Breaking the Bot Barrier: Evaluating Adversarial AI Techniques Against Multi-Modal Defense Models

Behzad Ousat
bousat@fiu.edu
Florida International University

Dongsheng Luo
dlou@fiu.edu
Florida International University

Amin Kharraz
ak@cs.fiu.edu
Florida International University

ABSTRACT

Websites utilize several approaches to detect automated agents. The agents are deployed either for beneficial purposes such as search engine crawlers, or to perform tasks on behalf of the adversary such as scanning for vulnerabilities. Recent methods in detecting such agents include the analysis of the behavior that the agents show when visiting the website. In this paper, I) we describe a deep learning framework that analyzes the triggered browser events to classify the visitor. II) We develop two adversarial attacks in order to bypass the defense by generating adversarial vectors that are misclassified by the model. III) We discuss how applicable the attacks are by reviewing the limitations of the popular tools (i.e., Selenium and Puppeteer) used for the development of automated agents based on full-fledged browsers.

CCS CONCEPTS

• **Security and privacy** → **Web application security**; *Authentication.*

KEYWORDS

Web Application Security, Bot Detection, Forensics Engine, Adversarial Machine Learning

ACM Reference Format:
Behzad Ousat, Dongsheng Luo, and Amin Kharraz. 2024. Breaking the Bot Barrier: Evaluating Adversarial AI Techniques Against Multi-Modal Defense Models. In *Companion Proceedings of the ACM Web Conference 2024 (WWW '24 Companion), May 13–17, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3589335.3651474

1 INTRODUCTION

Automated web attacks are adversarial operations on web applications, aiming to discover potential vulnerabilities and respond appropriately. Software exploitation, data breaches, service unavailability, and account takeover are just a few examples of more prevalent incidents. The consequences of such attempts are getting more severe as web applications tend to increasingly manage essential functions and process sensitive information. For instance, the Merchant Risk Council (MRC), a non-profit association for payments and fraud prevention has reported that annual online fraud losses, as a result of credential stuffing, are projected to exceed $48 billion in 2023 [3]. To respond to these emerging threats, several approaches have been proposed. Challenge-based approaches (e.g., reCaptcha), behavioral fingerprinting [7], and dynamic analysis [4, 15] are some of the common methods currently used to identify and attribute these attempts. However, automated attacks are getting more sophisticated by imitating human behavior more effectively. It is now not very difficult to trigger the event listeners (e.g., mouse movement) in most of the available automation packages (e.g. Selenium [13] and Puppeteer [12]) to satisfy some of the heuristics that are being used in current techniques.

The emerging trend on the defense side is to include more data points in the detection process to raise the complexity and cost of evasion for adversaries. That is, instead of solely using mouse movement [1, 11], newer techniques incorporate multi-modal data such as spatio-temporal artifacts (e.g., event types, time, and location) to reason about the nature of the connecting agents. The integration of multi-modal web artifacts related to web traffic with deep learning algorithms, such as Long Short-Term Memory (LSTM), presents a promising strategy for enhancing defensive mechanisms against automated adversarial traffic. However, the efficacy of this approach remains somewhat ambiguous. In particular, the adoption of adversarial machine learning methods on the trained models can make the defense mechanisms vulnerable to evasion.

In this study, we aim to answer how multi-modal defense mechanisms can be defined in practice to model human behavior against bots and how it can increase the robustness of the detection mechanism (Section 2). We then investigate to what extent the trained model is robust against possible adversarial machine learning methods such as fast gradient and brute force (Section 3). Finally, we discuss how applicable are such attacks in a real-world scenario and propose possible future works in this area (Section 4).

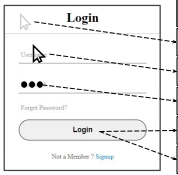
For empirical analysis, we set up a survey page to collect human interactions. We utilize the collected dataset which includes 825,701 artifacts from 46 users to train an LSTM model and implement different adversarial methodologies against it. We conclude that despite the success of the adversarial methods in generating vectors that satisfy the trained distribution, translating the crafted vector into a real-world scanning experiment was not possible due to the fundamental limitations of the automated tools to satisfy the required spatio-temporal properties used in defense. Samples of data, multi-modal defense model, and implementation details of attacks are available at https://doi.org/10.5281/zenodo.10806368.

2 MULTI-MODAL DEFENSE FRAMEWORK

In this section, we describe a deep learning approach that analyzes the behavior of the visitor based on the triggered browser events and classifies them into different categories. First, we describe the

WWW '24 Companion, May 13–17, 2024, Singapore, Singapore

ACM ISBN 979-8-4007-0172-6/24/05
https://doi.org/10.1145/3589335.3651474



Time	Event	Position	Element
T1: 2023-01-14 17:57:34.980	Mousemove	(456, 490)	div[id=login]
T2: 2023-01-14 17:57:35.315	Click	(482, 425)	input[id=username]
T3: 2023-01-14 17:57:35.510	KeyPress	N/A	input[id=password]
T4: 2023-01-14 17:57:35.585	KeyPress	N/A	button[id=submit]
T4: 2023-01-14 17:57:35.585	Submit	N/A	button[id=submit]

Figure 1: Sample Trace of the Recorded Events based on a Simplified Login Scenario

tools we have developed to collect user interactions with the target website. Then, we present the classification framework.

2.1 Artifact Collection

We build our web forensics engine on top of [9] to listen for events on the page and send the data via a web socket to the server. The forensics engine is a JavaScript library that automatically attaches event listeners to a broad array of features within the web application. Upon the activation of any such feature, a corresponding callback function is invoked. In the following, we briefly explain the multi-modal artifacts we collect as well as the experiment pipeline.

2.1.1 Multi-modal Artifact. This callback captures and logs spatio-temporal data related to the event, detailing the timestamp and the location within the webpage where the event was triggered. Figure 1 provides an illustrative example of the granularity of the data logged for a session initiated by a human user. The library is capable of capturing 44 events which can be found in detail at [9].

2.1.2 Human Data Collection. To collect human interaction data, we developed a survey website to asses how familiar the participants are with security best practices. The survey was conducted anonymously and did not collect any identifiable information about the participants. To avoid priming effects, the participants were asked to contribute to a survey on data security. After the experiment, we held a debriefing session where we explained the goal of the experiment. The survey is composed of 40 heterogeneous questions, formatted as multiple-choice checkboxes, dropdown menus, and free-response text fields to exercise different ways of interaction. To ensure that only real users can access the survey page, one specific pair of username and password was shared among all the participants to minimize the amount of data about participants. The average duration of interaction per user was approximately 35 minutes. Across the data collection period, we successfully recorded 46 distinct interaction traces, comprising a rich dataset of 825,701 individual event artifacts.

Data Protection. Before collecting human interactions, we submitted an application to the Institutional Review Board (IRB) and received approval to run the experiment. We did not collect any potentially identifiable information such as username, email address, or IP address in the course of the experiment. All the research personnel had also passed the necessary training courses to collect, preserve, and use human object data before running the experiment.

2.1.3 Bot Data Collection. To collect the artifacts from an automated scanner, we utilized *gremlins.js* [6]. Prior work [2, 14] have used the Gremlins framework to automatically generate realistic user activities in interaction with web applications. We deployed 500 instances of the gremlins separately and collected the artifacts. In the collected dataset, each instance contains an average of more than 400 artifacts. Leading to a total of 219,524 artifacts.

2.2 Data Preprocessing

Before feeding the data into our model, we take a step to preprocess the data and create subtraces that are used in the training and test phases. The length of the subtraces is a hyperparameter that can be fine-tuned based on the requirements. First, we apply one-hot encoding using the index of the events to generate the first section of the encoded vector for each of the recorded events. Then, we calculate the velocity of each mouse movement and discretize the information through quantization using several bins. The number of bins can be set as a hyper-parameter of the framework.

2.3 Detection Model

To distinguish human and automated bot sessions, we make use of an LSTM network composed of three successive LSTM layers, each with 300, 200, and 100 units, respectively. These layers utilize the hyperbolic tangent (tanh) activation function. This is followed by a fully-connected dense layer and a final Sigmoid layer. The model was trained on Nvidia Tesla K80 hardware for 50 epochs, completing the training process in a duration denoted as 15 minutes. We take two different setups into account. In the first setup (Uni-Modal setup), we only extract the mouse movement events which makes the framework comparable with previous results from related work [1, 11]. For the second case (Multi-Modal setup), we utilize all of the collected event types to find the full potential of the framework and compare the results with the uni-modal approach.

2.4 Detection Results

Before discussing possible attacks on the proposed model, we analyze the effectiveness of the trained model in differentiating human and web bots in both uni-modal and multi-modal setups. During the training phase, each available label was represented by 15,000 artifacts for training purposes, with the remaining artifacts reserved for the test phase. The test and training phases were performed three times, each with random initialization of trace indices. The accuracy versus time-to-detection tradeoff for different settings is shown in Figure 2. It can be observed that the probability of misclassification of various automated agents approaches zero in less than one second using the Multi-Modal approach.

3 ADVERSARIAL ANALYSIS

In the previous section, we described a deep learning framework that utilizes spatio-temporal artifacts to classify visitors. In this section, we first discuss the threat model and then we go over the details of the attacks. We are using an encoded version of the collected artifacts to fine-tune the model. Therefore, attacking the model means finding a sequence of encoded events that can mislead the model to misclassify the generated input.

3.1 Threat Model

The detection mechanism is fine-tuned to analyze the intricate behaviors exhibited by visitors, extracting insights from a set of

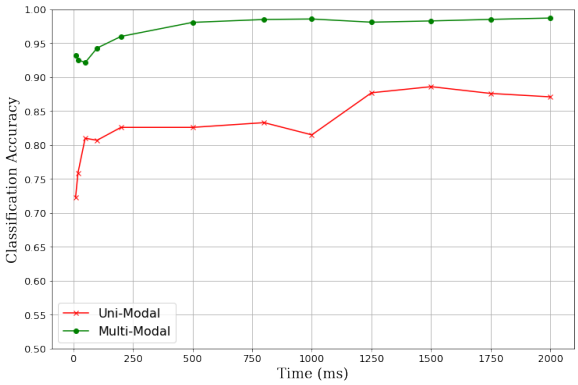


Figure 2: Classification Accuracy of Human Against Automated Agents using Multi-Modal and Uni-Modal Settings

artifacts collected during each session. We assume that the adversary possesses the ability to furnish arbitrary identities such as configurable user agents. This characteristic poses a challenge to traditional detection methods reliant on such features, rendering them unsuitable. Additionally, we assume the adversary's proficiency in crafting automated agents. These agents not only emulate the functionalities of full-fledged browsers but also simulate user interactions by making use of available APIs such as mouse movement, scroll, and keyboard events. This elevated level of sophistication necessitates a recalibration of detection strategy, as conventional approaches may prove inadequate in the face of such tactics. Finally, we assume a white-box threat model where the adversary has access to the model as well as a set of real traces that can be used to generate adversarial inputs. This allows us to investigate different attack scenarios and test the robustness of the model against them from a practical point of view.

3.2 Fast Gradient Sign Attack

The first attack focuses on developing the Fast Gradient Sign Attack (FGSM) [5] for the trained model. FGSM has been widely used to showcase adversarial attacks against learning models in image classification and has shown to be successful for tasks such as adversarial face recognition [10, 16]. While FGSM is effective in generating adversarial examples that cause misclassification by subtly perturbing continuous-valued inputs, it presents a practical challenge. In Section 2.2, we described the steps taken to generate the input of the model. These input vectors comprise discrete numerical values, indicating specific user events and their spatio-temporal properties. The generation of adversarial examples as float numbers does not align with the nature of the discrete input space, making it less practical and easy to detect with a specific module designed for the purpose. On the other hand, rounding and deterministic techniques [8] did not yield promising results on our data. This discrepancy between the continuous perturbations generated by the attack and the discrete nature of the input features introduces a gap that needs to be addressed for the attack to be applicable to real-world scenarios. Consequently, we transitioned to a brute-force attack strategy tailored to the discrete nature of the encoded vectors.

3.3 Brute Force Attack

In the brute-force approach, we aim to generate a sequence of encoded events with the same format mentioned in Section 2.2 to be classified as a specific target label. In other words, we try to create a modified sequence classified as a target class (i.e., human visitor). The brute force method exhaustively tries all possible combinations of input vectors to find an adversarial example. Although the brute force approach takes exponentially longer than FGSM to complete, it generates valid discrete inputs. The encoded vectors of events consist of two parts: the first part is the encoded event name and the second part is the spatio-temporal data which includes the position of the event on the browser and the relative time difference between two consecutive events. We build the adversarial inputs by focusing on the spatio-temporal section of the encoded vectors. There may be other possible attack vectors if we consider every sequence of events and test the output but generating such sequences is not feasible and makes the attack inapplicable.

Spatio-Temporal Tampering. The attack involves a detailed exploration of potential X, Y, and relative time values. As outlined in Section 2.2, we employ a strategy incorporating the spatial dimensions (X and Y) by discretizing them into five bins based on velocity. This discretization yields five distinct possible values for both X and Y, forming a foundational element of our attack's discrete nature. We establish a minimum and maximum relative time for generating sequences. Additionally, the subtrace's length emerges as a parameter associated with the generated sequence. Longer sequences result in larger search space for possible attack vectors. Specifically, the number of every possible attack vector can be calculated as $(P_X * P_Y * T)^l$ where $P_{X,Y}$ are the number of possibilities for X and Y, T is the difference between min and max used for a relative time, and l is the sequence length. As shown, the length of the substrate has an exponential effect on the number of possible vectors.

3.3.1 Results. Our brute force attack on bot traces resulted in successful attack vectors, manipulating the model to classify modified input as human. Figure 3 presents the number of successful attack vectors for three different sequence lengths using a relative time between 0 to 40 selected based on manual analysis of human traces. As illustrated in the Figure, the number of successful attack vectors increases for a specific range for each sequence length. Additionally, it can be observed that the number of successful attack vectors is significantly higher for longer sequence lengths. The reason for this is twofold: First, the model is trained based on a less diverse or insufficient number of long sequences which makes the model overfit to specific patterns. Second, by using longer sequences, we are increasing the attack surface which provides more opportunities for an attacker to successfully exploit the model.

4 DISCUSSION

The transition from generating adversarial inputs to translating them into actual interactions with a webpage necessitates the use of an automation tool. An ideal choice is a browser automation tool that mimics interactions closely resembling human behavior. To explore the feasibility of converting an adversarial trace into authentic browser interactions, we employed Selenium [13] and Puppeteer [12]—two widely recognized browser automation tools.

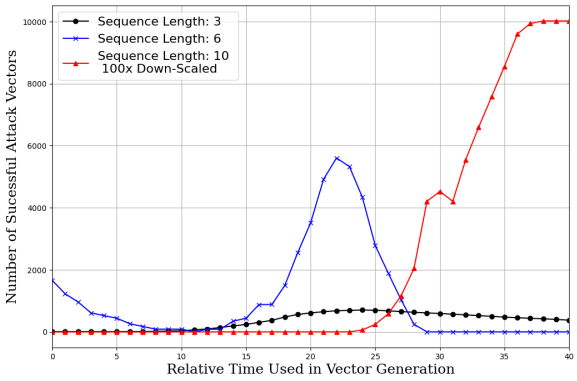


Figure 3: Number of successful adversary inputs using different sequence lengths. Longer sequence lengths result in an exponentially higher number of successful attacks.

As highlighted in Section 3.3, our emphasis has been on spatio-temporal tampering, requiring the development of a tool capable of replicating the generated numbers for X, Y, and timing. Our investigation uncovered a noteworthy challenge: while generating a sequence of coordinates and defining velocity can be accomplished, achieving precise timings corresponding to the adversarial vectors is not straightforward. This is due to from two primary reasons:

Event Sequence Generation. Our empirical analysis suggests that generating the sequence of events using automated tools to bypass trained models is not trivial. While generating common events like mouse moves or clicks is possible, the exact number of recorded events highly depends on the tracker settings. In other words, a similar setting on two bots may result in a different sequence of events if polling time configurations on the server side are different.

Timing Constraints. The second and equally important limitation of automated tools is generating the exact timing for a successful attack vector. We found out that even if we set a specific number of milliseconds between two artificially triggered events, there may be a slight difference in the recorded relative time and the target timing. Furthermore, we observed that the relative time between two consecutive events cannot be smaller than a threshold because triggering each event using an automated script, requires at least a fraction of time to be completed. In our case, we were not able to get the relative time between two events lower than 12 milliseconds.

Future Directions. The described experiments are the early steps toward evaluating the effectiveness of emerging learning-based defenses against automated attacks. As adversarial machine learning techniques become more mature and their impacts get more consequential, the need for empirical methods to enhance corresponding defense models becomes more vital. As a follow-up work, we plan to extend the experiment by incorporating a larger set of human as well as bot datasets. This is a critical step to evaluate the generalizability of our findings and possible enhancements to construct adversarial examples in a real-world setting. We also plan to define possible directions for how adversarial examples can be integrated into the training pipeline – making the defense mechanisms more effective against possible evasion techniques.

5 CONCLUSION

In this paper, we present a framework that utilizes multi-modal artifacts collected from triggered browser events to successfully distinguish human visitors from bots. We explore two adversarial attacks aiming to make the model classify modified bot interactions as human traces. Despite the success of the FGSM attack, the generated input vectors were in the form of float numbers and not the correct form of discrete numerical values. Through a brute force approach, we achieve adversarial input vectors in the acceptable format. We find that longer subtrace lengths increase the success rate but demand exponentially higher computation. Finally, we discuss that generating the actual automated agents based on the crafted adversarial inputs is not a trivial task due to the timing constraints when implementing the agent with popular tools.

6 ACKNOWLEDGEMENTS

We would like to thank the anonymous reviewers for their thoughtful feedback. This project was supported by Microsoft AI Security, National Security Agency (NSA) under Grant No. H98230-21-1-0324, National Science Foundation (NSF) Grant IIS-2331908, and CITES/IUCRC Grant No. 2113880. Opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NSA or NSF.

REFERENCES

[1] Alejandro Acien, Aythami Morales, Julian Fierrez, and Ruben Vera-Rodriguez. 2022. BeCAPTCHA-Mouse: Synthetic mouse trajectories and improved bot detection. *Pattern Recognition* 127 (2022), 108643.
[2] Babak Amin Azad, Pierre Laperdrix, and Nick Nikiforakis. 2019. Less is more: quantifying the security benefits of debloating web applications. In *28th USENIX Security Symposium (USENIX Security 19)*. 1697–1714.
[3] Cara Malone. 2023. Online Payment Fraud: Market Forcasts, Emerging Threats and Segment Analysis, 2023-2028. https://www.juniperresearch.com/researchstore/fintech-payments/online-payment-fraud-research-report/.
[4] Zi Chu, Steven Gianvecchio, and Haining Wang. 2018. Bot or human? A behavior-based online bot detection system. *From Database to Cyber Security: Essays Dedicated to Sushil Jajodia on the Occasion of His 70th Birthday* (2018), 432–449.
[5] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).
[6] gremlins.js: Monkey testing library for web apps and Node.js. 2020. https://github.com/marmelab/gremlins.js/. Accessed: 01-16-2023.
[7] hCaptcha. [n. d.]. hCaptcha Documentation. https://docs.hcaptcha.com/.
[8] Alex Huang, Abdullah Al-Dujaili, Erik Hemberg, Una-May O'Reilly, et al. 2018. Adversarial deep learning for robust detection of binary encoded malware. *arXiv preprint arXiv:1801.02950* (2018).
[9] Luis A. Leiva and Roberto Vivó. 2013. Web Browsing Behavior Analysis and Interactive Hypervideo. *ACM Transactions on the Web* 7, 4 (2013).
[10] Arbena Musa, Kamer Vishi, and Blerim Rexha. 2021. Attack analysis of face recognition authentication systems using fast gradient sign method. *Applied artificial intelligence* 35, 15 (2021), 1346–1360.
[11] Hongfeng Niu, Ang Wei, Yunpeng Song, and Zhongmin Cai. 2023. Exploring visual representations of computer mouse movements for bot detection using deep learning approaches. *Expert Systems with Applications* 229 (2023), 120225.
[12] Puppeteer. [n. d.]. https://pptr.dev/. Available: https://pptr.dev/.
[13] Selenium. [n. d.]. https://www.selenium.dev/documentation/webdriver/.
[14] Erik Trickel, Fabio Pagani, Chang Zhu, Lukas Dresel, Giovanni Vigna, Christopher Kruegel, Ruoyu Wang, Tiffany Bao, Yan Shoshitaishvili, and Adam Doupé. 2023. Toss a fault to your witcher: Applying grey-box coverage-guided mutational fuzzing to detect sql and command injection vulnerabilities. In *IEEE Symposium on Security and Privacy (SP)*. 116–133.
[15] Ang Wei, Yuxuan Zhao, and Zhongmin Cai. 2019. A deep learning approach to web bot detection using mouse behavioral biometrics. In *Biometric Recognition: 14th Chinese Conference, CCBR 2019, Zhuzhou, China, October 12–13, 2019, Proceedings 14*. Springer, 388–395.
[16] Jianli Zhou, Chao Liang, and Jun Chen. 2020. Manifold projection for adversarial defense on face recognition. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXX 16*. Springer, 288–305.