. end for
23. robustnessSubstitututions substitutions that disablet beneath currentNode.marking
24. for every Θ ϵ robustnessSubstitututions, do
25. newNode.parent currentNode
26. newNode.marking currentNode.marking
27. newNode.transitiont
28. newNode.substitution Θ
29. newNode.isRobustness true
30. add newNode to currentNode.children
31. end for
32. end for
33. end while
34. come root
35. end

After data formatting, formula one initial creates a node for each initial marking, and adds the node to the queue for growth (lines 3-6). Then it takes a node from the queue for growth (line 8). for each transition, it finds all substitutions that modify the transition below the marking of this node (called clean substitutions, line 10), creating a successor node through the transition firing for each substitution (lines 12-18), and shot the new node into the queue for any growth if the state has not appeared before (line 19-21). Substitutions area unit computed through unification and backtracking techniques supported the definition of transition enabledness. A clean substitution for a transition is obtained by unifying the arc label of each input or substance place with the tokens throughout this place, and evaluating the guard condition (an substance arc indicates negation, though). once a substitution is obtained, backtracking is applied to the unification methodology until all clean substitutions area unit found. The generation of strength tests (lines 23-31) area unit attending to be mentioned below. although formula one follows the ultimate structure of tree generation and traversal, the computation of unpolluted and strength substitutions distinguishes MISTA from this work on testing with state machines. Computing clean and strength substitutions could also be a technique of finding actual parameters of variables to dynamically verify state transitions so as that complete take a glance at sequences could also be generated. formula one returns the premise of the transition tree so

as that the tree could also be traversed for take a glance at code generation (line 34). in associate extremely transition tree, each leaf node indicates a take a glance at sequence, starting from its corresponding initial state node to the leaf node. All the sequences generated from identical initial state represent a take a glance at suite. Therefore, a transition tree contains one or plenty of take a glance at suites. MISTA provides a GUI to look at transition trees.

V. GENERATING TEST CODE

Algorithm two below briefly describes but a check class for the whole transition tree is generated for associate object-oriented language (e.g., Java, C#, C++, and VB). First, it creates the header (e.g., package and import statements in Java) and conjointly the signature of the check class (lines 2-3). once the SUT is also a class or a cluster of classes, it to boot creates the declaration of associate instance variable whose kind is ID (lines 4-6). Then, for each initial state, it generates a setup technique to line the SUT to the given state by pattern the mutator operate (lines 7-17) (when there aren't any user-provided setup methods). Given a token p (a1,..,ak) in associate initial state, the formula transforms model-level objects ai to implementation-level objects f0a(i) , then calls the mutator operate fm (line 14). This approach is analogous for managing system settings under control sequences (line 25). for each check sequence retrieved from the tree, the formula generates a check technique (lines 20-37). The body of the check technique first invokes the corresponding setup technique (line 22), then for each call at intervals the sequence it configures the system settings for the choice (lines 24-26), issues the choice (line 27), and verifies oracle values of the choice (lines 28-33, see the definitions of oracle values). For part call tiΘi objects to implementation-level objects f0b(i) , then calls the part operate fc (line 27). The mapping of objects conjointly applies to the generation of assertions for oracles before the accessor operate syllable is utilized (lines twenty 9 and 32). The check technique to boot calls the teardown code if made public (line 35). finally check ways square measure completed, the check suite technique for each initial state is made to execute the alpha code if made public, invoke every take a look at technique, and perform the omega code if outlined (lines 38-40). Finally, the algorithmic program imports the user-defined code (line 41), and creates the most technique (line 42).

Algorithm 2 Generate test code in an object-oriented language (Java, C#, C++ or VB).

Input: transition tree root,MIM=<ID,f0,fc,fa,fm,fs,h>
Output: check code
Declare: initialStates could be a set of initial markings initState is associate degree initial marking leafNodes could be a set of leaf nodes
testSequences could be a set of check sequences
testSequence refers to 1 check sequence
1. begin
2. produce header consistent with h (e.g., package and import statement in Java)
3. produce the signature of check category consistent with ID and coverage criterion
4. if SUT could be a category or a cluster of categories
5. declare associate degree instance variable whose sort is ID (ID is that the entry class)
6. end if
7. initialStates notice all initial markings from the kid nodes of root
8. for every initState ϵ initialStates, do
9. if SUT could be a category or a cluster of categories
10. produce an announcement for the declared instance variable to reference a brand new object of ID
11. end if
12. produce the signature of a brand new setup technique definition
13. for every p ϵ P and every token <a1,..,ak>in place p in initState, do
14. produce fm(p(f0(a1),…,f0(ak))) within the technique body
15. end for
16. produce the closing a part of the setup technique
17. end for
18. leafNodes all leaf nodes by traversing the tree from root
19. checkSequences all test sequences consistent with leafNodes
20. for every testSequenceM0k[t1Θ1>M1k,…,[tnΘn>M0k ϵ testSequences, do
21. produce the signature of the check technique
22. decision the setup technique equivalent to the initial state
23. for (i=1 to n) do
24. for every input place p of ti such P ϵ ls and <a1,..,ak>ϵ Mki(p), do

25. produce system setting code produce fm(p(f0(a1),…,f0(ak))) for p(a1,..,ak)
26. end for
27. produce part decision code, fc(c(f0(b1),…,f0(bk))) , for tiΘi=c(b1,..,bk)
28. for every p(a1,..,ak) such <a1,..,ak>ϵ Mki(p), do
29. produce assertion fa(p(f0(a1),…,f0(ak))) For p(a1,..,ak)
30. end for
31. for every p(a1,..,ak) such <a1,..,ak>ϵ Mki-1(p), but <a1,..,ak> Mki(p) , do
32. produce assertion ! fa(p(f0(a1),…,f0(ak))) For p(a1,..,ak)
33. end for
34. end for
35. produce a decision to h(teardown) if outlined
36. produce the closing a part of the check technique
37. end for
38. for every initState ϵ initialStates, do
39. produce a check suite execution technique together with a decision to h(alpha) if outlined, a decision to every check technique generated for initState, and a decision to h(omega) if outlined
40. end for
41. import helper code h (teardown, local, etc.)
42. produce the most technique to incorporate a decision to the check suite execution technique for every initial state
43. end

VI. CONCLUSION

We have given a way for automated generation of practicable purposeful or security checks from a take a look at model in conjunction with the mapping from the modeling elements to the implementation constructs. Complete model-level checks additionally be|is also} computed as a results of the check model specifies every the management and also the data dependencies of take a look at targets. The mapping makes it attainable to rework the model-level tests into the practicable sort. varied case studies have incontestable that MISTA is economical and effective.

The main contribution of this paper could also be a completely unique technique for integrated model-based testing of system functions, access management policies, and security threats. The technique can generate practicable checks with relevancy a ramification of coverage criteria of take a look at models, delineate by PrT nets. It in addition supports type of programming languages (e.g., Java, C#,C++, VB), and check execution frameworks (e.g., JUnit, number thirty four IDE, and automaton Framework). thanks to the technique's protractile style, it's easy to introduce a innovative check generator, target language, or check execution atmosphere.

VII. REFERENCES

[1]. J. Zender, I.Schiefewrdecker, and P.Mosterman, Eds., Model-Based Testing for Embedded Syst.Boca Raton, FL, USA: CRC Press, 2011.

[2]. M.Utting and B.Legeard, Practical Model-Based Testing: A Tools Approach.San Francisco, CA, USA: Morgan Kaufmann, 2006.

[3]. H.J.Genrich, "Predicate/move nets," in Petri Nets: Central Models and Their Properties.New York, NY, USA: Springer, 1987, pp.207-247.

[4]. K.Jensen, Colored Petri Nets: Basic Concepts, Analysis Methods and Practical Use.New York, NY, USA: Springer-Verlag, 1992, vol.26.

[5]. T.Murata, "Petri nets: Properties, investigation and applications," Proc.IEEE, vol.77, no.4, pp.541-580, Apr.1989.

[6]. W.Reisig, "Petri nets and arithmetical particulars," Theoret.Comput.Sci., vol.80, pp.1-34, 1991.

[7]. D.Xu, "An instrument for mechanized test code era from abnormal state Petri nets," in Proc.32nd Int.Conf.Applicat.also, Theory of Petri Nets and Concurrency (Petri Nets 2011), LNCS 6709, Springer-Verlag, Berlin, Heidelberg, Germany, Newcastle, U.K., Jun.2011, pp.308-317.

[8]. D.Xu, L.Thomas, M.Kent, T.Mouelhi, and Y.Le Traon, "A model-based way to deal with mechanized testing of get to control strategies," in Proc.seventeenth ACM Symp.Get to Control Models and Technologies (SACMAT'12), Newark, NJ, USA, Jun.2012.

[9]. D.Xu, M.Tu, M.Sanford, L.Thomas, D.Woodraska, and W.Xu, "Mechanized security test era with formal danger models," IEEE Trans.Depend.Secure Comput., vol.9, no.4, pp.525-539, Jul./Aug.2012.

[10]. D.Xu and K.E.Nygard, "Danger driven demonstrating and check of secure programming utilizing viewpoint situated Petri nets," IEEE Trans.Softw.Eng., vol.32, no.4, pp.265-278, Apr.2006.

[11]. D.Xu, J.Yin, Y.Deng, and J.Ding, "A formal building model for intelligent operator versatility," IEEE Trans.Softw.Eng., vol.29, no.1, pp.31-45, Jan.2003.
[12]. D.Xu, R.A.Volz, T.R.Ioerger, and J.Yen, "Displaying and examining multi-operator practices utilizing predicate/move nets," Int.J.Softw.Eng.Knowl.Eng., vol.13, no.1, pp.103-124, 2003.
[13]. N.J.Nilsson, Principles of Artificial Intelligence.San Francisco, CA, USA: Morgan Kaufmann, 1980.
[14]. R.V.Fastener, Testing Object-Oriented Systems: Models, Patterns, and Tools.Perusing, MA, USA: Addison-Wesley, 2000.
[15]. [Online].Accessible: http://www.magentocommerce.com
[16]. [Online].Accessible: http://www.zen-cart.com
[17]. Y.Jia and M.Harman, "An investigation and study of the improvement of change testing," IEEE Trans.Softw.Eng., vol.37, no.5, pp.649-678, 2010.
[18]. Y.L.Traon, T.Mouelhi, A.Pretschner, and B.Baudry, "Test-driven evaluation of get to control in heritage applications," in Proc.first IEEE Int.Conf.Programming, Testing, Verification and Validation (ICST'08), Norway, 2008, pp.238-247.
[19]. T.Mouelhi, F.Fleurey, B.Baudry, and Y.L.Traon, "A model-based system for security strategy detail, arrangement and testing," in Proc.ACM/IEEE eleventh Int.Conf.Show Driven Eng.Dialects and Syst.(MODELS'08), Toulouse, France, 2008.
[20]. [Online].Accessible: https://sites.google.com/site/servalteam/apparatuses/Mutax
[21]. OWASP.The Ten Most Critical Web Application Security Risks [Online].Accessible: http://www.owasp.org
[22]. M.Shafique and Y.Labiche, A deliberate survey of model based testing apparatus bolster Carleton Univ., 2010, Tech.Rep.SCE-10-04.
[23]. J.Jacky, M.Veanes, C.Campbell, and W.Schulte, Model-based Software Testing and Analysis with C#.Cambridge, U.K.: Cambridge Univ.Press, 2008.
[24]. H.S.Hong, Y.G.Kim, S.D.Cha, D.H.Bae, and H.Ural, "A test arrangement determination strategy for statecharts," J.Softw.Test., Verif., Rel., vol.10, no.4, pp.203-227, 2000.
[25]. L.Gallagher, J.Offutt, and T.Cincotta, "Coordination testing of objectoriented segments utilizing limited state machines," J.Softw.Test., Verif., Rel., vol.16, no.4, pp.215-266, 2006.
[26]. L.Gallagher and J.Offutt, "Test arrangement era for coordination testing of part programming," Comput.J., Advance Access, Nov.2007.
[27]. L.Briand, Y.Labiche, and Q.Lin, "Enhancing the scope criteria of UML state machines utilizing information stream investigation," J.Softw.Test., Verif., Rel., vol.20, no.3, pp.177-207, Sep.2010.
[28]. M.v.d.Bijl, Applied model-based testing: Automatically create, execute, and assess Tests, 2011 [Online].Accessible: fmt.cs.utwente.nl/dwftt2007/introductions/MacBij_MBT_POS_public.pdf
[29]. J.Offutt, S.Liu, A.Abdurazik, and P.Ammann, "Creating test information from state-based details," J.Softw.Test., Verif., Rel., vol.13, no.1, pp.25-53, 2003.
[30]. M.Gaudel, A.Denise, S.Gouraud, R.Lassaigne, J.Oudinet, and S.Peyronnet, "Scope one-sided arbitrary investigation of substantial models," in Proc.fourth ETAPS Workshop on Model Based Testing, 10 of Electron.Notes in Theoretical Comput.Sci., 2008, vol.220, pp.3-14.
[31]. C.Jard and T.Jéron, "TGV: Theory, standards and calculations: An instrument for the programmed amalgamation of conformance test cases for non-deterministic responsive frameworks," Int.J.Softw.Instruments Technol.Exchange (STTT), vol.7, no.4, pp.297-315, Aug.2005.
[32]. P.Pelliccione, H.Muccini, A.Bucchiarone, and F.Facchini, "TeStor: Deriving test successions from display based details," in Proc.eighth Int.SIGSOFT Symp.Part Based Software Eng.(CBSE'05),m LNCS 3489, 2005, pp.267-282.
[33]. S.Ali, L.Briand, M.J.Rehman, H.Asghar, M.Z.Z.Iqbal, and A.Nadeem, "A state-based way to deal with reconciliation testing in light of UML models," Inf.Softw.Technol., vol.49, no.11-12, pp.1087-1106, 2007.
[34]. A.Chander, D.Dhurjati, K.Sen, and D.Yu, "Ideal test input succession era for limited state models and pushdown frameworks," in Proc.2011 Int.Conf.Programming Testing, Verification and

Validation (ICST'11), Berlin, Germany, Mar.2011.
[35]. R.Ubar and M.Brik, "Multi-level test era and blame analysis for limited state machines," in Dependable Computing—EDCC-2 Second Eur.Trustworthy Computing Conf., LNCS 1150, Taormina, Italy, 1996, pp.264-281.
[36]. H.Zhu and X.He, "A philosophy for testing abnormal state Petri nets," Inf.Softw.Technol., vol.44, pp.473-489, 2002.
[37]. S.Barbey, D.Buchs, and C.Péraire, "A hypothesis of detail based testing for protest arranged programming," in Dependable Computing—EDCC-2 Second Eur.Tried and true Computing Conf., LNCS 1150, Taormina, Italy, 1996, pp.303-320.
[38]. L.Lucio, L.Pedro, and D.Buchs, "Self-loader experiment era from CO-OPN particulars," in Proc.Workshop Model-Based Testing and Object-Oriented Syst., 2006, pp.19-26.
[39]. L.Lucio, "SATEL—a test aim dialect for question situated details of receptive frameworks," Ph.D.paper, Université de Genève, Center Universitaire d'Informatique, Geneva, Switzerland, 2009.
[40]. J.Desel, A.Oberweis, T.Zimmer, and G.Zimmermann, "Approval of data framework models: Petri nets and experiment era," Proc.SMC'97, pp.3401-3406, 1997.
[41]. C.C.Wang, W.C.Pai, and D.- J.Chiang, "Utilizing Petri net model way to deal with question situated class testing," Proc.SMC'99, pp.824-828, Oct.1999.
[42]. A.Masood, R.Bhatti, A.Ghafoor, and A.Mathur, "Adaptable and powerful test era for part based get to control frameworks," IEEE Trans.Softw.Eng., vol.35, no.5, pp.654-668, 2009.
[43]. A.Masood, A.Ghafoor, and A.Mathur, "Conformance testing of worldly part based get to control frameworks," IEEE Trans.Depend.Secure Comput., vol.7, no.2, pp.144-158, 2010.
[44]. H.Hu and G.Ahn, "Empowering check and conformance testing for get to control show," in Proc.thirteenth ACM Symp.Get to Control Models and Technologies, 2008, pp.195-204.
[45]. W.Mallouli, J.M.Orset, A.Cavalli, N.Cuppens, and F.Cuppens, "A formal approach for testing security rules," in Proc.twelfth ACM Symp.Get to Control Models and Technologies, 2007, pp.127-132.
[46]. J.Jurjens, "Display based security testing utilizing UMLsec," Electron.Notes Theoret.Comput.Sci.(ENTCS), vol.220, no.1, pp.93-104, Dec.2008.
[47]. K.Li, L.Mounier, and R.Groz, "Test era from security approaches indicated in Or-BAC," in Proc.31st Comput.Programming and Applicat.Conf.(COMPSAC'07), 2007, pp.255-260.
[48]. A.Pretschner, Y.L.Traon, and T.Mouelhi, "Demonstrate based tests for get to control strategies," in Proc.first Int.Conf.Programming Testing Verification and Validation (ICST'08), Lillehamer, Norway, Apr.2008.
[49]. J.Julliand, P.A.Masson, and R.Tissot, "Producing security tests notwithstanding practical tests," in Proc.third Int.Workshop Automation of Software Test, 2008, pp.41-44.
[50]. H.Huang and H.Kirchner, "Formal determination and confirmation of secluded security arrangement in view of shaded Petri nets," IEEE Trans.Depend.Secure Comput., vol.8, no.6, pp.852-865, Nov./Dec.2011.
[51]. B.Shafiq, J.Joshi, and A.Ghafoor, "Petri-net based displaying for confirmation of RBAC approaches," Tech.Rep., Center for Education and Research in Information Assurance and Security, Purdue Univ., 2002.
[52]. Y.Deng, J.C.Wang, J.Tsai, and K.Beznosov, "An approach for displaying and examination of security framework designs," IEEE Trans.Information Data Eng., vol.15, no.5, pp.1099-1119, Sep.2003.
[53]. K.H.Mortensen, "Programmed code era strategy in light of hued Petri net models connected on a get to control framework," in Application and Theory of Petri Nets.New York, NY, USA: Springer-Verlag, 2000, pp.367-386.
[54]. K.Knorr, "Dynamic get to control through Petri net work processes," in Proc.sixteenth Annu.Conf.Comput.Security Applicat., 2000, pp.159-167.
[55]. A.Marback, H.Do, K.He, S.Kondamarri, and D.Xu, "A threat modelbased approach to security testing," in Software: Practice and Experience, Expanded Version of the AST'09Workshop Paper, Feb.2013, vol.43, pp.241-258.

[56]. L.Wang, W.Wong, and D.Xu, "A threat model driven approach for security testing," in Proc.3rd Int.Workshop Software Eng.for Secure Syst.(SESS'07), May 2007.
[57]. B.Schneier, "Attack trees,"Dr.Dobb's J.Softw.Tools, vol.24, no.12, pp.21-29, 1999.
[58]. F.Swiderski and W.Snyder, ThreatModeling.Redmond, WA, USA: Microsoft Press, 2004.
[59]. J.P.McDermott, "Attack net penetration testing," in Proc.2000 Workshop New Security Paradigms, 2000, pp.15-21.