Development of an Autonomous Retesting Penetration Testing Paradigm

Jeremy Straub
Department of Computer Science
North Dakota State University
Fargo, ND, USA
jeremy.straub@ndsu.edu

***Abstract*—This paper presents a paradigm for automating penetration testing for the purpose of retesting systems to ensure that previously detected issues have not been reintroduced. This approach, which is patterned off of path-based attack techniques, is described and an implementation of the proposed paradigm, using the Blackboard Architecture, is presented. The efficiencies and potential pathway to more automated testing provided by the proposed paradigm are also discussed.**

Keywords—artificial intelligence, penetration testing, retesting, cybersecurity, ethnical hacking, Blackboard Architecture

I. Introduction

Cybersecurity has become a major area of international focus. Numerous causes for this exist, such as its implications for nations' security, the losses incurred by firms that have suffered data breaches and individuals concern for their personal information. In some cases, security vulnerabilities are caused by human error or the compromise of human-held information through social engineering. In other cases, misconfiguration is responsible. In many cases, though, issues that originated during software or hardware design and implementation are to blame.

Penetration testing is used by many firms as part of a multi-faceted cybersecurity strategy. During a penetration test, ethical hackers attempt to break into IT systems to identify vulnerabilities so that they can be corrected before nefarious individuals find and exploit them. Due to the skillset required and limited number of individuals with these skills, penetration tests are expensive and, in many cases, occur over a short period of time. While some firms have internal penetration testing teams, most external penetration tests are short-term projects with time gaps in between them. Even with internal testers, there may be time gaps due to the need for the testers to split their time between testing multiple internal systems.

This paper discusses the efficacy of automating penetration testing in a manner similar to how identified software bugs' retesting is automated during some software engineering processes. It presents a multi-step process for testing, including test identification, test development, primary testing, test outcome evaluation, automated retesting, and ongoing outcome evaluation. It also describes a methodology for implementing this testing paradigm.

II. Background

This section provides an overview of prior work in several different areas which inform the work presented herein. First, prior work on cybersecurity, generally, is presented. Next, testing automation is discussed. Following this, the Blackboard Architecture is reviewed. Finally, prior work on command and control for cybersecurity is presented.

A. Cybersecurity

The interconnectivity of modern information technology systems has created an increased surface area for attackers to target and numerous security vulnerabilities [1]. Security incidents have been shown to be likely to cost companies at least \$10,000, while data breaches can cost \$1 million or more [2]. Attacks can also damage organizations' reputations [3].

In response to these issues, numerous approaches to cybersecurity have been proposed. For example, King, et al. [4] proposed a focus on "human factors" while Mateski, et al. [5] developed a "threat matrix" which can draw upon a wide number of techniques. Techniques which focus on identifying threat vectors – which include the Microsoft STRIDE [6], MITRE ATT&CK [7] and Lockheed Martin Cyber Kill Chain [8] – can provide input to this matrix. Tree-based systems [9], such as attack graphs [10], can also be used. The use of these trees for cybersecurity automation has also been previously considered [11]. Manual analysis can also use these techniques to direct funding and effort to the areas of highest need [12].

B. Testing automation

Stefinko, Piskozub and Banakh [13] proffer that "manual penetration tests are still more popular and useful" than automated ones. However, a variety of automated penetration testing tools are publicly available [14]. Many penetration testing tools utilize an attack library; however, testing a limited set of scenarios doesn't prove that a system is secure, due to the potential of unforeseen [15] and not included attack types. Systems may also be vulnerable to complex attacks which cannot be exposed by a single test [16]. Testing tools may also inadvertently damage infrastructure [17] and create outages themselves [18]. Tools can also create information overload for their users and present other issues [19].

In addition to the existing tools, research has been conducted on automation techniques that could support the development of new tools or the augmentation of existing ones to support testing systems such as web services [20]–[23], cloud applications [24], [25], internet of things devices [26] and WiFi networks [27], as well as technologies such as Blockchain contracts [28]. Vulnerability-specific testing software has also been developed [29]. A number of approaches to testing automation have been

979-8-3503-2028-2/22/\$31.00 ©2022 IEEE
DOI 10.1109/CSCI58124.2022.00229

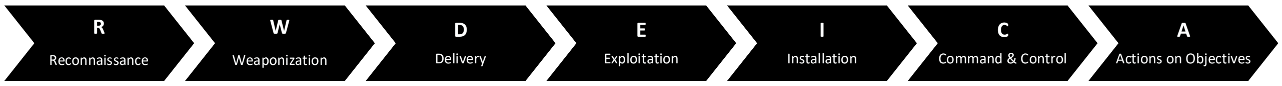

Figure 1. Lockheed Martin Cyber Kill Chain [67].

proposed which utilize techniques such as machine learning [30]–[36], static and dynamic analysis [37], agent-based modeling [38], expert systems [39] and threat models [40].

C. Blackboard Architecture

One prospective approach to implementing testing automation is to utilize the Blackboard Architecture, which was introduced by Hayes-Roth [41] based on the Hearsay II system [42]. The Blackboard Architecture adds an actualization capability to rule-fact expert systems which trace their roots to the early AI systems of Dendral and Mycin [43]–[45].

At its most basic, a Blackboard Architecture system has a central Blackboard that stores the data used by the system [46]. However, numerous enhancements to the basic system have been proposed, such as techniques to increase speed [47], [48] and to facilitate distributed [49] processing, message handling [50] and parallel processing [51]. Solving [52], [53] has also been demonstrated as a way to facilitate the implementation of goal-driven Blackboard Architecture systems.

The Blackboard Architecture has found numerous uses. Examples include robotic command [54], [55], vehicle control [56] and modeling proteins [57].

D. Cybersecurity command and control

Command and control (C2) capabilities are critical for both nefarious attackers and penetration testers [58]. Penetration testers, in particular, must be aware of adversaries' potential C2 capabilities, to ensure they conduct testing in similar ways. Attackers, for example, could use mutators [59] to avoid signature-based detection [60] and malware that lays dormant for a period of time to avoid anti-malware detection [60].

Both nefarious attackers and penetration testers may need to use lateral movement [61] and decentralization [62] to access systems that are not directly attackable from their current vantage point. Path-based models, such as the MITRE ATT&CK framework [12], [61], can be used to inform C2 decision making and capability needs.

III. Challenges of Penetration Testing

Penetration testing is an inherently manually intensive activity, at present. While some tools are used for scanning and carrying out attacks, much of the process relies upon the skill and expertise of human testers who identify potential points of vulnerability within networks and computing systems that they believe are likely to be exploitable.

While a number of methodologies for testing exist, penetration testing is typically not considered conclusive. Instead, it seeks to find vulnerabilities so that they can be corrected. However, a penetration test that ends without identifying vulnerabilities does not guarantee that a network is secure. In fact, if penetration testers lack an appropriate level of skill and experience, are not up to date on current vulnerabilities, or are unfamiliar with some areas of a system, the test outcome may be largely meaningless.

Because of the heavy reliance on human testers' skill and expertise and the limited number of people with this skillset, testers command high wage levels and can be hard to recruit. Many organizations may find them unaffordable altogether or may only be able to afford short tests at infrequent intervals.

When testers are available at an organization, the results may vary tremendously by the individual or individuals conducting the test. The types of attacks used, systems targeted and, thus, the overall results may be highly dependent on the tester performing the analysis.

Because of this, subsequent human testing may focus on finding new vulnerabilities, as opposed to verifying that previously known ones have not reoccurred. In fact, a new group of penetration testers may not even have access to the report from an earlier group – particularly if an intervening discovered-vulnerability retest has occurred to verify that the vulnerability has (at least temporarily) been resolved.

IV. Path-based models

A variety of attack, threat and penetration testing models have been developed based on a path-finding approach. Two of the most commonly known ones are the Cyber Kill Chain and ATT&CK models. The Microsoft STRIDE framework, similarly, uses a threat tree-like model focused on data. These models, and their potential use in automation, are now discussed.

The Lockheed Martin Cyber Kill Chain model [63], which is shown in Figure 1, begins with a reconnaissance phase, where the target landscape is assessed. This is then followed by several phases concerned with actually preparing and launching an attack and a phase where a local command capability is established. Finally, actions on objectives – the actual goal of the attack – are performed during the final phase.

The MITRE ATT&CK model [7], [64] is very similar to the Cyber Kill Chain model. While both models differ in the supplemental materials and systems that they offer, their core has significant overlap [11], as shown in Table 1. There are a limited number of differences, though. The ATT&CK model combines the "exploitation" and "installation" phases, from Cyber Kill Chain, into a single "exploit" phase. The ATT&CK model also includes a "maintain" phase focused on retaining a footprint on and the ability to conduct future attacks against a system, that is explicitly included in the Cyber Kill Chain model.

The Microsoft STRIDE [6] model is comprised of five steps: "decomposition", "create data flow diagrams", "analyze the data flow diagrams for threats", "identification of vulnerabilities based on these threats", and "develop mitigation approaches". This approach is data-centric and allows the modeling of both system components and interactions between them. A key rationale for this approach is that simply analyzing components may not capture all threats relevant to a system [6].

Table 1. Comparison of ATT&CK and Cyber Kill Chain models [11].

ATT&CK	Cyber Kill Chain
Recon	*Reconnaissance*
Weaponize	*Weaponization*
Deliver	*Delivery*
Exploit	*Exploitation*
▨	*Installation*
Control	*Command and Control*
Execute	*Actions on Objectives*
Maintain	▨

Prior work [11] has demonstrated how these models can be implemented using an artificial intelligence system. Figure 2 combines the ATT&CK, Cyber Kill Chain and Microsoft STRIDE models into a single combined model. Then, in Figure 3, an example of this combined operational model is shown implemented as a Blackboard Architecture system.

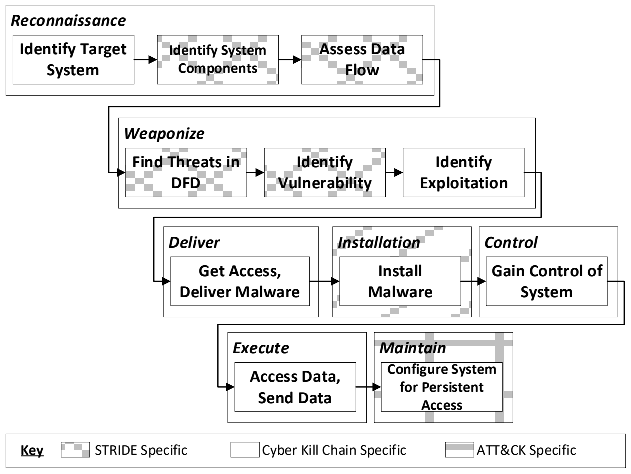

Figure 2. Example of attack framework combining Cyber Kill Chain, STRIDE and ATT&CK frameworks [11].

This Blackboard Architecture-based implementation can be further extended with tools, such as the testing mechanism proposed in [58] and the system knowledge collection and modeling capability proposed in [65]. However, the full implementation of testing automation may be beyond the capabilities and budgets of many firms. Thus, this paper focuses on a retest-based paradigm that is much simpler than the fully automated approach proposed in [65].

V. RETEST-BASED PARADIGM

The retest-based paradigm (RBP) is based on a simple concept of developing all penetration tests used during a testing campaign in an automatable manner, implemented via a tool. This can be done in three ways. First, the test can be developed using an automation tool initially and the initial testing can be performed using the tool. Second, a manual approach can be used to initially conduct the test (which may be particularly useful if exploration or experimentation is needed before a particular test is finalized). This is immediately followed by the embodiment of this test into a tool. Finally, a bank of existing tests may exist (either generally or from prior firm activities) which can be used without requiring new implementation.

The RBP, which is depicted in Figure 4, begins with a process of test identification. During this process, the human penetration tester identifies a particular system that they wish to attack-test and a method that they wish to use to do so. Then, a test is developed or an existing test is obtained and testing is performed (or testing is performed and the test is developed afterwards). In all cases, the outcomes of this initial test (i.e., whether it detected a vulnerability) are evaluated.

Finally, unless the test is judged to not be useful for the system through the outcome evaluation process, it is setup for retesting. This is performed on an ongoing basis with outcome evaluation being performed after each iteration of testing.

The goal of this testing paradigm is to significantly reduce the cost of the retesting phase by removing the need for a human to perform the test (though testing is still human initiated, controlling when it occurs). Allowing the testing to be performed without the expensive resource of the human penetration tester removes a major cost. It also facilitates recurrent retesting of all areas of a system to allow reintroduced vulnerabilities to be detected more quickly. In at least some cases, vulnerabilities may be introduced by gaps in system administrator knowledge or administrator misunderstandings. It

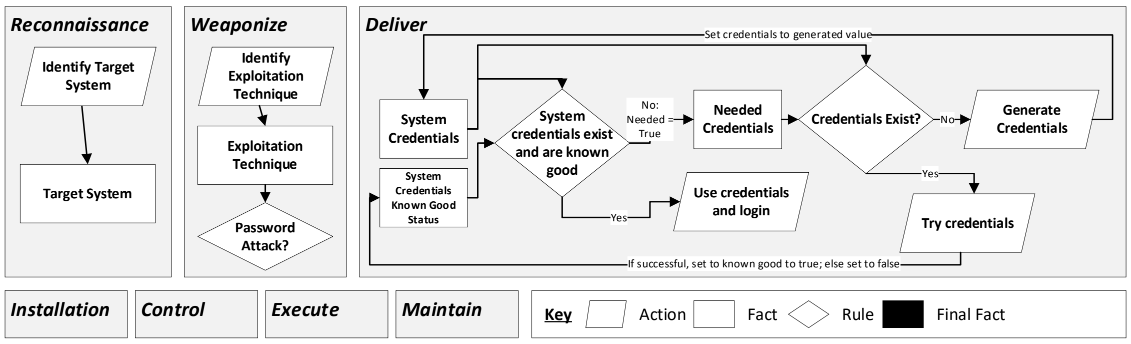

Figure 3. Example of attack using combined Cyber Kill Chain, STRIDE and ATT&CK framework [11].

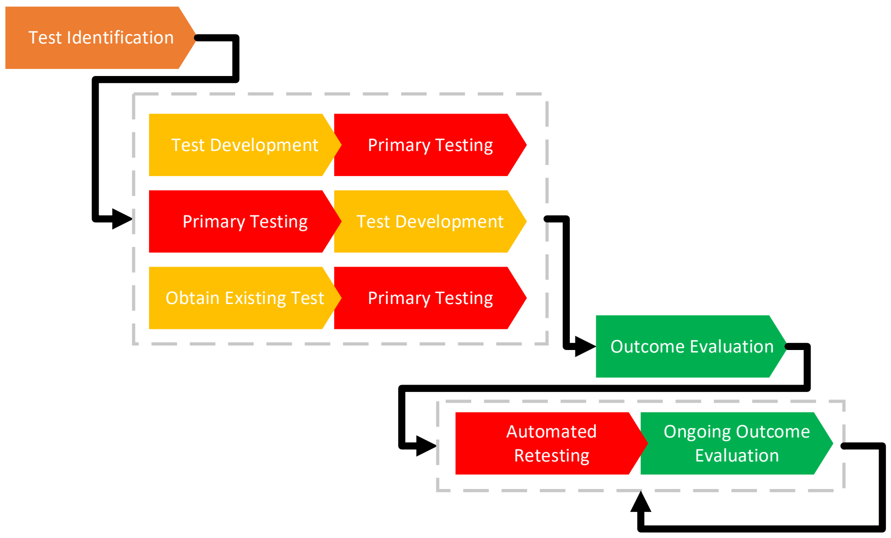

Figure 4. Automated penetration testing retest paradigm.

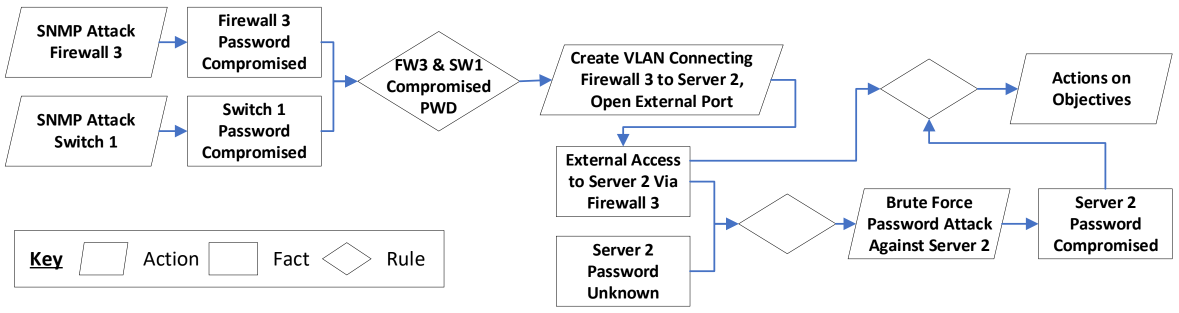
SNMP Attack Firewall 3
Firewall 3 Password Compromised
SNMP Attack Switch 1
Switch 1 Password Compromised
FW3 & SW1 Compromised PWD
Create VLAN Connecting Firewall 3 to Server 2, Open External Port
External Access to Server 2 Via Firewall 3
Server 2 Password Unknown
Brute Force Password Attack Against Server 2
Server 2 Password Compromised
Actions on Objectives
Key
Action
Fact
Rule

Figure 5. Blackboard Architecture implementation of re-testing paradigm.

is, thus, likely that the same (or similar) issues could be reintroduced during the normal course of operations due to the same individuals making configuration changes that reintroduce them. Software defect-based vulnerabilities could be reintroduced by the manufacturer or through the installation of the same (vulnerable) software version on the same or other computers. Attackers who are able to maintain a foothold on the system or network during the time that a vulnerability is corrected may even re-introduce it to facilitate their access into different systems or areas. Automated retesting should, ideally, provide a low-cost way to rapidly reidentify these issues.

VI. Use of Blackboard Architecture for Retesting

To implement the retesting paradigm using the Blackboard Architecture, a network of rules, facts and actions is created. This network includes knowledge about system state (embodied in facts), decision-making rules, and attack steps (embodied in actions). An attack-test is, thus, modeled using a collection of these three node types.

Every attack and system implementation will differ somewhat; however, the data elements needed for a given type of attack progression will be similar. In many cases, if they are well designed, the actual attack-actions will be able to be reused with only settings modifications. This Blackboard Architecture network is developed (and potentially used) during the primary testing, as previously described, and can then be used for subsequent testing.

Figure 5 depicts an example of how a simple attack could be implemented. In this case, a simple network management protocol (SNMP) attack is used to compromise the passwords of a firewall and switch. These devices are reconfigured to create a VLAN connecting the firewall to a server and to open an outside port on this firewall. Finally, a brute force attack is implemented against the server which compromises it and allows the actions on objectives to be performed.

Notably, this is simplified somewhat for presentation. A real-world implementation would potentially have more detail in several areas. Additional actions may be needed to collect data regarding the state of devices after an attack (to determine if it was successful or not). Some actions, such as the one that creates the VLAN and opens the ports, could be further decomposed into multiple nodes for greater understandability by operators and component reusability.

This simple example shows that the creation of a Blackboard Architecture implementation of an attack-test is logical and relatively straight forward. Notably, a tool that allowed the attack to be created visually, by creating the structure shown in

Figure 5, could aid its use greatly. The Blackboard Architecture implementation requires additional time over simply carrying out the attack; however, it also produces documentation that could aid the human penetration tester in the development of their testing report (particularly if a tool was available to facilitate this). The exact time cost to the primary testing will vary somewhat, as reusability of attacks could save significant amounts of time; however, new attack development and structure creation would likely be somewhat slower than simply performing a given test. Given this, the additional time cost should be considered in light of the combined value of attack-test reusability and resting (versus the additional time costs incurred) to determine whether to utilize this paradigm for a particular testing program or portion of a testing program.

VII. Advantages and Drawbacks

There are a number of advantages and drawbacks to the proposed system. Several key ones are now discussed.

A. Advantages

Several of the most pronounced benefits of the proposed system have already been discussed. The paradigm facilitates retesting to identify recurring vulnerabilities. It also can provide time savings via the reuse of attack-tests, once they have been implemented and utilized initially. The potential documentation benefits of the approach have also been briefly discussed.

In addition to these, a number of other potential sources of benefit merit consideration. First, the retesting tasks and the adaptation of attack-tests, developed for one context, for other uses provides tasks that can be performed by junior penetration testers. Due to the high reliance on the skills and abilities of the testers for test effectiveness, pathways for junior penetration testers to gain the skills needed, while also contributing to the performance of tests, are not always readily available. This paradigm creates a clear set of roles for junior penetration testers which are comparable to junior roles for IT staffers and programmers.

Second, the paradigm promotes greater penetration test rigor. The ability to define a test in terms of standard modules that are used (and can be re-used for re-testing) removes ambiguity regarding exactly what tests were performed (or not) and allows both the areas of robustness and vulnerability of the system under test to be documented. This may also have a notable benefit for IT professionals who may be upset with penetration testing results that only document system security failures.

Third, the proposed paradigm could have documentation benefits for testers and testing clients, particularly if attack-tests are annotated with metadata and attack-test step details at the point of creation. This data could facilitate the rapid creation of robust reports that explain what was tested, how it was tested, document the results of testing and how assessment was conducted. These reports could be more detailed, potentially use references to standard attack-test module details to remove redundant text, and also be faster to create than less detailed manually created vulnerability documentation reports.

Fourth, by facilitating greater documentation of all activities, the proposed paradigm may facilitate quality assurance of penetration testing activities. A focus on documenting all tests that were run and their results, as opposed to producing only vulnerability documentation, allows the testers' decisions to be assessed to determine whether they were selecting and performing a robust collection of tests. Additionally, testing routines, for testing specific types of systems performing specific functions, could be developed which could serve as design patterns for testing, further augmenting its robustness.

B. Drawbacks

Several key drawbacks to the proposed approach also exist. First, the approach could result in testing that is very mechanical and which does not explore the nuances of the system being tested. While performing well under a standardized collection of tests provides some assurance as to the security level of the system, it does not mean the system is secure. Penetration testing should always include a focus on the 'unknown unknown' which an adversary could utilize to target the system. Thus, while using common modules can save time and retesting can identify if problems have returned, this shouldn't be a substitute for manual exploration for implementation-specific issues. Ideally, time saved from performing automatable tasks could be redirected to higher value manual exploration use.

Second, the proposed approach could potentially lead to a false sense of confidence if it were to completely replace exploratory manual testing. Systems might not be vulnerable to the common test suite elements; however, this should not be taken as saying that they are secure. Defining testing requirements (e.g., for specific industries) in terms of test suites could further reinforce this problematic overconfidence and lead to exploratory penetration testing not being performed.

Third, this same type of issue could manifest itself with firms trying to conduct penetration testing entirely with inexperienced or less experienced, than would be typical, penetration testers. This approach would rely highly on the collection of pre-defined attack-tests. While using only less experienced testers for re-testing may be appropriate (though not a replacement for regular exploratory testing by highly qualified testers), utilizing inexperienced testers for primary testing would be problematic and result in tests that are potentially of minimal value (particularly if they don't identify issues for correction).

Finally, the use of pre-packaged tests or custom-designed tests for retesting may create issues if minimal changes are made to the system under test that make it no longer vulnerable to the specific test implementation without fixing the underlying issue. While retesting can be an effective way to potentially identify recurring or not-yet-fixed vulnerabilities, it should not be a substitute to new testing.

VIII. Pathway to Autonomous Testing

As previously discussed, the development of a collection of attack-tests facilitates reduced work being required, over time, for the manually-controlled test-retest model described herein, due to the potential for test reuse. However, perhaps one of the largest longer-term benefits of the utilization of the proposed approach is that this same library of tests can be used to facilitate automated testing, if the tests are annotated with details indicating criteria for their appropriate use. This facilitates movement towards the autonomous testing approach described in [58], [65], [66], as it removes the issue of a large upfront investment being required to develop the test suite that is needed to make the system useful. While the fully autonomous system is of limited use with a limited attack-test set collection (unless it was curated for a specific limited-purpose use), the automation of specifically identified testing for the approach described herein can be immediately useful while also developing a collection of attack-tests that can support automated use.

IX. Conclusions and Future Work

This paper has presented a test and retest-based paradigm for penetration testing automation which is patterned off of attack-path based techniques, such as the ATT&CK and Cyber Kill Chain models. Notably, while these models demonstrate one way the system could be utilized, its efficacy is not limited to approaches compliant with them. This paper has also demonstrated how attacks could be implemented using the Blackboard Architecture to facilitate retesting and automation.

It is hoped that this paradigm can have an immediate benefit for organizations by facilitating rapid and automated retesting to verify that prior configuration issues and software vulnerabilities have not been inadvertently or deliberately reintroduced. Over time, organizations may benefit from the development of a collection of reusable attack modules to facilitate the testing of new or reconfigured environment. Attack modules could also be shared between organizations to provide additional benefit.

The development of modules and attack logic, for use during retesting, will also facilitate the acquisition of knowledge and resources that can be used to facilitate more fully automated testing. This, potentially, will provide a pathway to greater levels of primary test penetration testing automation.

References

[1] S. Hasan, A. Ghafouri, A. Dubey, G. Karsai, and X. Koutsoukos, "Vulnerability analysis of power systems based on cyber-attack and defense models," 2018 IEEE Power Energy Soc. Innov. Smart Grid Technol. Conf. ISGT 2018, pp. 1–5, Jul. 2018, doi: 10.1109/ISGT.2018.8403337.

[2] P. Dreyer et al., "Estimating the Global Cost of Cyber Risk: Methodology and Examples," Santa Monica, CA, 2018. Accessed: Jan. 26, 2022. [Online]. Available: www.rand.org/jie/stp.

[3] N. Kesswani and S. Kumar, "Maintaining Cyber Security: Implications, Cost and Returns," in SIGMIS-CPR'15, Jun. 2015, pp. 161–164, doi: 10.1145/2751957.2751976.

[4] Z. M. King, D. S. Henshel, L. Flora, M. G. Cains, B. Hoffman, and C. Sample, "Characterizing and measuring maliciousness for cybersecurity risk assessment," Front. Psychol., vol. 9, no. FEB, pp. 1–19, 2018, doi: 10.3389/fpsyg.2018.00039.

[5] M. Mateski et al., "Cyber Threat Metrics," Sandia Natl. Lab. Rep., Mar. 2012, Accessed: Feb. 19, 2022. [Online]. Available: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online.

[6] R. Khan, K. McLaughlin, D. Laverty, and S. Sezer, "STRIDE-based threat modeling for cyber-physical systems," in 2017 IEEE PES Innovative Smart Grid Technologies Conference Europe, ISGT-Europe 2017 - Proceedings, Jul. 2017, vol. 2018-January, pp. 1–6, doi: 10.1109/ISGTEurope.2017.8260283.

[7] B. E. Strom et al., "Finding Cyber Threats with ATT&CKTM-Based Analytics," Jun. 2017.

[8] T. Yadav and A. M. Rao, "Technical Aspects of Cyber Kill Chain," in International Symposium on Security in Computing and Communication, Springer, 2015, pp. 438–452.

[9] H. S. Lallie, K. Debattista, and J. Bal, "A review of attack graph and attack tree visual syntax in cyber security," Comput. Sci. Rev., vol. 35, p. 100219, Feb. 2020, doi: 10.1016/J.COSREV.2019.100219.

[10] A. K. Nandi, H. R. Medal, and S. Vadlamani, "Interdicting attack graphs to protect organizations from cyber attacks: A bi-level defender–attacker model," Comput. Oper. Res., vol. 75, pp. 118–131, Nov. 2016, doi: 10.1016/J.COR.2016.05.005.

[11] J. Straub, "Modeling Attack, Defense and Threat Trees and the Cyber Kill Chain, ATTCK and STRIDE Frameworks as Blackboard Architecture Networks," in 2020 IEEE International Conference on Smart Cloud, Nov. 2020, pp. 148–153.

[12] T. H. Bhuiyan, A. K. Nandi, H. Medal, and M. Halappanavar, "Minimizing expected maximum risk from cyber-Attacks with probabilistic attack success," 2016 IEEE Symp. Technol. Homel. Secur. HST 2016, Sep. 2016, doi: 10.1109/THS.2016.7568892.

[13] Y. Stefinko, A. Piskozub, and R. Banakh, "Manual and automated penetration testing. Benefits and drawbacks. Modern tendency," Mod. Probl. Radio Eng. Telecommun. Comput. Sci. Proc. 13th Int. Conf. TCSET 2016, pp. 488–491, Apr. 2016, doi: 10.1109/TCSET.2016.7452095.

[14] M. P. Shah, "Comparative Analysis of the Automated Penetration Testing Tools," National College of Ireland, Dublin, 2020.

[15] F. Wotawa, "On the automation of security testing," Proc. - 2016 Int. Conf. Softw. Secur. Assur. ICSSA 2016, pp. 11–16, Feb. 2017, doi: 10.1109/ICSSA.2016.9.

[16] H. H. Thompson, "Why security testing is hard," IEEE Secur. Priv., vol. 1, no. 4, pp. 83–86, Jul. 2003, doi: 10.1109/MSECP.2003.1219078.

[17] F. Guo, Y. Yu, and T. C. Chiueh, "Automated and safe vulnerability assessment," Proc. - Annu. Comput. Secur. Appl. Conf. ACSAC, vol. 2005, pp. 150–159, 2005, doi: 10.1109/CSAC.2005.11.

[18] S. M. Mohammad and L. Surya, "Security Automation in Information Technology," Int. J. Creat. Res. Thoughts, vol. 6, no. 2, Jun. 2018, Accessed: Jan. 28, 2022. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3652597.

[19] M. Metheny, "Continuous monitoring through security automation," Fed. Cloud Comput., pp. 453–472, 2017, doi: 10.1016/B978-0-12-809710-6.00013-5.

[20] A. Falkenberg, C. Mainka, J. Somorovsky, and J. Schwenk, "A new approach towards DoS penetration testing on web services," Proc. - IEEE 20th Int. Conf. Web Serv. ICWS 2013, pp. 491–498, 2013, doi: 10.1109/ICWS.2013.72.

[21] N. Antunes and M. Vieira, "Penetration testing for web services," Computer (Long. Beach. Calif)., vol. 47, no. 2, pp. 30–36, 2014, doi: 10.1109/MC.2013.409.

[22] C. Mainka, J. Somorovsky, and J. Schwenk, "Penetration testing tool for web services security," Proc. - 2012 IEEE 8th World Congr. Serv. Serv. 2012, pp. 163–170, 2012, doi: 10.1109/SERVICES.2012.7.

[23] N. Singh, V. Meherhomji, and B. R. Chandavarkar, "Automated versus Manual Approach of Web Application Penetration Testing," 2020 11th Int. Conf. Comput. Commun. Netw. Technol. ICCCNT 2020, Jul. 2020, doi: 10.1109/ICCCNT49239.2020.9225385.

[24] V. Casola, A. De Benedictis, M. Rak, and U. Villano, "A methodology for automated penetration testing of cloud applications," Int. J. Grid Util. Comput., vol. 11, no. 2, pp. 267–277, 2020, doi: 10.1504/IJGUC.2020.105541.

[25] V. Casola, A. De Benedictis, M. Rak, and U. Villano, "Towards automated penetration testing for cloud applications," Proc. - 2018 IEEE 27th Int. Conf. Enabling Technol. Infrastruct. Collab. Enterp. WETICE 2018, pp. 30–35, Oct. 2018, doi: 10.1109/WETICE.2018.00012.

[26] G. Yadav, A. Allakany, V. Kumar, K. Paul, and K. Okamura, "Penetration Testing Framework for IoT," Proc. - 2019 8th Int. Congr. Adv. Appl. Informatics, IIAI-AAI 2019, pp. 477–482, Jul. 2019, doi: 10.1109/IIAI-AAI.2019.00104.

[27] S. P. Kadam, B. Mahajan, M. Patanwala, P. Sanas, and S. Vidyarthi, "Automated Wi-Fi penetration testing," Int. Conf. Electr. Electron. Optim. Tech. ICEEOT 2016, pp. 1092–1096, Nov. 2016, doi: 10.1109/ICEEOT.2016.7754855.

[28] A. Bhardwaj, S. B. H. Shah, A. Shankar, M. Alazab, M. Kumar, and T. R. Gadekallu, "Penetration testing framework for smart contract Blockchain," Peer-to-Peer Netw. Appl., vol. 14, no. 5, pp. 2635–2650, Sep. 2021, doi: 10.1007/S12083-020-00991-6/TABLES/7.

[29] S. Shah and B. M. Mehtre, “An automated approach to vulnerability assessment and penetration testing using net-nirikshak 1.0,” Proc. 2014 IEEE Int. Conf. Adv. Commun. Control Comput. Technol. ICACCCT 2014, pp. 707–712, Jan. 2015, doi: 10.1109/ICACCCT.2014.7019182.

[30] M. C. Ghanem and T. M. Chen, “Reinforcement Learning for Intelligent Penetration Testing,” Proc. 2nd World Conf. Smart Trends Syst. Secur. Sustain. WorldS4 2018, pp. 90–95, Jan. 2019, doi: 10.1109/WORLDS4.2018.8611595.

[31] J. Schwartz and H. Kurniawati, “Autonomous Penetration Testing using Reinforcement Learning,” arXiv Prepr., May 2019, Accessed: Feb. 18, 2022. [Online]. Available: https://arxiv.org/abs/1905.05965v1.

[32] R. Gangupantulu et al., “Using Cyber Terrain in Reinforcement Learning for Penetration Testing,” arXiv Prepr., Aug. 2021, Accessed: Feb. 18, 2022. [Online]. Available: https://arxiv.org/abs/2108.07124v1.

[33] M. C. Ghanem and T. M. Chen, “Reinforcement Learning for Efficient Network Penetration Testing,” Inf. 2020, Vol. 11, Page 6, vol. 11, no. 1, p. 6, Dec. 2019, doi: 10.3390/INFO11010006.

[34] S. Chaudhary, A. O’Brien, and S. Xu, “Automated Post-Breach Penetration Testing through Reinforcement Learning,” 2020 IEEE Conf. Commun. Netw. Secur. CNS 2020, Jun. 2020, doi: 10.1109/CNS48642.2020.9162301.

[35] Z. Hu, R. Beuran, and Y. Tan, “Automated Penetration Testing Using Deep Reinforcement Learning,” Proc. - 5th IEEE Eur. Symp. Secur. Priv. Work. Euro S PW 2020, pp. 2–10, Sep. 2020, doi: 10.1109/EUROSPW51379.2020.00010.

[36] K. Tran et al., “Deep hierarchical reinforcement agents for automated penetration testing,” arXiv Prepr., Sep. 2021, Accessed: Feb. 18, 2022. [Online]. Available: https://arxiv.org/abs/2109.06449v1.

[37] W. G. J. Halfond, S. R. Choudhary, and A. Orso, “Improving penetration testing through static and dynamic analysis,” Softw. Testing, Verif. Reliab., vol. 21, no. 3, pp. 195–214, Sep. 2011, doi: 10.1002/STVR.450.

[38] G. Chu and A. Lisitsa, “Poster: Agent-based (BDI) modeling for automation of penetration testing,” 2018 16th Annu. Conf. Privacy, Secur. Trust. PST 2018, Oct. 2018, doi: 10.1109/PST.2018.8514211.

[39] M. Rak, G. Salzillo, and D. Granata, “ESSecA: An automated expert system for threat modelling and penetration testing for IoT ecosystems,” Comput. Electr. Eng., vol. 99, p. 107721, Apr. 2022, doi: 10.1016/J.COMPELECENG.2022.107721.

[40] N. A. Almubairik and G. Wills, “Automated penetration testing based on a threat model,” 2016 11th Int. Conf. Internet Technol. Secur. Trans. ICITST 2016, pp. 413–414, Feb. 2017, doi: 10.1109/ICITST.2016.7856742.

[41] B. Hayes-Roth, “A blackboard architecture for control,” Artif. Intell., vol. 26, no. 3, pp. 251–321, 1985.

[42] L. D. Erman, F. Hayes-Roth, V. R. Lesser, and D. R. Reddy, “The Hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty,” ACM Comput. Surv., vol. 12, no. 2, pp. 213–253, 1980.

[43] E. A. Feigenbaum, B. G. Buchanan, and J. Lederberg, “On generality and problem solving: A case study using the DENDRAL program,” Stanford Univ. Rep., 1970, [Online]. Available: https://ntrs.nasa.gov/api/citations/19710028679/downloads/19710028679.pdf.

[44] V. Zwass, “Expert system,” Britannica, Feb. 10, 2016. https://www.britannica.com/technology/expert-system (accessed Feb. 24, 2021).

[45] R. K. Lindsay, B. G. Buchanan, E. A. Feigenbaum, and J. Lederberg, “DENDRAL: A case study of the first expert system for scientific hypothesis formation,” Artif. Intell., vol. 61, no. 2, pp. 209–261, Jun. 1993, doi: 10.1016/0004-3702(93)90068-M.

[46] J. Dong, S. Chen, and J.-J. Jeng, “Event-based blackboard architecture for multi-agent systems,” in Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on, 2005, vol. 2, pp. 379–384.

[47] G. Goodman and R. Reddy, “Alternative control structures for speech understanding systems,” 1977.

[48] I. D. Craig, “CASSANDRA-II: a distributed Blackboard system,” Department of Computer Science, University of Warwick, Coventry, UK, 1987. [Online]. Available: http://eprints.dcs.warwick.ac.uk/1210/1/cs-rr-090.pdf.

[49] K. S. Ettabaa, I. R. Farah, B. Solaiman, and M. Ben Ahmed, “Distributed blackboard architecture for multi-spectral image interpretation based on multi-agent system,” in Information and Communication Technologies, 2006. ICTTA’06. 2nd, 2006, vol. 2, pp. 3070–3075.

[50] I. D. Craig, “A new interpretation of the Blackboard architecture,” Department of Computer Science, University of Warwick, Coventry, UK, 1993. [Online]. Available: http://eprints.dcs.warwick.ac.uk/1368/1/cs-rr-254.pdf.

[51] H. Velthuijsen, B. J. Lippolt, and J. C. Vonk, “A parallel blackboard architecture,” in Proc. Third Int. Expert Systems Conf, 1987, vol. 487, p. 493.

[52] J. Straub, “Evaluation of a multi- goal solver for use in a blackboard architecture,” Int. J. Decis. Support Syst. Technol., vol. 6, no. 1, 2014, doi: 10.4018/ijdsst.2014010101.

[53] J. Straub, “Comparing the Effect of Pruning on a Best-Path and Naive-Approach Blackboard Solver,” Int. J. Autom. Comput., vol. 12.5, pp. 503–510, 2015.

[54] G. Brzykcy, J. Martinek, A. Meissner, and P. Skrzypczynski, “Multi-agent blackboard architecture for a mobile robot,” in 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2001, vol. 4, pp. 2369–2374.

[55] Y. Yang, Y. Tian, and H. Mei, “Cooperative Q learning based on blackboard architecture,” in International Conference on Computational Intelligence and Security Workshops, 2007, 2007, pp. 224–227.

[56] A. M. de Campos and M. J. Monteiro de Macedo, “A blackboard architecture for perception planning in autonomous vehicles,” in 1992 International Conference on Industrial Electronics, Control, Instrumentation, and Automation, 1992, pp. 826–831.

[57] M. V Johnson Jr and B. Hayes-Roth, “Integrating Diverse Reasoning Methods in the BBP Blackboard Control Architecture,” in AAAI-87 Conference, 1987, pp. 30–35, [Online]. Available: https://www.aaai.org/Papers/AAAI/1987/AAAI87-006.pdf.

[58] J. Hance, J. Milbrath, N. Ross, and J. Straub, “Distributed Attack Deployment Capability for Modern Automated Penetration Testing,” Comput. 2022, Vol. 11, Page 33, vol. 11, no. 3, p. 33, Feb. 2022, doi: 10.3390/COMPUTERS11030033.

[59] P. Fogla, M. Sharif, R. Perdisci, O. Kolesnikov, and W. Lee, “Polymorphic Blending Attacks,” in Security ’06: 15th USENIX Security Symposium, 2006, pp. 241–256.

[60] J. Gardiner, M. Cova, and S. Nagaraja, “Command & Control: Understanding, Denying and Detecting - A review of malware C2 techniques, detection and defences,” arXiv Prepr. arXiv1408.1136, Aug. 2014, Accessed: Jan. 25, 2022. [Online]. Available: https://arxiv.org/abs/1408.1136v2.

[61] CrowdStrike, “What is Lateral Movement,” CrowdStrike Website, 2022. https://www.crowdstrike.com/cybersecurity-101/lateral-movement/ (accessed Jan. 28, 2022).

[62] D. Dittrich and S. Dietrich, “Command and Control Structures in Malware,” Login, vol. 32, no. 6, pp. 8–17, Dec. 2007.

[63] M. S. Khan, S. Siddiqui, and K. Ferens, “A Cognitive and Concurrent Cyber Kill Chain Model,” in Computer and Network Security Essentials, Springer International Publishing, 2018, pp. 585–602.

[64] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington, and C. B. Thomas, “MITRE ATT&CKTM: Design and Philosophy,” Jul. 2018.

[65] N. Ritter, R. Fedor, M. Johnson, D. Newstrum, and J. Straub, “Evaluation of the Efficacy of a Blackboard Architecture-Based Automated Cybersecurity Assessment Sensing Tool,” Submitt. to Expert Syst. with Appl.

[66] J. Straub, “POSTER: Blackboard-Based Electronic Warfare System,” in Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 2015, pp. 1681–1683.

[67] J. Straub, “Software Engineering: The First Line of Defense for Cybersecurity,” Proc. IEEE Int. Conf. Softw. Eng. Serv. Sci. ICSESS, vol. 2020-Octob, pp. 531–536, 2020, doi: 10.1109/ICSESS49938.2020.9237715.