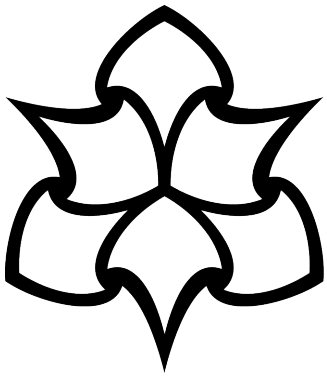

Evolutionary Algorithms in Web Security: Exploring Untapped Potential

1st Sam Attwood
School of Computing and Mathematics
Manchester Metropolitan University
Manchester, United Kingdom
S.Attwood@mmu.ac.uk

2nd Wanpeng Li
School of Computing and Mathematics
Manchester Metropolitan University
Manchester, United Kingdom
W.Li@mmu.ac.uk

3rd Rupak Kharel
School of Computing and Mathematics
Manchester Metropolitan University
Manchester, United Kingdom
R.Kharel@mmu.ac.uk

***Abstract*—Many aspects of our lives are dependent upon the Web. Research that considers how we can better protect the Web and the products and services delivered using it–that is research into Web Security–is therefore of the upmost importance. Consequently, this paper considers evolutionary algorithms and their potential (which is to tap into and harness the power of natural evolution) within the field of Web Security. The paper provides a concise overview of existing studies that have used evolutionary algorithms as a tool within Web Security. More specifically, the paper considers the manner in which evolutionary algorithms have been applied, the types of problems that they have been applied to, and the way in which they have been assessed or evaluated. Furthermore, an opportunity to better harness the power of natural evolution within the field of Web Security, by applying modern evolutionary algorithms that draw inspiration from the open-ended aspect of natural evolution, is highlighted and discussed.**

***Index Terms*—web security, evolutionary computation, SQL injection, cross-site scripting, open-ended evolution**

I. Introduction

The Web is an intrinsic part of society and utilised by a variety of organisations. Naturally, the services these organisations provide via the Web are an appealing target for malicious actors that will look to exploit a myriad of potential vulnerabilities [1], [2] for personal gain. A fact that is demonstrated by the recent Internet Security Threat Report from Symantec [3]. The importance of Web Security as a research area is therefore paramount. Research into Web Security facilitates the creation of new technologies that have the potential to minimize the inherent risks associated with the Web, such that we can ultimately achieve a safer and more secure Web.

Evolutionary algorithms (EAs) are algorithms studied within the field of Evolutionary Computation that mimic the process of natural evolution, typically by implementing mutation, crossover and selection operators [4]. They are often thought of as a general approach that can be applied to solve many different difficult optimisation problems [4]. However, there is an emerging argument that the true potential of EAs is best realised beyond the context of straightforward optimisation [5]–[12]. This viewpoint has motivated the design of several modern EAs that draw inspiration from the open-ended aspect of natural evolution and resemble conventional machine learning algorithms less than their predecessors [5]–[12].

Motivated by the aforementioned modern EAs, this paper considers both the fields of Web Security and Evolutionary Computation and searches for untapped potential where they intersect. The underlying supposition that drives this work is that the divergent quality of modern EAs is desirable in Web Security and can be used to overcome some of the challenges faced within this important field.

To investigate the potential of EAs in Web Security this paper first considers preexisting studies in this area. More specifically, it attempts to establish: the approaches that have been used when applying EAs within Web Security, the purposes for which EAs have been applied within Web Security, and the way in which the EAs have been evaluated within the context of Web Security. It achieves this via a concise narrative literature review that is presented in section III. Section IV builds on this review and explores a particularly promising area of untapped potential–the application of modern EAs to problems within the field of Web Security.

II. Background

A. Web Security

In terms of security, the Web is a fast changing battleground. Malicious actors are always looking to find and exploit new vulnerabilities whilst those tasked with securing our online services look to stay ahead of them. Web Security has therefore long been, and will likely continue to be, a highly active research area with an ever-changing landscape.

In our increasingly app-driven world resources produced by the Open Web Application Security Project (OWASP) can be used to ascertain the most prevalent issues in Web Security. According to OWASP injection flaws, flaws that enable cross-site scripting, and broken authentication and authorisation mechanisms are some of the most prevalent vulnerabilities on the Web [1], [2]. Much of the recent research in Web Security relates to these vulnerabilities. For example, a number of studies, that are of particular relevance to this study, introduce testing techniques designed to help expose injection and cross-site scripting vulnerabilities [13]–[21]. The following subsections therefore consider these two vulnerabilities in more detail.

1) SQL Injection: An injection vulnerability usually occurs when a web server trusts and executes commands or queries using input from untrusted sources. Malicious actors can exploit injection vulnerabilities to cause the web server to execute malicious commands, which could have a huge impact on an organisation and result in data loss, disclosure to unauthorized parties, or denial of access [1], [2]. SQL injection is arguably the most well known type of injection but other forms, such as NoSQL Injection and Command Injection, pose a similar threat [1], [2].

An SQL Injection occurs when a web server does not filter a user's input and feeds it directly into a SQL statement. Practically, this often occurs when SQL statements are formed via string concatenation, like in the following example:

```
sqlStatement = "SELECT * FROM Users
    WHERE Name = '" + userName + ';"
```

The example shows a user input `userName` variable being used to form a SQL statement via string concatenation. If a malicious actor entered `' OR '1'='1' --` as an input and this was assigned to the `userName` variable, the assignment to the `sqlStatement` variable would be equivalent to the following:

```
sqlStatement = "SELECT * FROM Users
    WHERE Name = '' OR '1'='1' -- ';"
```

This is problematic as although a valid query has been assigned, the query does more than the author of the code intended. The `OR '1'='1'` clause will always be true and allows for the original condition to be bypassed. The vulnerability could therefore be exploited by a malicious actor such that they retrieve all of the data associated with all of the users in the database, which would of course be extremely damaging to any organisation.

2) Cross-Site Scripting: According to OWASP [22], Cross-Site Scripting (XSS) attacks involve the injection of malicious scripts into otherwise benign and trusted web sites. XSS attacks rely on subtle vulnerabilities being present in victim websites. XSS attacks occur when a malicious actor uses a flawed but otherwise legitimate web application to send malicious code, typically in the form of a browser script, to a different end user. Web application flaws that allow these attacks to succeed are quite widespread and occur whenever a web application includes input from one user in generating its output for another user without validating or encoding it.

A malicious actor can exploit a XSS vulnerability to send a malicious script to an unsuspecting user. The victim user's browser has no way of knowing that the script should not be trusted, and will execute it. The scope of the malicious script is almost limitless. The victim's browser has no way of knowing that the script should not be trusted and allows it access to cookies, session tokens, or any other sensitive information that it retains.

Formjacking attacks involve the theft of credit card details and other information from payment forms. The type of attack–visualised in figure 1–has recently been highlighted

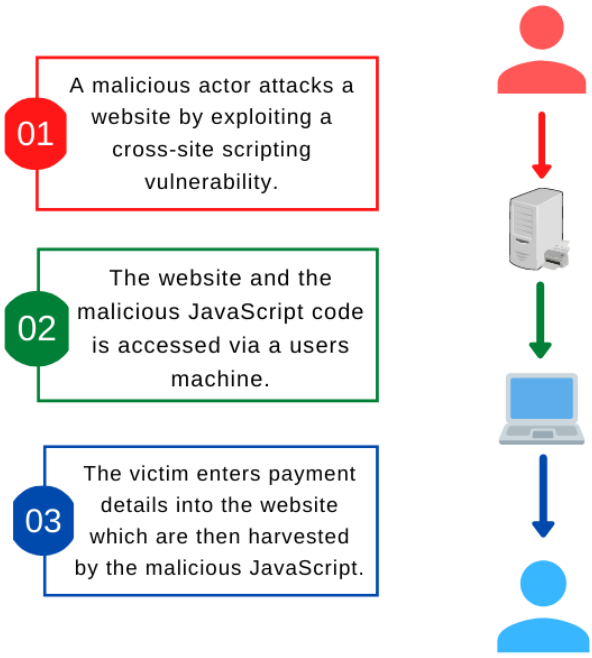

Fig. 1: The stages of a formjacking attack in which a malicious actor exploits a cross-site scripting vulnerability to steal payment details.

by Symantec in their recent Internet Security Threat Report [3] and can be considered a type of XSS attack. In their report Symantec claim to have blocked over 3.7 million formjacking attempts in 2018 [3] and thereby highlight the prevalence of XSS.

B. Evolutionary Computation

Evolutionary Computation is a broad research area in which the process of natural evolution is considered in a computational context. It can more or less be thought of as a spectrum, with studies that examine artificial life simulations at one end, and studies that examine the application of evolutionary algorithms to real-world problems at the other. Many studies within the field of Evolutionary Computation lie somewhere in the middle of this spectrum and propose new algorithms inspired by some aspect of natural evolution.

A simplified canonical EA is shown in algorithm 1. It shows the application of variation operators, such as mutation and crossover operators, to produce offspring subtly different to their parents at each generation. It then shows the combined population of parents and offspring being evaluated. In practice, this evaluation is implemented using a domain specific objective function, which is sometimes referred to as a fitness function. Finally, it shows the best individuals, according to the objective function, being selected using a selection mechanism, and then used to create the population of the next generation.

Of course, the EA shown in algorithm 1 is simplified and very general. In practice, EAs can often be hard to

implement. One problem is that the variation operators can be hard to design if the problem under consideration cannot be represented in a well understood manner. Another problem, perhaps more fundamental, is that an effective objective function can often be difficult to craft without a highly detailed understanding of a problem domain [5], [6].

Algorithm 1 A simplified canonical EA. Inspired by Back, Hammel, and Schwefel [4].

```
1:  procedure EA(max)
2:      t := 0
3:      initialize P(t)
4:      evaluate P(t)
5:      while t ≠ max do
6:          P'(t) := variation [P(t)]
7:          evaluate [P'(t)]
8:          P(t+1) := select [P'(t)]
9:          t := t + 1
10:     end while
11: end procedure
```

Despite the difficulties than can be encountered when implementing them, research into EAs has continued as the (apparently) astounding power of natural evolution as an optimizer holds its allure. However, the power of natural evolution has so far remained elusive and EAs are often criticized as their performance rarely compares with that of other optimisation algorithms [4]. This fact has contributed towards a relatively recent paradigm shift within the field of Evolutionary Computation. This paradigm shift has seen a rise in modern EAs that draw inspiration from the open-ended aspect of natural evolution and focus more on divergence, compared to traditional EAs that focus more on converging towards a global optimum (this is the best case scenario at least) [5]–[12].

Open-ended evolution is complex and not well understood– it is a grand question that will require many different fields to coalesce before it is fully understood [23]. The modern EAs that draw inspiration from the open-ended aspect of natural evolution can therefore be viewed as a part of a larger quest to fully understand the open-ended evolution phenomenon. As modern EAs are applied to more and more problems the generality and power of the mechanisms they employ will become increasingly apparent. If a mechanism is shown to be particularly general and powerful in this way, it could be argued that similar mechanisms drive natural evolution and other open-ended processes.

III. Evolutionary Algorithms in Web Security: Current Landscape

There is an emerging trend to apply evolutionary algorithms to Web Security problems. This section therefore attempts to answer three research questions, which are stated at the beginning of each subsection, by analysing the relevant preexisting literature. By doing this it aims to provide clarity across these two research areas and identify ways in which both can be advanced.

TABLE I: A summary of how EAs have been implemented, applied, and examined within Web Security.

Legend: T - traditional single-objective optimisation; M - multi-objective optimisation; I - applications related to SQL injection; X - applications related to cross-site scripting; N - not compared to an alternative approach; R - compared to a random search; ... - other approach, application, or assessment.

Studies	Approach			Application			Assessment		
	T	M	...	I	X	...	N	R	...
[13], [14]	✓				✓			✓	
[15]	✓			✓					✓
[16]		✓		✓				✓	
[17]	✓				✓		✓		
[18]	✓			✓			✓		
[24], [25]	✓					✓	✓		
[26]			✓			✓		✓	
[19]–[21]	✓				✓				✓
[27], [28]		✓				✓			✓
[29]			✓			✓			✓
[30]	✓					✓		✓	
Total	11	3	2	3	6	7	4	5	7

A. Approaches to Evolutionary Algorithms in Web Security

RQ1: *What approaches have been used when applying EAs within Web Security?*

Table I reports that 11 out of 16 surveyed studies implemented a traditional EA, where traditional EA refers to an EA that searches with a single objective in mind. One of these studies is worth briefly differentiating from the others here as it uses human input to guide a search (interactive evolution) [24]. The table also reports that 3 of the 16 studies implemented a multi-objective EA that searched for multiple Pareto optimal solutions. Finally, it reports that 2 of the 16 studies implemented an EA that can be categorized in a different way, in both cases these studies implemented a co-evolutionary algorithm.

The findings summarized in table I therefore suggest that much of the research regarding EAs in Web Security has, so far, not been particularly concerned with regards to the underlying EA that is used. The studies that implement traditional single-objective EAs best demonstrate this, however, the observation also applies to the studies that implement multi-objective EAs as all of these use NSGA-II as their underlying EA, when it could be argued an alternative such as ParEGO may be more appropriate given that solutions within a Web Security domain will most likely be computationally expensive to evaluate [31]. A similar observation has been made regarding the application of EAs for unit test generation [32].

The apparent lack of attention that has been given to the underlying EA highlights a promising research direction, one that the 2 studies implementing co-evolutionary algorithms

have arguably already begun to explore. By drawing inspiration from the latest research in the field of Evolutionary Computation it seems likely that better solutions within the field of Web Security can be achieved. Modern EAs that draw inspiration from the open-ended aspect of natural evolution seem particularly promising in this regard (see section IV). Intuition suggests the divergent aspect of these modern EAs will lend itself to test data generation within Web Security, where there is a need to find multiple input vectors that expose real vulnerabilities.

B. Applications of Evolutionary Algorithms in Web Security

RQ2: *For what purposes have EAs been applied within Web Security?*

Table I reports that 3 out of the 16 studies have applied EAs within the context of SQL injection, 6 have applied them within the context of cross-site scripting, and 7 have applied them to other areas of Web Security. Of the 7 studies applying EAs to other areas of Web Security 3 focus on spam email [25], [27], [28], 1 focuses on XML injection [29], 1 targets published vulnerabilities in a server program [30], 1 considers denial-of-service attacks [26], and 1 generates CAPTCHAs [24].

Table I therefore suggests that EAs have been applied to a reasonable variety of Web Security problems. Having said that many of the applications are similar in the sense that they generate test data to expose a particular type of vulnerability. Further research could therefore consider whether EAs can be meaningfully applied within Web Security for purposes other than test data generation.

C. Assessments of Evolutionary Algorithms in Web Security

RQ3: *How have EAs been assessed and examined when applied within Web Security?*

Table I reports that 4 out of the 16 studies did not compare their implementation to an alternative, 5 sanity checked their implementation against a random search, and 7 compare their implementation to some other alternative (often an alternative already sanity checked against a random search). The use of a random search as a sanity check is sensible [33], particularly when an EA is being applied in a new context, and it could be argued that the 4 studies not comparing their implementation could be improved by drawing comparisons against a random search. It could also be argued that more should be done to compare EAs to state-of-the-art approaches within the field of Web Security.

IV. Evolutionary Algorithms in Web Security: Untapped Potential

The review presented in section III highlights an area of untapped potential with regards to the application of evolutionary algorithms in the field of Web Security. Broadly speaking, this area of untapped potential can be explored through the application of modern evolutionary algorithms–that draw inspiration from the open-ended aspect of natural evolution and focus on divergence more than traditional EAs–within the field of Web Security. This exploration could lead to new techniques and technologies that make the Web safer and more secure. Additionally, because the exploration will involve the application of modern EAs to new problems, it will help determine the generality and power of the mechanisms employed by these EAs and thereby contribute towards the grand quest to understand open-ended evolution.

The following subsections therefore provide brief descriptions of several notable modern EAs, discuss their potential applications within the field of Web Security, and thereby begin to answer this final research question (which will be investigated further in future work):

RQ4: *What is the potential of modern EAs within Web Security?*

A. Novelty Search

The original Novelty Search algorithm, proposed by Lehman and Stanley in 2008 [5], abandoned the search for an objective entirely and instead searched for novelty. It achieved this by replacing the fitness function traditionally employed by evolutionary algorithms with what is termed a novelty metric. This novelty metric is essentially a special fitness function that measures how different new solutions are in comparison to past solutions, whereas a traditional fitness function measures progress towards a given objective. Given that it abandons the traditional notion of fitness entirely, Novelty Search can be thought of as the canonical example of a divergent search algorithm.

The divergent nature of Novelty Search lends itself to test data generation and several studies have investigated its potential in this area. However, none of the studies considered in this paper use Novelty Search to generate test data within the field of Web Security. There is therefore an opportunity to investigate Novelty Search as a tool for test data generation within the field of Web Security, but the fruitfulness of such investigations remains to be seen. The fact that Novelty Search abandons the traditional notion of fitness entirely means it can get 'lost' in large search spaces and alternative divergent search algorithms that reintroduce fitness in some form may prove to be a better option.

B. Novelty Search with Local Competition

As its name suggests, Novelty Search with Local Competition (NSLC) is a variant of the original Novelty Search algorithm that reintroduces the notion of fitness via local competitions. The algorithm aims to produce good solutions across a range of diverse niches [7]. The NSLC algorithm has been cited as the first in a new class of algorithms, referred to as quality diversity algorithms [10]. Other quality diversity algorithms, such as the Multi Dimensional Archive of Phenotypic Elites algorithm (MAP-Elites) [8], operate in different ways but similarly hope to achieve a diversity of high quality solutions in a single run.

It seems clear that NSLC and other quality diversity algorithms have something to offer the field of Web Security.

This is perhaps best demonstrated by considering the work of Avancini and Ceccato [13], who apply a traditional EA to find solutions to different path predicate expressions–an act known as path sensitization. Avancini and Ceccato had to apply their EA multiple times to find solutions to different path predicate expressions [13], whereas a quality diversity algorithm could potentially be applied just once and find solutions to many different path predicate expressions. This suggests that quality diversity algorithms, like NSLC, could prove to be useful tools with regards to path sensitization (and therefore useful tools that could help reduce the risk of SQL injection or XSS). Figure 2 hopes to illustrate this potential by highlighting the parallels between an asymmetric gauntlet domain designed explicitly to examine quality diversity algorithms [10] and a simple control flow graph, which is the type of domain considered by Avancini and Ceccato [13]. The key similarity between the domains being that both have multiple paths that can be travelled in order to find different valid solutions (in the control flow graph domain these valid solutions are highlighted by the yellow nodes and equate to test data the causes a vulnerable path to be traversed).

We have already begun investigating the potential of quality diversity algorithms as a tool to find multiple solutions to path predicate expressions (and therefore as a tool to improve the security of the Web). We have implemented an instrumented web API that contains probes on branches. Whenever the instrumented web API responds to a request it includes information with regards to which probes have been triggered in its response, which can then be used to infer the logic that has been executed by the web API (or the path that has been taken through the control graph). We have also begun implementing the quality diversity algorithm that will generate the requests, or test data, that will be sent to the web API. We intend to describe our approach more thoroughly and report empirical results in a future study.

C. Minimal Criterion Coevolution

The Minimal Criterion Coevolution (MCC) algorithm evolves a population of solutions alongside a population of problems [11]. To apply MCC in a given domain two minimal criterion–something that must be satisfied in order for a problem or solution to reproduce–must be defined [11]. For the population of problems this minimal criterion should be defined with respect to the solutions and vice versa [11]. For example, when Brant and Stanley introduced MCC they evolved mazes alongside agents tasked with solving these mazes, the mazes had to be solved by at least one solver to satisfy their minimal criterion, and the agents had to solve at least one maze to satisfy theirs [11].

The fact the Brant and Stanley observed the evolution of increasingly complex mazes and therefore inferred the evolution of increasingly complex solvers in their original study highlights the immense potential of MCC and subsequent approaches, such as the Paired Open Ended Trailblazer (POET) algorithm [12], as tools within the field of Web Security [11]. Again, this potential is best demonstrated by

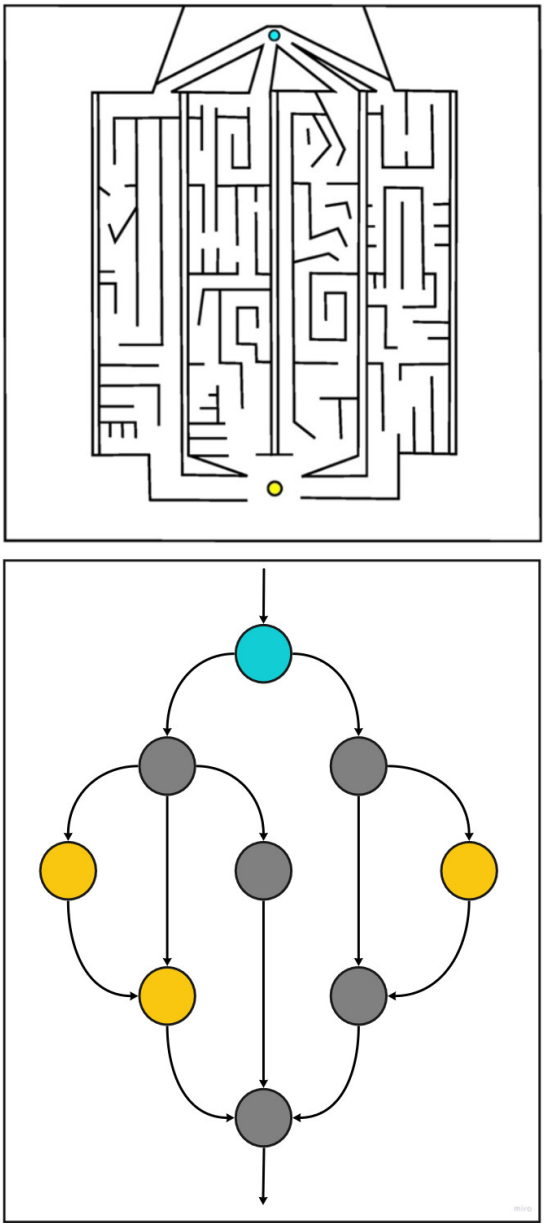

Fig. 2: A side by side comparison of the asymmetric gauntlet domain put forward by Pugh, Soros, and Stanley [10] (top) and an illustrative control flow graph (bottom).

considering an example. Appelt, Nguyen, Panichella, and Briand propose and examine a highly sophisticated (but nonetheless traditional in the sense that it is not directly inspired by aspects of open-ended evolution) EA, named ML-Driven, capable of testing web application firewalls [15]. The quartet also consider how the attack patterns found by ML-Driven could be used to repair web application firewalls [15] and thereby highlight a potential application of algorithms like MCC and POET within the field of Web Security–they could be used to achieve increasingly sophisticated SQL injection attacks alongside increasingly sophisticated web

application firewalls.

V. Conclusion

Motivated by the emerging trend of modern evolutionary algorithms (EAs) this paper has presented the findings of a search for untapped potential at the intersection of Web Security and Evolutionary Computation. It has given a brief narrative overview of preexisting studies that apply EAs within the field of Web Security. Further research could of course build on this by conducting a review that is more thorough and systematic. Nonetheless, meaningful conclusions and directions for future research have been identified.

The most significant of these conclusions is that much of the research regarding EAs in Web Security has not been particularly concerned with regards to the underlying EA that is used and that there is a great deal of untapped potential in this area. Section IV constitutes a preliminary exploration that begins tapping into (just some of) this potential. This preliminary exploration suggests that further work in this direction would be beneficial and we therefore intend to conduct further research along these lines. The purpose of this new research will be twofold: with regards to Web Security its ultimate purpose will be to facilitate the achievement of a safer and more secure Web, with regards to Evolutionary Computation its ultimate purpose will be to contribute towards our understanding of open-ended evolution.

References

[1] "Owasp top 10 - 2017," Tech. Rep., 2017, https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/.

[2] "Owasp api security top 10 2019," Tech. Rep., 2019, https://owasp.org/www-project-api-security/.

[3] "Internet security threat report," Tech. Rep., 2019, https://docs.broadcom.com/doc/istr-24-2019-en.

[4] T. Bäck, U. Hammel, and H.-P. Schwefel, "Evolutionary computation: comments on the history and current state," *IEEE Trans. Evolutionary Computation*, vol. 1, pp. 3–17, 1997.

[5] J. Lehman and K. O. Stanley, "Exploiting open-endedness to solve problems through the search for novelty." in *Proc. of the Eleventh Int. Conf. on Artificial Life (ALIFE XI)*. MIT Press, 2008, pp. 329–336.

[6] ——, "Abandoning objectives: Evolution through the search for novelty alone," *Evolutionary Computation*, vol. 19, pp. 189–223, 2011.

[7] ——, "Evolving a diversity of virtual creatures through novelty search and local competition," in *Proc. of the Genetic & Evolutionary Computation Conf.* ACM, 2011, pp. 211–218.

[8] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, pp. 503–507, 2015.

[9] A. M. Nguyen, J. Yosinski, and J. Clune, "Innovation engines: Automated creativity and improved stochastic optimization via deep learning," in *Proc. of the 2015 Annual Conf. on Genetic and Evolutionary Computation*, 2015, pp. 959–966.

[10] J. K. Pugh, L. B. Soros, and K. O. Stanley, "Quality diversity: A new frontier for evolutionary computation," *Frontiers in Robotics and AI*, vol. 3, p. 40, 2016.

[11] J. C. Brant and K. O. Stanley, "Minimal criterion coevolution: a new approach to open-ended search," in *Proc. of the Genetic and Evolutionary Computation Conf.*, 2017, pp. 67–74.

[12] R. Wang, J. Lehman, J. Clune, and K. O. Stanley, "Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions," *arXiv preprint arXiv:1901.01753*, 2019.

[13] A. Avancini and M. Ceccato, "Towards security testing with taint analysis and genetic algorithms," in *Proc. of the 2010 ICSE Workshop on Software Engineering for Secure Systems*. ACM, 2010, pp. 65–71.

[14] M. A. Ahmed and F. Ali, "Multiple-path testing for cross site scripting using genetic algorithms," *Journal of Systems Architecture*, vol. 64, pp. 50–62, 2016.

[15] D. Appelt, C. D. Nguyen, A. Panichella, and L. C. Briand, "A machine-learning-driven evolutionary approach for testing web application firewalls," *IEEE Trans. on Reliability*, vol. 67, no. 3, pp. 733–757, 2018.

[16] D. Appelt, A. Panichella, and L. C. Briand, "Automatically repairing web application firewalls based on successful sql injection attacks," *Proc. IEEE 28th Int. Symposium on Software Reliability Engineering*, pp. 339–350, 2017.

[17] A. Avancini and M. Ceccato, "Security testing of web applications: A search-based approach for cross-site scripting vulnerabilities," in *Proc. IEEE 11th Int. Working Conf. on Source Code Analysis & Manipulation*. IEEE, 2011, pp. 85–94.

[18] B. Aziz, M. Bader-El-Den, and C. Hippolyte, "Search-based sql injection attacks testing using genetic programming," in *European Conf. on Genetic Programming*. Springer, 2016.

[19] F. Duchene, S. Rawat, J.-L. Richier, and R. Groz, "Kameleonfuzz: Evolutionary fuzzing for black-box xss detection," in *Proc. of the 4th ACM Conf. on Data & Application Security & Privacy*. Association for Computing Machinery, 2014, p. 37–48. [Online]. Available: https://doi.org/10.1145/2557547.2557550

[20] A. W. Marashdih, Z. F. Zaaba, and H. K. Omer, "Web security: detection of cross site scripting in php web application using genetic algorithm," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 5, 2017.

[21] I. Hydara, A. B. M. Sultan, H. Zulzalil, and N. Admodisastro, "Cross-site scripting detection based on an enhanced genetic algorithm," *Indian Journal of Science and Technology*, vol. 8, no. 30, pp. 1–7, 2015.

[22] "Cross-site scripting (XSS)," Tech. Rep., 2016, https://owasp.org/www-community/attacks/xss/.

[23] T. Taylor, M. Bedau, A. Channon, D. Ackley, W. Banzhaf, G. Beslon, E. Dolson, T. Froese, S. Hickinbotham, T. Ikegami *et al.*, "Open-ended evolution: Perspectives from the oee workshop in york," *Artificial life*, vol. 22, no. 3, pp. 408–423, 2016.

[24] E. Y. Chen, L.-S. Huang, O. J. Mengshoel, and J. D. Lohn, "Darwin: a ground truth agnostic captcha generator using evolutionary algorithm," in *Proc. of the Companion Publication of the 2014 Annual Conf. on Genetic and Evolutionary Computation*, 2014, pp. 165–166.

[25] U. Sanpakdee, A. Walairacht, and S. Walairacht, "Adaptive spam mail filtering using genetic algorithm," *8th Int. Conf. Advanced Communication Technology*, vol. 1, pp. 441–445, 2006.

[26] M. Cordy, S. Muller, M. Papadakis, and Y. Le Traon, "Search-based test and improvement of machine-learning-based anomaly detection systems," in *Proc. of the 28th ACM Int. Symposium on Software Testing and Analysis*. Association for Computing Machinery, 2019, p. 158–168. [Online]. Available: https://doi.org/10.1145/3293882.3330580

[27] J. Dudley, L. Barone, and R. L. While, "Multi-objective spam filtering using an evolutionary algorithm," *IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pp. 123–130, 2008.

[28] Y. Zhang, H. Li, M. Niranjan, and P. Rockett, "Applying cost-sensitive multiobjective genetic programming to feature extraction for spam e-mail filtering," in *European Conf. on Genetic Programming*. Springer, 2008.

[29] S. Jan, A. Panichella, A. Arcuri, and L. Briand, "Search-based multi-vulnerability testing of xml injections in web applications," *Empirical Software Engineering*, vol. 24, no. 6, pp. 3696–3729, 2019.

[30] S. Sparks, S. Embleton, R. Cunningham, and C. C. Zou, "Automated vulnerability analysis: Leveraging control flow for evolutionary input crafting," *Twenty-Third Annual Computer Security Applications Conference*, pp. 477–486, 2007.

[31] J. Knowles, "Parego: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems," *IEEE Trans. on Evolutionary Computation*, vol. 10, no. 1, pp. 50–66, 2006.

[32] J. Campos, Y. Ge, N. Albunian, G. Fraser, M. Eler, and A. Arcuri, "An empirical evaluation of evolutionary algorithms for unit test suite generation," *Information Software Technology*, vol. 104, pp. 207–235, 2018.

[33] A. Arcuri and L. Briand, "A practical guide for using statistical tests to assess randomized algorithms in software engineering," in *Proc. of the 33rd Int. Conf. on Software Engineering*. IEEE, 2011, pp. 1–10.