International Journal of Soft Computing 11 (2): 76-82, 2016
ISSN: 1816-9503

Verifying an Anti-Phishing Model Using Formal Methods

[1]Abdullah M. Alnajim and [2]Hazim S. AlRawashdeh
[1]Department of Information Technology, Qassim University, Buraydah, Saudi Arabia
[2]Department of Computer Science, Buraydah Private Colleges, Buraydah, Saudi Arabia

Abstract: Phishing is an attack that tricks people into revealing sensitive information. An effective anti-Phishing approach was implemented and evaluated by conducting a controlled lab user experiments. Before the approach is deployed in the real world, this study presents a formal verification for the approach model and shows that unreachability case does not exist. This research has successfully validated the model using SPIN Model checker. The research has also validated the model against two LTL properties and no error was found.

Key words: Phishing, fraud, internet, model checking, formal methods, networking, security

INTRODUCTION

Phishing is a social engineering attack that aims at exploiting the weakness found in system processes as caused by system users (Khonji *et al*., 2013). This attack is a case of internet fraud that uses the internet as a medium to carry out financial frauds (Philippsohn, 2001). The parties involved in any transaction never need to meet and the user may have no idea whether the goods or services exist. Due to this, the internet is a good vehicle to defraud people who use it to buy goods or services (Philippsohn, 2001). An online system's access keys could be stolen. Online systems such as electronic commerce, electronic banking, electronic voting and electronic mail are targets for fraudsters.

Internet fraud has a multiplicity of forms including Phishing attacks. Phishing is an attack that seeks to trick people into revealing sensitive information about themselves and their internet accounts (Downs *et al*., 2006). Phishing aims to take advantage of the way humans interact with computers rather than taking advantage of technical system vulnerabilities.

Phishing is mitigated by various approaches. Anti-Phishing training for end-users is one of these approaches that is complementary to any proposed technical solution such as TrustBar and SpoofGuard toolbars. It is tested and recommended that end-user training is a key component in Phishing attacks mitigation (Alnajim and Munro, 2009; Robila and Ragucci, 2006).

This study proposes a formal verification for an anti-Phishing approach that helps users to detect Phishing websites (Alnajim and Munro, 2009). The verification helps to check whether the model is feasible in order to apply it in the real world. In this research, there is an assumption that Phishing attacks do not use software to change the host files in users' operating systems. This is called 'Pharming' and is different from Phishing.

Literature review

Anti-Phishing approaches: There have been technical (e.g., toolbars) and training (e.g., IQ tests) approaches to mitigate Phishing. The anti-Phishing toolbars are web browser plug-ins that warn users when they reach a suspected Phishing website (Cranor *et al*., 2006). Anti-Phishing tools use two major methods for mitigating Phishing websites. The first method is to use heuristics to check the host name and the URL for common spoofing techniques. The second method is to use a blacklist that lists Phishing URLs.

The heuristics approach is not 100% accurate since it produces low False Negatives (FN), i.e., a Phishing website is mistakenly judged as legitimate which implies they do not catch all Phishing website. The heuristics often produce high False Positives (FP), i.e., incorrectly identifying a legitimate site as fraudulent. Blacklists have a high level of accuracy because they are constructed by paid experts who verify a reported URL and add it to the blacklists if it is considered as a Phishing website (Zhang *et al*., 2007).

Many financial and commercial, private and government institutions (e.g., eBay and HSBC) have provided anti-Phishing training tips for detecting Phishing emails and websites. The aim of the tips is to train users to look for Phishing clues located in emails and websites to enable them to make better decisions in distinguishing Phishing emails and websites. People in general do not read anti-Phishing online training materials although some of them are found effective when used (Alnajim and Munro, 2008).

Corresponding Author: Abdullah M. Alnajim, Department of Information Technology, Qassim University, Buraydah, Saudi Arabia

There are also anti-Phishing training courses such as IQ tests and class assessments (Robila and Ragucci, 2006). The courses explain to users what Phishing attacks are and how to prevent them. The disadvantage of the courses' approach is that typically people are unlikely to attend them.

An online game was proposed in order to teach users good habits to help them avoid Phishing attacks (Sheng *et al*., 2007). Kumaraguru *et al*. (2007) considered training people about Phishing email during their normal use of email. Their aim was to teach people what Phishing clues to look for located in emails. They found that email training approach works better than the current practice of publishing or sending anti-Phishing tips. However, Kumaraguru *et al*. (2007)'s approach does not consider teaching people with Phishing website-related tips. Phishing website can be reached via various methods in addition to emails such as online advertisements.

Embedded training is considered to be effective (Anderson *et al*., 1996). When using embedded training, training is not a separate activity but it is an ongoing activity that is an integral part of the workplace and its system (Kozlowski *et al*., 2001). When training materials incorporate the context of the real world, training will be most effective (Anderson *et al*., 1996). Embedded training idea is applied in anti-Phishing approaches. One of these approaches is shown in Fig. 1.

The idea is to check whether a user is Phishing aware when they surf the internet and visit a Phishing website. If the user tries to submit their sensitive information to the Phishing website they are shown intervening message to help them understand what Phishing websites are and how to detect them. The approach also keeps anti-Phishing training ongoing process. This means, in all time, once a user tries to submit information to a Phishing website, they will be trained. In the case where the user is Phishing aware, the approach does nothing and lets the user keep surfing the internet.

The approach shown in Fig. 1 is implemented and evaluated by conducting a controlled lab user experiments (Alnajim and Munro, 2009). It is found that this approach brings information to end-users and helps them immediately after they have made a mistake in order to recognize Phishing websites by themselves. The results

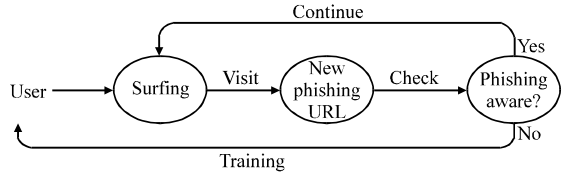

Fig. 1: The idea of the Anti-Phishing Model

analysis states that there is a significant positive effect of using the approach in comparison with an old approach of sending anti-Phishing tips by email. The approach is better in helping users properly judge legitimate and Phishing websites.

The approach shown in Fig. 1 as a model will be evaluated and verified formally before it is deployed in the real world. The model will be tested to make sure formally that unreachability case does not exist and for its functional correctness.

Formal verification: Formal verification is considered as an important topic in the field of formal methods. Formal methods are then considered as mathematical techniques and tools which can be used for the modeling, specification and verification of systems (Wing, 1990). All these aspects are concerned with formal behavior description of systems and to which degrees these systems' reflect the specification.

Formal verification is performed by looking up the state space of the system using model checkers for embracement of the specification properties (Aziz *et al*., 1994). If these properties hold then model checkers mostly return empty file but if there is a violation in a property then a TRAIL file (i.e., the output file of SPIN model checker) or a counter example (i.e., the output file of SMV Model checker) will be generated to show how the violation has taken place. Model checker can represent the result as a sequence of model states that contain the model variables and their values at that state which incrementally cause the violation (Bolton *et al*., 2014).

The finite state machine has to be checked to validate the modeled system. SPIN verification is to carried out against safety and liveness aspects (Hegde *et al*., 2012). The desired aim from the safety properties is not have bad experience. For an example, the model should be free of invariant and deadlock so the system will never reach no possible states.

SPIN checks a safety property by trying to find a trace leading to the "undesired" thing. If there is not such a trace, the property is satisfied (Strunk *et al*., 2006).

SPIN Model checker will evaluate assertions that could be placed between any two statements in the state space (Hegde *et al*., 2012). If it finds a calculation which causes a false assertion then that could mean either the program is incorrect or the assertion doesn't properly express a correctness property (Hegde *et al*., 2012). The evaluation is carried on in the simulation process with true or false expression for specific statement. If s statement which is proceeded correctly then the model checker will

proceed to the next statement but if it is not proceeded correctly then the program will terminate showing a trail with the number of these errors (Samanta *et al*., 2008).

LTL (Linear Tree Logic) can be used for some cases that models time sequence as they can be translated by SPIN into a never claim that is executed together with the finite automaton that represents the PROMELA Program. LTL may be used to check certain properties of the system like safety, liveliness and absence of deadlocks (Huth and Ryan, 2001).

Having evaluated the effectiveness of ant-Phishing approach shown in Fig. 1 (Alnajim and Munro, 2009), this study evaluates the approach's efficiency by using formal methods. This research verifies the Anti-Phishing Model presented in Fig. 1 formally. The verification helps to check whether the model is feasible (i.e., unreachability case does not exist) in order to apply it in the real world. The Anti-Phishing Model is verified formally using SPIN Model checker via UML modeling language. The approach in this study uses UML class and state diagrams to specify the state model and the state transitions based on these diagrams. These UML diagrams are then translated into Promela (i.e., the input language of SPIN Model checker) so it can be analyzed by SPIN.

MATERIALS AND METHODS

This research expands the application of model checking in several domains. It focuses on the formal verification of Anti-Phishing Model using SPIN Model checker to check for vulnerabilities based on the system components as shown in Fig. 2. Each component of the model is considered as a different process. These processes are identified in PROMELA language (Process or Protocol Meta Language) which reads the behavior of the processes as mentioned in the state chart diagram shown in Fig. 3. These model entities interconnect with each other using global channels. To achieve this goal, various properties of the Anti-Phishing Model are verified.

Therefore, the steps taken to perform the formal verification are described as follows. First of all, the Anti-Phishing Model is designed by UML (Unified Modeling Language) state diagram using ArgoUML CASE tool. The UML Model is mapped into a Promela code using a tool called Hugo/RT which can capture the properties of the model and save it as a Promela Program. The generated code is then combined with some LTL properties that will be used for verifying the timing of the model responses as shown in Table 1.

Figure 2 shows the ant-Phishing scenarios with various components. If the user tries to submit their sensitive information to the Phishing website, they are given trainings. Otherwise they are considered aware and let go.

Having the previous scenario, the correspondent Promela code for the Anti-Phishing Model can be written as follows:

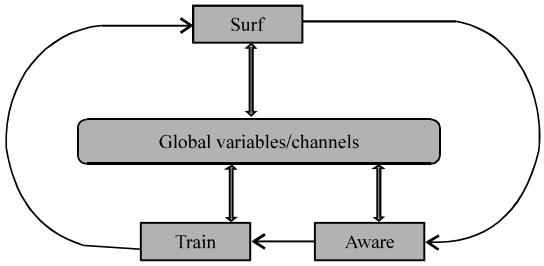

Fig. 2: The ant-Phishing scenarios with various components

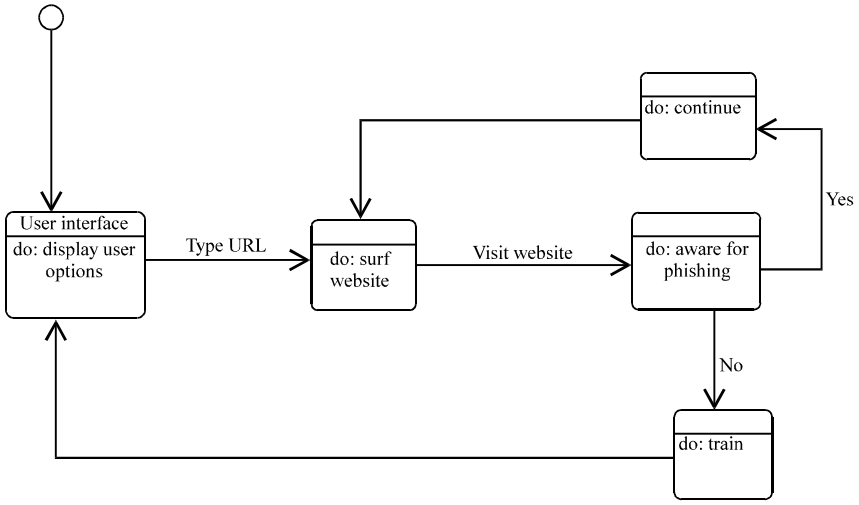

Fig 3: State diagram for Anti-Phishing System

Table1: Modeling formal properties

Properties	Description
Property 1	If a Phishing website is visited and the user is aware then the system lets the user continue surfing (Filename: preliminary-ltl-1.prp)
LTL property	G (website_visted Λ¬phishing_transaction¬ F (surfing_continued)
SPIN Syntax	[](website_visted && ! phishing_transaction -><>(surfing_continued))
Buchi automata	never { /* !([](website_visted -><>phishing_reached)) */ T0_init: if :: (! ((website_visted)) && (phishing_reached)) -> goto accept_S4 :: (1) -> goto T0_init fi; accept_S4: if :: (! ((phishing_reached))) -> goto accept_S4 fi; }
Property 2	If a phishing website is visited and there is no phishing awareness, it will do the training process for the user. (Filename: preliminary-ltl-2.prp)
LTL property	G (phishingunaware_reached¬ F user_learned)
SPIN Syntax	[](phishingunaware_reached -><>user_learned)
Buchi automata	never { /* !([](website_visted && !continue_transaction -><>(phishingunaware_reached && user_learned))) */ T0_init: if :: (((! ((user_learned)) && ! ((continue_transaction)) && (website_visted)) || (! ((continue_transaction)) &&! ((phishingunaware_reached)) && (website_visted)))) -> goto accept_S3 :: (1) -> goto T0_init fi; accept_S3: if :: (((! ((user_learned))) || (! ((phishingunaware_reached))))) -> goto accept_S3 fi; }

```
User (msg)
1   /*precondition: The user types a URL*/
2   /*postcondition: The user responds to the learning policy*/
3   switch
4   case msg = AwarForPhishing:
5   return Phishing (ContinueSurfing)
6   case msg = NoAwareForPhishing:
7   return Phishing (TrainUser)
8   case msg = Success:
9   return Success
10  case msg = Failure:
11  return Failure
12          Surf website !VISTED;
13          printf("Phishing Process: Prompt user...\n");
14          goto surfwebsite;
15  ::      printf("Phishing Process: Website is
        unphished...\n");
16          goto training;
17          goto SurfWebsitState;
18  fi;
19
20  ...
21
22  }
```

On the other hand, Fig. 3 illustrates the system model using a state diagrams for the Anti-Phishing System.

RESULTS AND DISCUSSION

The Promela code for the Anti-Phishing Model is translated and written as its shown:

```
1   proctype phishing(){
2   printf("initiating Phishing Process...\n");
3   DisplayMenuState:
4   UserInterface!DISPLAY_Options->
5     printf("Phishing Process: URL Typed...\n");
6   SurfWebsit?SVist_WebsiteL->
7       goto AwareForPhishing;
8
9   AwareForPhishing:
10    if
11    ::      printf("Phishing Process: WebSite is
        phished...\n");
```

SPIN verification is run to check for deadlock and unexecuted code in the Anti-Phishing Model. As shown Fig. 4, SPIN did not report "invalid end state" as there is no deadlock in the model. In addition, there is no error and unexecuted codes as all processes have 0 unreached states.

The states PrintAwareForPhishingState and NoAwareForPhishingState after SurfWebsiteState have multiple entry points. Thus, the properties of these two states with LTL properties are checked as shown in Table 1. The two properties are as follow. The first one is "If a Phishing website is visited and the user is aware then the system lets the user continue surfing". The second property is "If a Phishing website is visited and there is no Phishing awareness, it will do the training process for the user".

For verifying these LTL properties in Table 1, they are written as a LTL property that is certainly suitable for

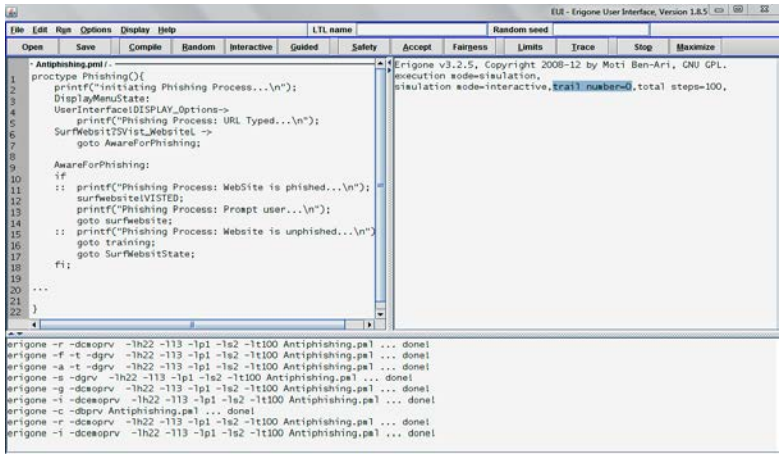

Fig 4: Spin Model checker run result

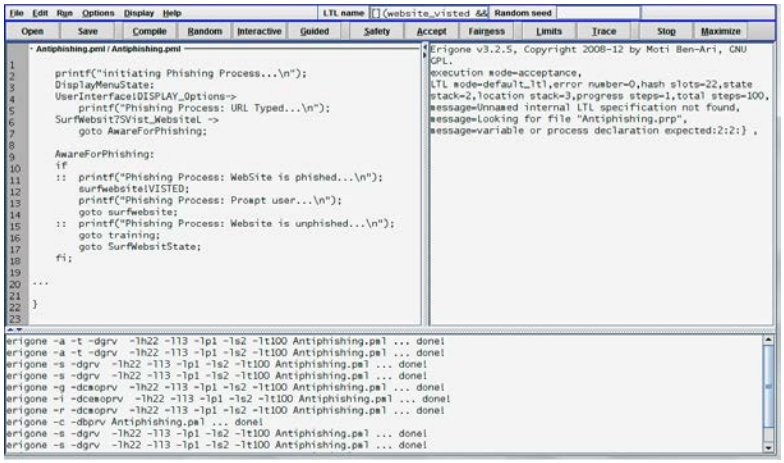

Fig. 5: First property

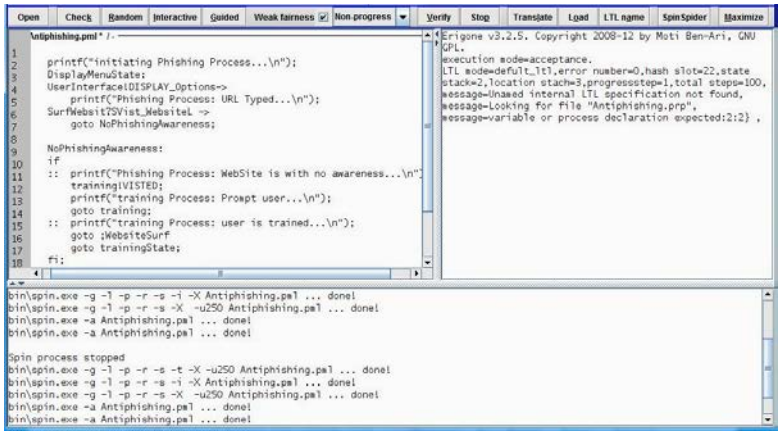

Fig. 6: Second property

SPIN Model checker syntax using EUI Version of iSPIN whereas it is translated to Buchi automata and verified in the acceptance mode.

Having the above procedures performed run, SPIN reported no errors for the two LTL properties as shown in Fig. 5 and 6.

The stage 2 checking result showed that the Anti-Phishing Model has passed these properties in the validation process. In summary, SPIN Model checker has helped to validate that the model has no deadlock and unreachable code.

CONCLUSION

This research has achieved its goal and expectations by mapping the model and formally verifying the Anti-Phishing Model. The research has successfully coded the model using Promela language and validated it using SPIN Model checker. The research performed the verification by checking deadlock and unreachable code. It is found that deadlock and unreachability states do not exist in the model. The model has also have been validated against two LTL properties (Table 1) and no error was found.

Directions of future work, another model checker such as Symbolic Model Verifier (SMV) can be used and run over the model to achieve a clear result about the correctness of the Anti-Phishing Model behavior.

REFERENCES

Alnajim, A. and M. Munro, 2008. An evaluation of user's tips effectiveness for Phishing websites detection. Proceedings of the 3rd IEEE International Conference on Digital Information Management, November 13-16, 2008, IEEE Press, London, pp: 63-68.

Alnajim, A. and M. Munro, 2009. An anti-Phishing approach that uses training intervention for Phishing websites detection. Proceedings of the 6th International Conference on Information Technology: New Generations, April 27-29, 2009, Las Vegas, NV., pp: 405-410.

Anderson, J.R., L.M. Reder and H.A. Simon, 1996. Situated learning and education. Educ. Res., 25: 5-11.

Aziz, A., F. Balarin, S.T. Cheng, R. Hojati and T. Kam *et al*., 1994. HSIS: A BDD-based environment for formal verification. Proceedings of the 31st Conference on Design Automation, June 6-10, 1994, San Diego, CA., pp: 454-459.

Bolton, M.L., N. Jimenez, M.M. van Paassen and M. Trujillo, 2014. Automatically generating specification properties from task models for the formal verification of human-automation interaction. IEEE Trans. Human-Machine Syst., 44: 561-575.

Cranor, L., S. Egelman, J. Hong and Y. Zhang, 2006. Phinding Phish: An evaluation of anti-Phishing toolbars. Technical Report, CMU-CyLab-06-018, Carnegie Mellon University, Pittsburgh, PA., USA., pp: 1-20.

Downs, J.S., M.B. Holbrook and L.F. Cranor, 2006. Decision strategies and susceptibility to Phishing. Proceedings of the 2nd Symposium on Usable Privacy and Security, July 12-14, 2006, Pittsburgh, PA., USA., pp: 79-90.

Hegde, M.S., H.K. Jnanamurthy and S. Singh, 2012. Modelling and verification of extensible authentication protocol using spin model checker. Int. J. Network Secur. Applic., 4: 81-98.

Huth, M. and M. Ryan, 2001. Logic in Computer Science: Modelling and Reasoning about Systems. Cambridge University Press, Cambridge.

Khonji, M., Y. Iraqi and A. Jones, 2013. Phishing detection: A literature survey. IEEE Commun. Surveys Tutorials, 15: 2091-2121.

Kozlowski, S.W.J., R.J. Toney, M.E. Mullins, D.A. Weissbein, K.J. Brown and B.S. Bell, 2001. Developing adaptability: A theory for the design of integrated-embedded training systems. Adv. Hum. Perform. Cognitive Eng. Res., 1: 59-123.

Kumaraguru, P., Y. Rhee, A. Acquisti, L.F. Cranor, J. Hong and E. Nunge, 2007. Protecting people from Phishing: The design and evaluation of an embedded training email system. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, San Jose, CA, USA., April 30-May 03, 2007, ACM, Press, New York, pp: 905-914.

Philippsohn, S., 2001. Trends in cybercrime-An overview of current financial crimes on the internet. Comput. Secur., 20: 53-69.

Robila, S.A. and J.W. Ragucci, 2006. Don't be a phish: Steps in user education. Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, June 26-28, 2006, ACM, Press, New York, pp: 237-241.

Samanta, R., J.V. Deshmukh and E.A. Emerson, 2008. Automatic generation of local repairs for boolean programs. Proceedings of the International Conference on Formal Methods in Computer-Aided Design, Portland, OR, USA., November 17-20, 2008, IEEE Press Piscataway, New Jeresy, USA., pp: 1-10.

Sheng, S., B. Magnien, P. Kumaraguru, A. Acquisti, L.F. Cranor, J. Hong and E. Nunge, 2007. Anti-Phishing Phil: The design and evaluation of a game that teaches people not to fall for phish. Proceedings of the 3rd Symposium on Usable Privacy and Security, July 18-20, 2007, Pittsburgh, PA., USA., pp: 88-99.

Strunk, E.A., M.A. Aiello and J.C. Knight, 2006. A survey of tools for model checking and model-based development. Technical Report CS-2006-17, Department of Computer Science University of Virginia June 2006. http://www.cs.virginia.du/~eas9d/papers/CS-TR-2006-17.pdf.

Wing, J.M., 1990. A specifier's introduction to formal methods. IEEE Comput., 23: 8-23.

Zhang, Y., J. Hong and L. Cranor, 2007. CANTINA: A content-based approach to detecting Phishing web sites. Proceedings of the 16th International Conference on World Wide Web, May 8-12, 2007, Banff, Alberta, Canada, pp: 639-648.