Captcha me if you can: Imitation Games with Reinforcement Learning

Ilias Tsingenopoulos, Davy Preuveneers, Lieven Desmet, Wouter Joosen
imec-DistriNet, KU Leuven
3001 Leuven, Belgium
{*firstname.lastname*}*@cs.kuleuven.be*

2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P) | 978-1-6654-1614-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/EUROSP53844.2022.00050

Abstract—**Since their inception, Captchas have been widely used as reverse Turing tests for combating bot proliferation on the web. This has resulted in an arms race between bot developers that automate Captcha solvers and Captcha services that adjust the challenges accordingly or come up with new ones altogether. Ultimately, older generations could be bypassed consistently, and thus in the third version of reCAPTCHA, Google offers zero user friction. The intent in the new system is not only to avoid interrupting user experience but to also obfuscate the nature of the challenge itself, being much less prominent than a text or image recognition task.**

We introduce a methodology that learns through interaction how to evade detection, while collecting and analyzing reCAPTCHA v3 scores over fifteen months and various web environments. With reinforcement learning as the backbone, we build models that can simulate human-like web browsing behaviour by using the returned score as an informative signal. Our study exposes an important vulnerability: while the score is influenced by a multitude of undisclosed factors, it is easily accessible and it enables adversaries to learn and perfect evasive models. Notably, we demonstrate that our automation models, which integrate general web browsing capabilities, transfer between websites with an evasion rate up to 99.6%.

1. Introduction

The prospect of automation reshapes our notions of work, productivity, and efficiency, as it bestows us with the capacity to simplify laborious tasks or turn them obsolete. Despite that promise, advances in automation and AI can also benefit abusive and malicious behavior in online activities, with examples ranging from social media bots [12], to botnets and DGA-based malware [3], and even botting in multiplayer games [10]

In the fight against the proliferation of bots and abusive automated solutions, “Completely Automated Public Turing test to tell Computers and Humans Apart” or Captcha for short has proven to be as a group of approaches [48] indispensable and widely adopted as a defensive solution. Such abusive or malicious activity is often motivated by direct financial incentives so there has always been a high demand in automated solutions – or poorly compensated human solvers – for Captcha. In response, Captcha developers have consistently tried to stay one step ahead in the competition between human problem-solving skills and their algorithmic approximation. This competition has elicited various fundamental and ad-hoc changes in Captcha design that can render a solver ineffective, as each solver is usually crafted for specific types of Captcha. The arms race has come a long way, with bot authors devising novel solvers and Captcha services initially gravitating from text-based to image-based challenges. Most recently, the interactive part of the Turing test – the challenge itself – is gradually being removed as it becomes progressively more arduous for humans to solve. After all, a Captcha challenge maintains its practical utility insofar as it is easy for humans but difficult for automated approaches to solve.

Nevertheless, the burgeoning capabilities of AI, particularly in computer vision and image segmentation, call the security offered by text and image based Captcha into question. Regarding text-based Captcha, it has been demonstrated that a generic solution can be used to solve new Captcha schemes without requiring an ad-hoc approach [8]. At the same time, automated solving has been shown possible also for the full range of challenges presented by Google’s reCAPTCHA v2, challenges that are primarily image based [40]. As automated solutions proliferate, the foundations upon which challenge-based Captcha is built seem unreliable.

A recent version of Google reCAPTCHA is v3 and the first to offer completely frictionless user verification by simply observing the user and the information they generate. The motivation in v3 is twofold, to minimize the interruption in user experience and at the same time conceal the exact nature and presence of the challenge: it is the next move in the imitation games. In place of an explicit challenge that calls for a solution, reCAPTCHA v3 service returns a score to the back-end based on interactions within the website, and enables the host to take appropriate actions based on that score. In the absence of an explicit challenge, the whole web session of the user is the challenge. In fact, v3 constitutes a paradigm shift over all previous approaches in Captcha solutions so far: it verifies users without interrupting them or presenting any challenge.

We set out to explore three key research questions in this work: Can we disentangle the information that reCAPTCHA v3 is recording to determine how it influences the final score? Can we evade reCAPTCHA v3 detection by simulating human behavior and if yes, under which conditions is the above possible? Finally, can we utilize reCAPTCHA v3 as an oracle in order to learn a general evasive model of online behaviour? Answering these questions positively would indicate significant vulnerabilities in reCAPTCHA v3.

To that end, we build an automation framework that

DOI 10.1109/EuroSP53844.2022.00050

is able to navigate the web in a GUI-enabled manner. As a first step, we host our own website protected by reCAPTCHA v3 as a testbed and in order to evaluate the extent that it can be exploited. Then we step into the wild; we scale up our experiments to two live websites not under our control with considerable daily traffic. Finally, we build a substitute model that mirrors the reCAPTCHA v3 risk analysis system and perform an explainability analysis to it in order to evaluate how different web browsing aspects contribute to the score and thus how adversaries could potentially exploit it.

Over an extensive period of manual and fully automated experiments, we show that it is possible to consistently avoid detection in widely varied web environments. Additionally, we show that it is possible for an adversary to exploit the implicit challenge itself in order to learn a web browsing behaviour with up to a 99.6% evasion rate. To actualize that, we leverage a state-of-the-art reinforcement learning (RL) methodology as the learning component of our automation framework, which is trained by interacting in an online manner with the websites in question, requesting verification, and using the score as an informative signal. The task and the environment are formulated in a manner that allows a general evasive model to be learned, something that can subsequently enable various automation attacks, from fraudulent transactions, to credential stuffing attacks and click fraud to generate ad revenue.

As the gap between human and bot agents diminishes, Turing tests are shifting towards continuous observation and verification, and any such test that can be consistently queried will in practice serve to further diminish this gap; bots are inadvertently created in our own image. While we propose potential defenses, these practically slow down this process as reliable and pervasive human verification on the web remains essential.

The contributions of our work can be summarized as follows:

- To the best of our knowledge, this is the first comprehensive study of reCAPTCHA v3 along with its advanced risk analysis system.
- We propose a novel RL-based automation framework that consists of three different types of agents, each more capable than the previous in simulating human web browsing behaviour.
- We collect and study verification scores over a period of fifteen months in various web environments, and perform a state-of-the-art explainability approach to quantitatively evaluate how different factors influence the score.
- We perform an extensive evaluation of our proposed approach over three different websites. The most comprehensive agent – one that incorporates navigation, typing and scrolling as capabilities – successfully transfers to third-party websites reaching an evasion rate up to 99.6%.

The remainder of the paper is structured as follows: Section 2 provides the necessary background on the domain and reviews the related work. Section 3 analyzes how reCAPTCHA v3 operates, from its source code to its web mechanisms. Section 4 describes the web environments and our automation framework. In Section 5 we elaborate on our experimentation and analyze our results. In Section 6 we discuss insights, limitations and challenges, before concluding in Section 7.

2. Related Work

Completely Automated Public Turing test to tell Computers and Humans Apart, or CAPTCHAs as they are more commonly known were initially conceived as a challenge–response test to determine whether or not an entity interacting with a website is human or machine. While there is controversy around who came up with the concept, which has been around since the late nineties [34], the term Captcha itself initially appears in the work of von Ahn et al. in 2003 [44]. What is of particular interest in this work is that it constitutes the first exploration of utilizing AI problems as security primitives, drawing parallels with prime factorization in cryptography.

The first generation of Captcha schemes was based around recognizing combinations of distorted characters. The straightforward assumption is that it would be easy for humans to recognize the characters and complete the challenge, but difficult for AI. Text-based Captcha also proved instrumental in digitizing old archives and books, as many of the challenges presented to the users were texts flagged by optical character recognition (OCR) as unreadable. In terms of individual characters, it has been demonstrated that machine learning (ML) algorithms consistently outperform humans in recognizing them [9]. The resilience of a text-based Captcha to automated solving therefore lies in the difficulty of segmenting it into segments containing individual characters [8]. As anticipated, this has led to an arms race between Captcha creators and automated solving attacks, by applying new distortions and creating new segmentation techniques tailored to these distortions respectively [13].

Around 2014, Google introduced a new reCAPTCHA mechanism purported to be more human-friendly and secure. This version of reCAPTCHA which is still active to this day is known as v2, while in 2018 Google retired reCAPTCHA v1 that was based on text character recognition; it was no longer fit for purpose as it was demonstrably more difficult to solve for humans than AI. Unlike its predecessor, the reCAPTCHA v2 challenge consists of two sequential parts: a system that performs the initial risk analysis, and in case this analysis flags traffic as suspicious, then the user will be prompted to solve a Captcha challenge. The challenges presented in the latter case are image recognition problems, e.g. the user is asked to select among a collection of pictures the ones that contain the designated object or property.

AI problems were not always used as fundamental primitives in computer security. In their seminal work, von Ahn et al. [44] assert that the advantages of using hard AI problems as a means for security are twofold. If the problem cannot be solved, then it remains as a reliable method for distinguishing humans from AI. If the problem can be solved, then that is by itself beneficial in the progress towards stronger AI. In the case of image- and text-based Captcha, contemporary AI solutions are capable of better than human segmentation and recognition in both images and texts [8], [24], [40]. As reliable human verification online is of essential importance and will not go away in the foreseeable future, we inquire as to how can these Turing tests stay one step ahead.

One such approach is Captcha based on game playing, also known as Dynamic Cognitive Games (DCG), that attempts to make Captcha solving a fun activity for the user [31]. Such solutions are still vulnerable to automated solving and additionally require a higher engagement from the user which can be frustrating while browsing. While hard AI problems are becoming easier to solve, we observe a paradigm shift in current Captcha solutions. When bot detection [16] and biometric authentication [18] transition from one-shot to continuous tasks, this accelerates the competition between detection and evasion. As bot detection systems start relying on behaviometrics and constant checks to detect bots, the continuous relaxation of the problem – in the temporal and decision domains – introduces a new vulnerability. The implication here is that when designing new Captcha, the paradigm is no longer a competition between incremental design adjustments and ad-hoc tailored bypasses/solutions, but two AI-enabled systems that engage in a competitive, online game, one's decisions becoming the other's information.

2.1. Evasion

Currently, there are several approaches available in order to evade or solve Captchas: employing ML in order to solve the challenge or exploiting bugs in the implementation to bypass the Captcha altogether. A third option is through websites like anti-captcha or 2captcha that employ cheap human labor [46]; such availability has led to the emergence of Captcha solving markets [32].

While bug exploitation is short-lived as bugs and workarounds are quickly patched, the immediate threat is posed by automated solving-based attacks. Bursztein et al. [8] introduced a generic text CAPTCHA-solving algorithm based on RL and asserted that text based CAPTCHAs schemes are inherently vulnerable. Their approach obviates any hand-crafted components, so it generalizes to new text-based CAPTCHA schemes, and assigns a score to all possible ways of segmenting the text. Being one of the first approaches to use RL in Captcha solving, they ask humans to annotate segments that have been misclassified and the RL algorithm learns from feedback and selects the segmentation that gives the highest score. This conclusion was reinforced by Ye et al. [50] which presented a deep learning-based approach that could solve with high success rate 11 different text Captcha schemes commonly used at the time by the top-50 popular websites.

Regarding image Captcha, Sivakorn et al. [40] conduct a comprehensive study of reCAPTCHA v2 by exploring how the risk analysis process is influenced by various aspects of the request: browsing history, Google account, geo-location and browser checks. They also design a module that upon receiving a challenge automatically extracts the images and passes them for annotation to widely used tools like Google Reverse Image Search and Clarifai. They also conclude that ML capabilities have reached a point where distinguishing between humans and bots through such challenges is unrealistic. While reCAPTCHA v2 is constantly adapted to account for new attacks, it is still vulnerable to this day. The latest work on attacking v2 is from Hossen et al. [20], where they propose an automated, object detection based attack, and successfully solve most (83.25%) of the image-based reCAPTCHA v2 challenges.

Furthermore, even audio challenges did not prove resilient to automated solutions, as the authors in [5] have developed an AI-based solution to solve audio reCAPTCHAs. A most recent work [15] systematizes the knowledge around the broad domain of automated Captcha solving that includes text, image, audio and video challenges.

2.2. Behaviometrics

Up to this point, Captcha challenges have proven inadequate to effectively differentiate between humans and bots as they can be consistently bypassed by automated solutions. This has led researchers to consider introducing behavioral biometrics, or behaviometrics for short, into the Captcha challenge. The idea to utilize behaviometrics to differentiate between legitimate users and bots has existed for some time [11], [49]. Collecting behavior data is considered advantageous by merit of being unobtrusive while offering accurate identity verification. The first work to have successfully incorporated mouse behaviometrics as part of a Captcha challenge was by D'Souza et al. [16]. Specifically, they extract mouse features such as movement speed, click time, and angle of curvature, in order to build what they describe as Mouse Dynamic Signature (MDS) profiles. A Support Vector Machine (SVM) classifier is subsequently trained on such profiles that achieves near-perfect accuracy in bot detection. Behaviometrics offer several advantages over typical Captcha challenges as they do not interrupt user experience and conceal the nature of the challenge, a deterrence against automated solutions.

Another advantage of zero challenge Captcha is that since it does not interrupt the user, it can be continuously evaluated and thus over time and during a web session it can lead to more accurate detection. This property draws a direct parallel between such Captcha and active authentication, which is the process of continuously verifying a user based on their ongoing interactions with a computer. In [18], Fridman et al. present a methodology where they can continuously and successfully authenticate users based on multiple modalities: mouse and keystroke dynamics, as well as stylometry. The application might be different in that case, that is intruder detection, but the principle is readily applicable to the context of bot detection; telling a human apart from another is not that different from telling a human apart from a simulacrum. In the most recent work on behaviometric Captcha, Acien et al. [1] build a scheme that works solely on mouse trajectories, and they show that a mouse-based CAPTCHA can be a highly dependable bot detection system. Before attempting to break reCAPTCHA v3 though, we require an in-depth investigation and understanding of its mechanisms.

3. reCAPTCHA v3

Version 3 is one of latest additions in the range of Google's reCAPTCHA services, the first of the CAPTCHAs to achieve that without any user friction. It employs an advanced risk analysis system to generate a score ranging from 0 that is very likely a bot to 1 that is very likely a human, based on a host of information and interactions within a website. The back-end can request verification at any point or time a user takes a specific action within the website. At first sight, the transition from a conspicuous, distinct Captcha challenge to an obfuscation of where or what exactly is the challenge can be considered advantageous for both users and administrators, for a number of reasons: it does not interrupt user experience, it obfuscates the exact nature of the challenge, and in place of a binary decision it returns a more granular score that is delegated to the back end.

There is potentially another reason for the transition from one-shot, explicit challenges, to a model more akin to continuous authentication by observing interactions and generating intermittent risk scores. While AI might be constantly closing the "hard for AI - easy for humans problem" gap, von Ahn et al. [44] point out that no matter how small this gap becomes, as long as there remains one between the human and the AI ability with respect to some problem, this problem can be used as a primitive for security. Rather than asking the user to solve a problem once, we can ask them to solve the problem multiple times. Even better, if the problem does not require any explicit input from the user and does not interrupt them, it can be solved continuously.

As is the case with v2, the latest version of reCAPTCHA embeds a widget in the website that it protects while it offers granular and adaptive scoring to the web admins and frictionless and uninterrupted experience to the user. Naturally, to prevent analysis from third parties, the widget's JavaScript code is obfuscated. The JavaScript code from reCAPTCHA v2 has been de-obfuscated and has provided insights as to what information exactly is being recorded and forwarded to Google[1].

3.1. Source Code Analysis

The JavaScript code that implements reCAPTCHA v3, and that always kicks-off at page load, is polymorphic and protected by virtualization-based obfuscation. We managed to hook the debugger and witness the various events being logged and processed, yet we were unable to reverse their encoding or meaningfully control the final response. Our attempts were hampered further by multiple levels of timeouts and sanity checks, and while it was possible to directly manipulate some relevant variables, we did not manage to affect the score, most likely due to the variables being manipulated either at the wrong time or in the wrong way.

More specifically, we utilized the well-known Burp suite to capture the request responsible for sending the reCAPTCHA v3 JavaScript code back to the browser and, each time, insert a breakpoint in the code. The exact position of the breakpoint was determined via trial-and-error, as pausing in certain locations caused the multi-threaded code to become desynchronized. Subsequently we coarsely iterate through the code and see clear signs of session-related variables, such as the respective Navigator object, being manipulated. However, manually modifying the values of said variables did not have an observable effect on the score. By inspecting the named variables in the source code we identify a multitude of strings that relate to user browsing behavior; "mousedown", "mouseup", "mousemove", "mouseover", "mouseenter", "keydown", "keyup" and others. This information can pertain to functionalities separate from how the score is generated, or also used as a red herring.

While manipulating this information at a low-level by capturing requests and perturbing session variables proved ineffective, our low-level analysis yielded insights on how to affect the score through a higher level abstraction. The further prototyping of our attacks was facilitated by contrasting what is implicitly or explicitly monitored by the source code on the one hand and the wealth of information every user generates continuously and in principle unconsciously during a web session on the other. IP address, the presence of cookies, the user-agent string, all embody the invariable part of the session information collected. The variable part is user dynamic behavior – timings, mouse and keyboard dynamics – which has the potential to affect the score; the goal of our approach is to simulate such behavior so that it evades detection. After all, security by obscurity is a fundamentally flawed approach, so we proceeded to put our hypotheses to the test.

3.2. Website Integration & Evasion

Unlike reCAPTCHA v2 where the user is required to click the "I am not a robot" checkbox, v3 offers two ways of generating the token that is subsequently verified by Google. It can be generated either by binding the challenge to a button within the website, or by programmatically invoking it when the user takes a specific action while browsing the website. The latter offers higher granularity on where exactly does abusive behavior actually occur, and a detailed summary for these locations in Google's admin console. In order to adapt the risk analysis for each website, Google's admin console requests to specify an action name in each place that reCAPTCHA verification is executed, implying that separate statistics are kept for each such action. After the token is generated, the back-end forwards it to Google together with the website's sitekey in order to be verified. Finally, a score $s \in \{0.1, 0.3, 0.7, 0.9\}$ is returned to the back end, where the decision on how to act on it is transferred to the administrator. Based on the score, the context, and the adjustable threshold, various actions can be taken, from requesting further verification or downright blocking the traffic. This verification flow is shown in Figure 1.

ReCAPTCHA v3 was launched in October 2018 and while it is widely prevalent on the web, protecting at the time of writing 8.5% of the top 10K websites and about 1.1M in total[2], there has been a puzzling lack of research on it. One exception is by Akrout et al. [2], where they propose a methodology to bypass reCAPTCHA v3. However, there are some limitations that we intend to address through our work. The authors do not mention how they handle the lack of the "I am not a robot" checkbox – or of any other challenge for that matter – in reCAPTCHA v3; after all the goal in v3 is to offer completely frictionless,

[1] https://github.com/neuroradiology/InsidereCAPTCHA

[2] https://trends.builtwith.com/widgets/reCAPTCHA-v3

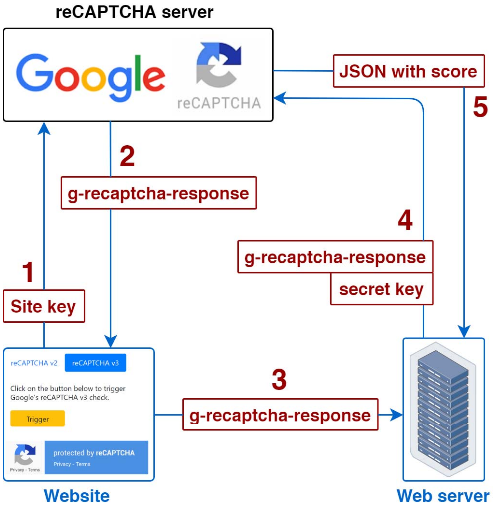

Figure 1: reCAPTCHA v3 verification workflow.

uninterrupted web experience. In the absence of such a transparent challenge and response interface that can be consistently queried, a methodology that endeavors to learn on it presupposes the definition and implementation of such an interface. Secondly, pixel by pixel mouse control and mouse trajectories as a composition of multiple smaller ones are insufficient to describe realistic, human-like mouse dynamics. Finally, mouse dynamics are just one aspect of human browsing behavior; we intend to explore a broader definition that includes other behavioral modalities as well as the effect they have on the score.

The research question behind [2] is compelling: given a black-box system that can differentiate between human and bot activity, an adversary can use it as an oracle – observe its decisions w.r.t. to their activity – in order to simulate a behavior that evades it. This capability has been widely investigated in the image classification domain, where black-box adversarial attacks against ML models are progressively more evasive and efficient [7], [21]. In text-based Captcha, Bursztein et al. [8] also rely on RL in order to solve the conundrum of segmenting the image into individual characters and classifying these characters concurrently. One reason RL enables optimal solutions to otherwise inscrutable problems is due to policy gradients, fundamental methods in RL that allow for the gradient-based optimization of non-smooth, non-differentiable objective functions [41].

In light of the above, the objective of our approach is to understand how the information collected influences the risk analysis score, to evade detection, and to confirm if a general, evasive model of online browsing can be learned by exploiting reCAPTCHA v3 as an oracle.

4. Approach

Having explored the manner in which reCAPTCHA v3 operates, we elaborate on the building blocks of our approach that will make evasion as well as learning possible. To that end we utilize three instruments:

- Three websites that incorporate reCAPTCHA v3 for bot protection, one self-hosted and two live with considerable traffic.
- The python library PyAutoGUI[3] to programmatically control the mouse and keyboard. We opt for PyAutoGUI as it is lightweight, works at system and not only at browser level, and to the best of our knowledge is the only library that offers fine-tuned control of user input – a prerequisite to construct human-like browsing. We also experimented with Selenium and we did not observe any negative impact on the reCaptcha scores; in that way it has the potential to work in conjunction with PyAutoGUI or even in its place.
- RL as a methodology to learn evasive browsing behavior, at multiple levels of resolution: controlling mouse trajectories, controlling timings, and finally controlling sequences of actions while browsing, e.g. navigating, scrolling, and typing.

4.1. Preliminaries

While the websites embedding reCAPTCHA v3 constitute the environment, we require a formal description of the RL agents behind our web automation; in this subsection we provide the necessary terminology and theoretical background. We consider web browsing as an environment where human or bot agents interact with, in sequential and episodic fashion. This web environment, that is the loaded web page together with the implicit Turing test performed by reCAPTCHA v3 can be modeled as a Markov Decision Process (MDP). MDPs formalize decision making in discrete, stochastic, sequential environments that change state randomly in response to action choices made by the agent. As reCAPTCHA v3 is a completely black-box system acting on a wealth of information that its precise nature we can only estimate, it is more appropriate to consider this environment as a Partially Observable Markov Decision Process (POMDP) [23].

RL is a mathematical formalism for learning-based control, suitable for solving MDPs and POMDPs. The objective is to learn optimal behavior – represented by policies – by optimizing reward functions while interacting with an environment. The reward function defines the goal of the agent, while the RL algorithm determines the way to reach it. The enhanced function approximation achieved through deep neural networks has allowed RL methods to obtain excellent results in a wide range of domains, from Atari and general game playing [30], [39] to various defensive approaches in cybersecurity such as cyber-physical systems [17], network attacks [29], smart grid security [35] and mobile edge caching [47]. In comparison to other approaches of ML, RL defines a fundamentally interactive learning paradigm that consists of an agent and an environment that interact for a number of turns.

Markov decision processes. In environments where outcomes are partially under the control of an agent and partially stochastic, MDPs are a discrete-time stochastic control framework for modeling decision making.

An MDP can be defined as a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is a set of states $\mathbf{s} \in \mathcal{S}$, which can be either discrete or continuous, $\mathcal{A}$ is a set of actions $\mathbf{a} \in \mathcal{A}$, which similarly can be discrete or con-

[3] https://github.com/asweigart/pyautogui

tinuous, $\mathcal{P}$ defines a conditional probability distribution $\mathcal{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ that describes the dynamics of the system, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ defines a reward function, and $\gamma \in (0, 1]$ is the discount factor.

A POMDP is an MDP where the agent is unable to observe the current state, and are thus defined by the extended tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \Omega, \mathcal{O}, \gamma)$, where Ω is a set of observations that are used to update the internal belief state b of the agent and $\mathcal{O}$ is a set of observation probabilities conditioned on actions and resulting states.

Objective. The objective in a RL problem is to learn a policy, that is a distribution over actions conditioned on states, $\pi(a_t|s_t)$. A sequence of states and actions of length N constitutes a trajectory and is given by $\tau = (s_0, a_0, ..., s_N, a_N)$. At each time t, the environment is in some state $s_t \in S$. The agent takes an action $a \in A$, which causes the environment to transition to state s_{t+1} with probability $\mathcal{P}(s_{t+1}|s, a)$. Depending on the environment and the learning task, this process can repeat in episodic fashion or continuously with intermittent rewards.

The reward $\mathcal{R}(s, a)$ is the signal that enables the agent's learning. The goal of the agent is to choose actions at each time step that maximize the expected future reward. This reward is a function J of the parameters θ of the policy π_θ that generates the actions a at each state s.

$$J(\theta) = \mathop{\mathbb{E}}_{s' \sim \mathcal{P}(s'|s,a), a \sim \pi(a|s)} \left[\sum_{t=0} \gamma^t \mathcal{R}_(s_t, a_t) \right] \quad (1)$$

where $\mathcal{R}_{s_t,a_t}$ is the reward earned at time t and γ determines how much immediate rewards are favored over more distant rewards. This function can become quite complex and usually is non-differentiable.

4.2. Websites

As a testbed for experimentation, we host our own website on a public web server, henceforth named **Website A**. Similar to other websites that use reCAPTCHA v3 to protect against bots, it embeds a variety of HTML5 form fields to offer similar opportunities for keyboard and mouse-based interaction. It includes text input fields and a password field to allow for collecting keystroke dynamics, as well as checkboxes and buttons to capture mouse dynamics.

We registered our website on Google reCAPTCHA, configured the site key and secret key provided by Google, and included the necessary JavaScript code in our website to capture the Google reCAPTCHA response. The front end invokes a JavaScript code snippet that uses the site key to retrieve the token generated by the Google server. This token is sent to the website back-end, which forwards it with the site key to the server for verifying the token and obtaining the score. To facilitate experimentation, our website embeds a 'Trigger' button and a log text area that displays the information that Google returns in JSON format.

A similar process is followed on **Website B** that faces diverse traffic with daily requests in the range of hundreds. After acquiring explicit consent from the website owners, we coordinated with the administrators and deployed reCAPTCHA v3 to it. In order to acquire the score that the back-end receives for our requests, as soon as it is returned by Google we embed it – encoded – in the front-end. Following Google's documentation, on this website we avoided influencing the risk analysis system during its training period by waiting for one week before initiating automated interactions.

Finally, **Website C** is a frequently visited website that gets 33K daily requests on average. This website had already integrated reCAPTCHA v3 for bot protection. After explicit consent from the website owners and a carefully planned experimentation protocol, we use it to evaluate our most capable model. The intention with this website is to represent the full black-box case, where an attacker has no knowledge other than a binary signal: if they are allowed to continue browsing or if further verification is required.

4.3. Environment

In the documentation, Google has established that the reCAPTCHA v3 score is based on interactions with the website. In order not to enable adversaries and bot developers, they will not divulge the extent of what exactly is being monitored – besides the named variables in the source code – or how it affects the score. This effectively conceals the game being played between the adversary and the detection system. Under this uncertainty, we need to formulate the structured form of rules that will represent this imitation game. Previous work [26], [40] identified a range of checks performed by reCAPTCHA v2 for detecting suspicious characteristics of the environment, as well as common automation tools that web bots use. Our intention is to avoid biasing the risk analysis system towards either bot or human indicators, i.e. avoid increasing its confidence state. The assumption is that in low-confidence states, dynamic behavior on the website will directly influence the score of a request. In light of the above and in order to imitate human-like behavior as close as possible, we obviate any web automation other than programmatically controlling the input devices.

The environment is composed by three types of information:

- The static information, like IP, user-agent, browser, cookies, etc.
- The dynamic information captured within the viewport of the website, like mouse or key events and timings.
- The decision that is returned when the web page submits a request for verification.

While many control problems take place in real time, in RL the temporal dimension is necessary to be discretized in order to make the problem tractable and create well-defined states where actions are taken from. To that end we fix the frames per second (FPS) to 50, which allows for good temporal granularity and smooth mouse trajectories without being too overbearing to the underlying library.

Our objective is to learn human-like web browsing behavior that evades detection. In order to successfully evade and subsequently learn from the risk analysis system, there is a plethora of variables to control thus equally as many degrees of freedom. We separate these variables into three categories: the ones that we keep constant, the ones that we stochastically vary, and the ones that are dynamically controlled by the agent. To achieve optimally evasive behavior, we defined three levels of control with expanding capabilities and generality:

- At the least-general level, the agent controls only the mouse trajectory, specifically the distance and angle of displacement at each frame. We call this the *Piecewise* mode.
- At the intermediate level, the agent controls the mouse trajectory and a range of timings – mouse hover, mouse press, and inter-request delay. We call this the *Bezier* mode.
- At the most-general level, the agent controls the mouse trajectory, the timings, as well as other capabilities like typing and scrolling. We call this the *Abstract* mode.

4.4. State-Actions-Reward

State. State representation in RL is a problem similar to that of feature representation and engineering in supervised or unsupervised learning. For an environment to be solvable by dynamic programming or a RL algorithm, it should exhibit the Markov property: the future is independent of the past given the present. This means that the current state representation should encode the history of information encountered so far and which knowledge is necessary in order to act on it. Formally, a state S_t upholds the Markov property if and only if: $P(S_{t+1}|S_t) = P(S_{t+1}|S_1, ..., S_t)$. Here, it is worth mentioning the difference between observation and state. In POMDPs a single raw observation might not offer enough information to constitute a suitable Markovian state, e.g. the speed that the mouse pointer is moving. To handle POMDPs there are two options: either utilize our domain knowledge in order to construct a better state from the available information, or use techniques that build missing parts of the state statistically, like a recurrent neural network (RNN). In the three different modes the states are constructed as following:

- *Piecewise* state is a tuple consisting of 5 values: $[a, b, x, y, v]$, where a, b are the initial relative coordinates, x, y are the current relative coordinates, and v is the current speed of the pointer in pixels per frame.
- *Bezier* state is a tuple consisting of 22 values: $[c_0, \dots, c_9, s_0, \dots, s_9, r, t]$ where $c_0, \dots, c_9$ are the cookie settings of the last 10 web pages visited – $c_n \in \{0, 1\}$ for cookies disabled and enabled respectively, $s_0, \dots, s_9$ are the last 10 scores received – $s_n \in \{0.1, 0.3, 0.7, 0.9\}$, r is the session progression and t is the time in day normalized – $r, t \in [0, 1]$. Session progression is calculated relative to the maximum number of requests per session.
- *Abstract* state is a tuple consisting of 20 values: $[s_0, \dots, s_9, a_0, \dots, a_7, r, d]$ where $s_0, \dots, s_9$ are the last 10 scores received – $s_n \in \{0.1, 0.3, 0.7, 0.9\}$, $a_0, \dots, a_7$ are the last 8 actions used – $a_n \in \{1, 2, 3\}$, r is the session progression and d is the delay between requests normalized – $r, d \in [0, 1]$.

Actions. Each agent mode has different sets of the actions that are being controlled and others that are defined stochastically. As for the temporal resolution: in Piecewise mode, one mouse trajectory corresponds to multiple steps and one episode; in Bezier mode one trajectory corresponds to one step and vice versa; in Abstract mode one trajectory corresponds to one step while one step can correspond to a trajectory, a scrolling action, or a typing action.

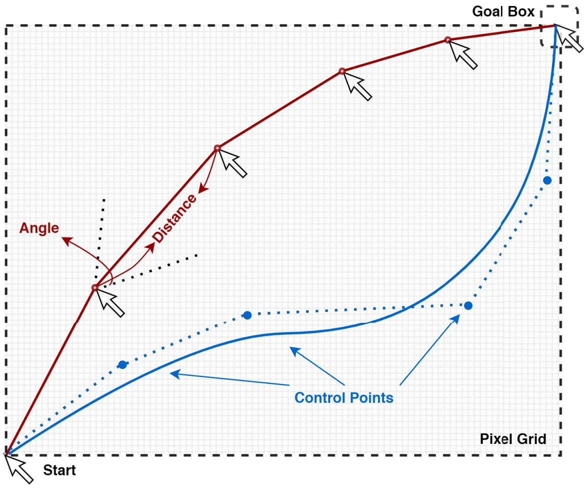

Figure 2: Agent controlled mouse movement. In red and blue are Piecewise and Bezier trajectories respectively.

Piecewise. We define a mouse trajectory from points $A \rightarrow B$ to be composed of individual linear segments $A \rightarrow a_1 \rightarrow a_2 \rightarrow ... \rightarrow B$. In order to define a segment for transition, two parameters are required: a) distance in pixels and b) angle in degrees. The direction of $0°$ is recalculated at each step as the line connecting the points between the current mouse position and the current goal coordinates.

We consider this parameterization of mouse trajectories fitting for several reasons: while simple and intuitive, sufficiently complex and human-like mouse trajectories can be generated and at 50 FPS the movement is smooth and plausible. Such an approach is also commonplace in stylometry, as a way to register and analyze mouse trajectories [18].

At each environment step, or actual frame, the agent based on the current state selects two continuous actions, distance $d, d \in [0, 1]$ and angle $a, a \in [-1, 1]$. These actions are scaled with the respective absolute values to calculate the actual distance and angle. For distance, the unit is $1/20$ of the initial distance between the two points. For less meandering trajectories and higher sample efficiency, we narrow down the solution space by constraining the angle in the range of $[-25°, 25°]$. Afterwards, the new position $[x, y]$ is calculated as well as the new distance to goal and the $0°$ direction. This process is illustrated in Figure 2.

Bezier. We define a mouse trajectory from points $A \rightarrow B$ to be along the Bezier curve connecting A and B that is generated by control points chosen at random in the range $[A_x, B_x], [A_y, B_y]$. We selected Bezier curves as they generate easy to parameterize smooth trajectories and are widely used in computer graphics and robotics [42], [45]; a good fit for human-like mouse control, if not too good. The agent controls 4 continuous values that are rescaled to the appropriate range. Duration $d \in [0.5, 1.5]$ that defines the trajectory speed, hover $h \in [0.2, 1.1]$ that is the amount of time after the trajectory ends and before the mouse is clicked, press $p \in [0.05, 0.55]$ that is the duration that the mouse button is pressed down, and delay $t \in [2, 10]$ which is the amount of delay between each step/request – listed timings are in seconds. A sample trajectory is shown in Figure 2.

Abstract. As in Bezier mode, in Abstract mode the agent controls 4 actions, with one key difference. In place of delay t, at each step the agent now selects between 3 actions to perform; that is scrolling, typing, or mouse control which is carried out exactly like in Bezier mode. All are parameterised by the duration, hover, and press values as before. The value ranges of these actions are determined by studies carried out in relevant literature [14], [25]. For more information on these actions we refer the reader to Appendix A.

Reward. The reward function embodies what we want the agent to learn or accomplish. In our case the score is affected by a multitude of factors and it can be a noisy signal to learn on, thus a simple reward mechanism would be most practical. For Bezier and Abstract modes, the reward is the difference between the last received score and the average cumulative score the agent has received in this episode:

$$r_{BA} = s_t - \overline{(s_{t-1} + \cdots + s_0)} \tag{2}$$

where s_t is the score at step t

In Piecewise mode, the reward is provided at the end of a trajectory/episode and is a weighted average of 3 terms: the difference between the current and last score, the amount of steps taken in order to incentivize precise trajectories, and a small constant term if the score remains high:

$$\begin{aligned} r_P &= s_t - s_{t-1} - 0.001 \cdot N_t + 0.01 \cdot h_t \\ h_t &= \begin{cases} 1, & \text{if } \geq 0.7 \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{3}$$

where s_t is the score at episode t and N_t is the number of steps in episode t.

4.5. Algorithm

To select an appropriate RL algorithm for the environment, we need to take the properties of the environment into consideration. As both state and action spaces are continuous, a value-based approach would be computationally intractable thus we opt for policy-based approaches [41]. Policy gradient methods optimize the policy $\pi_\theta(a|s)$ directly, where θ is usually approximated with neural networks, and they have been fundamental to recent breakthroughs in dynamical system control, 3D locomotion, and game playing. Put simply, the neural network learns the optimal policy directly, i.e. which action a to select in state s. From the range of policy-based approaches, we conducted our experiments with the Proximal Policy Optimization (PPO) algorithm, specifically the PPO-Clip variant. PPO [38] is a state-of-the-art approach in RL that solves many of the issues previous algorithms had, as it is simple to implement and tune, more general in the discovered policies, and has better sample complexity. PPO trains in an on-policy manner by sampling actions according to the latest version of its stochastic policy. This policy is updated by maximizing the PPO-Clip objective function. The reader is referred to Appendix B for further information on PPO-Clip, its architecture and hyperparameters.

The learning process for the Abstract agent is shown in Algorithm 1, where: n is the number of actions taken between each score request, u is the number of steps between updates to the neural network parameterizing the policy, M is the total number of episodes, T is the maximum number of steps in an episode, and a is the selected action. The pseudocode for Piecewise and Bezier agents is included in Appendix B. To navigate web pages and the system interface in general, all agents receive x, y values corresponding to exact screen coordinates. If the location within the screen is known in advance, coordinates are hard coded, otherwise they can be resolved through our browser extension that retrieves the location of HTML elements.

Algorithm 1: PPO-Clip Abstract

```
Set number of steps $n$ per request;
Set policy update frequency $u$ in steps;
for episode $e$ = 1, 2, ... $M$ do do
    Get initial score $S_0$;
    for step $t$ = 0, 1, ... $T$ do
        Select action $a$;
        Select duration $d$, hover $h$, press $p$;
        Perform action $a$ parameterised by $d, h, p$;
        if $t \pmod{u} == 0$ then
            Update policy and value networks
              based on Eq. 4 and Eq. 5;
        end
        if $t \mod n == 0$ then
            Get score $S_t$;
            Set $r_t$ based on Eq. 2;
        end
    end
end
```

5. Evaluation

In this section we elaborate on the evaluation we performed and report on the results. For the automated part of the experimentation, we use Firefox, Chrome, and Chromium as browsers, and a VPN service to shuffle our IP between browsing sessions. We attempt to control for the score bias introduced by privacy settings by defining two distinct settings between browsing sessions, that is with third-party cookies disabled and private/incognito browsing enabled – and vice versa. The intent is to represent the differing levels of privacy-aware browsing behavior and serve as one form of introduced variance in order to observe the effect on the score. Within each session, these configurations are kept constant as they represent the static, invariant aspect of browsing; our aim is to analyze how the risk analysis system is affected by controlling the interactive counterpart. To avoid any validity threats in our evaluation methodology, browsers did not retain any state or information between sessions: if private/incognito mode was not enabled, cache and cookies were cleared after each.

5.1. Empirical Evaluation

Throughout our experiments we spawn differently configured browsers on the same machine, so we investigated for potential tracking or cross-browser fingerprinting [6]. According to Google and Google Analytics documentation, Google can track unique IDs through their NID and SID cookies, with the stated intentions to authenticate users, prevent fraudulent use, perform targeted advertising, and others. For the average web user, an effective, practical albeit counter-intuitive method against web tracking is by releasing more private information rather than less, as privacy-aware policies can make a user stand out in the crowd. Previous work [40] studied the browser checks reCAPTCHA v2 performs, among them canvas fingerprinting [33]. These checks are used to determine the existence of web automation frameworks or the browser version, and discrepancies can be highlighted by comparing the results to the reported user-agent string. During our empirical and automated evaluation, we did not observe signs either of tracking or of a detrimental effect to the score when the reported user-agent string was manipulated.

Before performing the automated evaluation, the first step is to observe how the risk analysis system is affected by the various browser configurations, the presence of cookies, different IP addresses, and more importantly user behavior during browsing. We separate this information supplied to this system to its *static* and *dynamic* components. Keeping all but one of the features constant while varying others provides us with practical insights as to how each one of them affects the score that each request gets. More concretely we observe that:

- Google cookies (GRECAPTCHA, NID & SID) and enabled cookies in general have a decisive impact in the confidence of the system that the user is indeed human. This is to be expected, as cookies can track longer legitimate user activity. The experimentation with static features indicate that they affect the overall confidence of the system: when the confidence is high, dynamic features have little observable effect; conversely, when the confidence is low they appear to influence the score. The latter is also the setting we intend to induce and exploit in order to learn evasive human-like browsing behavior.
- Starting from a low score, something that occurs far more frequently with disabled cookies, we can consistently increase the score by engaging in typical browsing activity within the webpage, i.e. clicking, scrolling, typing in input fields, changing subpages, etc.
- Starting from a high score, it is possible to diminish it either by overloading the webpage with requests or by performing bot-like actions with mouse and keyboard automation.

The scores returned by reCAPTCHA v3 are coarse-grained and do not fall in a continuous range: there are 4 discrete scores that a request can get: $S \in \{0.1, 0.3, 0.7, 0.9\}$. A coarse-grained score offers two advantages in a Captcha system: (i) it simplifies for web administrators the process of drawing a threshold, and (ii) less information is divulged w.r.t. the model's decisions – akin to gradient masking in adversarial ML.

While high and low confidence refer to states of the risk analysis system that cannot be observed directly, a partial observation of them can be attained via the score behavior; high volatility in scores within a session indicates low confidence, while low volatility implies a high confidence state. Through our empirical observations, we further delineate two more states that can overlap to a degree with the high confidence state:

1) *Score saturation*. Saturation occurs whenever the risk analysis system has a high level of confidence that the user is human. This confidence may propagate to unrelated to the current session requests that come from different users. In saturation, the score is insensitive to user dynamic behavior.
2) *Score depletion*. Depletion represents a high confidence that the user is a bot where all session requests receive the minimum score. It can be caused by excessive requests – hence the reason why frequency of requests is part of the variables controlled – and it may propagate to unrelated requests irrespective of their source.

These states are extensively observed through the behavior of the score, in empirical observations and automated experiments. Naturally, from an adversarial perspective that intends to learn automated solutions a low confidence state is desirable and conducive to learning as it renders the score sensitive to user behavior. It then follows that high confidence states can be utilized also as a deterrence against intelligent adversaries; for reCAPTCHA v2 this is explicitly disclosed by Google. There is a prominent call in the Vulnerability Reward Program[4] to not submit reports when reCAPTCHA accepts invalid responses to challenges. It explains that in order to combat automated solutions, if the system determines you are likely a human – meaning it is in a state of high confidence/saturation – it accepts even invalid answers. Conversely, if it has high confidence that you are a bot, it rejects even correct answers. The stated reasoning is that by always accepting correct and always rejecting incorrect answers, the challenge for adversaries is greatly simplified.

5.2. Automated Evaluation

In all our experiments the automation is performed in a fully GUI manner – no headless browsers are used – and the simulation of user browsing behavior is performed by programmatically controlling the mouse and keyboard. Our experiments span a period of 15 months where more than 70K requests were submitted in total. During this period, we progressively refine our agents with more general capabilities in order to encapsulate abstract, human-like browsing behavior. To properly assess transferability and given the progressively more general agent capabilities,

[4] https://bughunters.google.com/learn/invalid-reports/google-products/4539275112349696

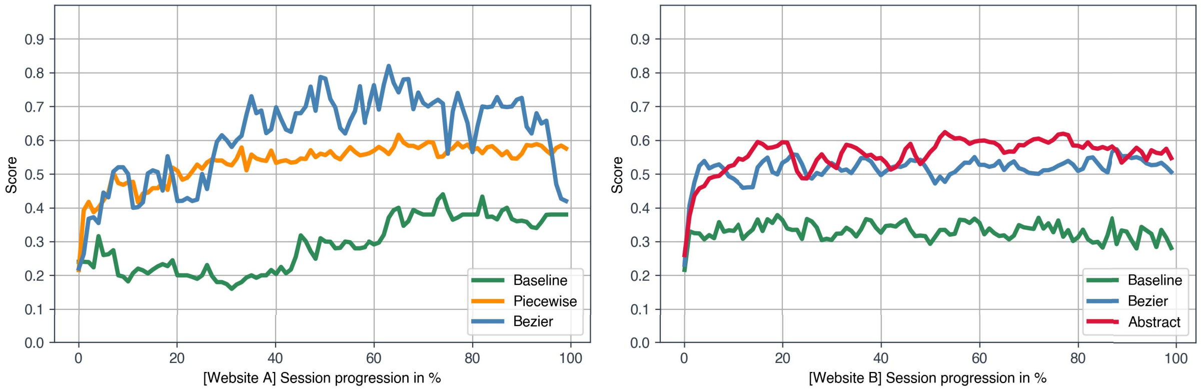

Figure 3: Score evolution in sessions initially flagged as bot, on Website A and Website B.

we conduct our evaluation in a cascading manner on three distinct web environments, as introduced in Section 4:

- Our self-hosted **Website A** which faces little actual traffic other than that from a few researchers, but where we had unconstrained request budget. Here Piecewise was in training *and* testing mode, while Bezier solely in training.
- External **Website B** which sees considerable web traffic. On this website, the Bezier agent trained on Website A is in testing mode to evaluate how well it transfers, while Abstract is in training mode.
- External **Website C** which sees extensive web traffic. This website already had reCAPTCHA v3 integrated for bot protection, and the intention here is to evaluate how evasive is the Abstract agent trained on Website B.

The practical difference between training and testing mode is if the parameters of the policy network are updated or not, i.e. if learning takes place or not. Next to the RL-enabled agents, we implemented and evaluated on every website a *Baseline* agent that differs in two key aspects: the mouse displacement is executed in a naive manner, i.e. either instantaneously or on a straight line connecting the origin and destination; additionally, no parameters are updated nor any learning takes place.

Website A. On this website, next to v3 we additionally deploy reCAPTCHA v2. The intention is to contrast their behavior throughout the evaluation. As the authors in [40] have demonstrated, when v2 has high confidence that the request is submitted by a human, it resolves to the “no Captcha reCAPTCHA” challenge that lets the user proceed without presenting a challenge. reCAPTCHA v2 never resolved to “no Captcha” mode while v3 was in depletion, but surprisingly many times it did not resolve to “no Captcha” mode even when v3 was in saturation. This indicates that the risk analysis systems behind the two versions of reCAPTCHA are potentially dissimilar in the way they process the information they collect as well as the risk profiles they build.

The main objective in this website is to have an initial estimation on how plausible the automated evasion of reCAPTCHA v3 is, but also whether we can learn from the score that the risk analysis system generates. To investigate the hypothesis that the scoring system can be exploited, we utilize the VPN service to intentionally look for and initiate sessions that start from a low score $s < 0.5$; more than two thirds of those sessions occur with disabled cookies and private/incognito mode. Statistically throughout our experiments, low-starting sessions exhibit higher score volatility (about twice as much) indicating low-confidence states more sensitive to interactions.

In Figure 3 we illustrate the score evolution averaged over all such sessions, for each methodology we employed. As sessions can have variable length, we normalize the time axis to a fixed $0 - 100\%$ range and we calculate the average over this range. There is a clear divide in performance between Baseline and both Piecewise and Bezier agents. For the latter two, the score has a clear upwards tendency that on average surpasses the typical detection threshold within 20-25% of a session’s duration. We observe a clear drop towards the end in the Bezier progression due to some sessions reaching more than 200 requests where the otherwise consistently high scores suddenly deplete to 0.1. This is strong evidence that the temporal dynamics of incoming requests are taken into consideration, especially since it occurs at the *same* time irrespective of which session we submit the requests from. Figure 4 plots the average cumulative score each methodology achieves over the period of evaluation. Once again there is a clear divide between the Baseline and the RL enabled agents, with Bezier being the approach getting most consistently high scores.

Website B. As website A is practically a synthetic one, albeit one that offers ease of access and unconstrained querying, it is probably not representative of the whole as the risk analysis system adapts by observing actual traffic. To properly evaluate the capabilities of our automation and put this to the test, we transition our experimentation to Website B as it faces actual live web traffic. As described in Section 4, in coordination with the administrators we deploy reCAPTCHA v3 on the website and wait for 1 week until it models the traffic patterns to avoid skewing them. In this setting, we evaluate how well the most performing agent trained on Website A transfers to the new context; that is without using the scores and without further re-training or fine-tuning. Additionally, we ascertain if an even more general and evasive model can be learned.

In Figure 3 we illustrate the score evolution averaged over all sessions that start below the bot threshold – 96 out of 180 – for each methodology we employed. We observe

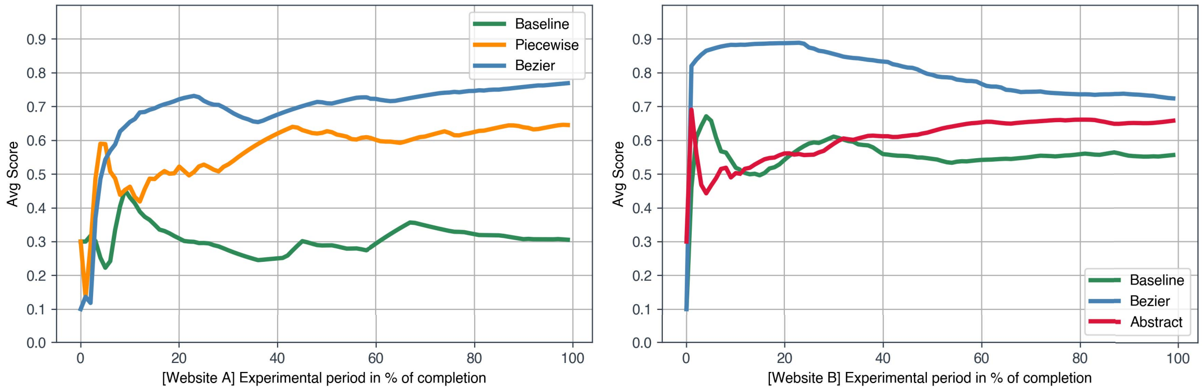

Figure 4: Average cumulative score over the period of evaluation, for Website A and Website B.

a clear but less prominent gap between the Baseline and the RL enabled agents. In this environment saturation and depletion still occur but differently than on Website A. While it is still possible to deplete the score due to an overabundance of requests, when it happens it concerns only the IP address where the requests were submitted from and does not propagate to others. The sessions where the score reached depletion due to excessive requests were less than 3% of the total.

In Figure 4 we illustrate the average cumulative score each methodology achieves over the period of evaluation. One thing we immediately observe is that Bezier performs the best out of the box, however Abstract diminishes the gap over time as the Bezier agent no longer updates its parameters. Another notable observation in this web environment is that the baseline is comparable to the RL methodologies. This indicates that even a rudimentary GUI-enabled automation can be sufficient to consistently evade reCAPTCHA v3 detection.

Website C. After concluding our experiments on Website B, we want to assess the level of threat our RL enabled automation poses on the web. The best candidate is the Abstract agent: it incorporates common browsing capabilities that so far were scripted if needed, and is thus the most general model we created. The next step is to evaluate its evasive performance and the degree this transfers to websites that get considerable traffic. Furthermore, our intention is to obviate any kind of learning and to transition to the fully black-box case. To that end we select Website C that compared to B gets two orders of magnitude more daily requests. In contrast to Website B, scores are no longer accessible. There is only a visible indicator that shows whether the score was above an unspecified threshold or not: the intention is to reflect attack scenarios on other websites where we have no access to the scores, only the binary verdict if further verification is required. Unlike previous websites where we track the average score, on Website C we report the percentage of requests that go undetected instead, denoted as evasion rate.

We evaluate the Baseline and Abstract agents in two principal contexts that aim to reflect and contrast the worst and best cases for the adversary: sessions that are initially flagged as bot (low starting) and not (high starting) respectively. Compared to A and B, it is exceedingly more difficult to find low starting sessions on Website C. The results are included in Table 1. We notice that for low starting sessions, the performance gap between Baseline and Abstract is quite prominent, while for high starting sessions less so. For high starting sessions, that adversaries have ample opportunity to initiate, the Abstract agent goes almost completely undetected with an evasion rate of 99.6%, while even the Baseline agent reaches an evasion rate of 84.3%. This demonstrates a stark decline on the level of bot mitigation that reCAPTCHA v3 offers, and a clear vulnerability that an attacker with enough resources can exploit.

5.3. Explainability

Despite the overall success that our approach has in evading detection and in exposing a vulnerability at the core of how such a bot detection scheme functions, some open questions remain:

- What is the risk analysis system behind reCAPTCHA v3 scoring exactly?
- Apart from showing the vulnerability itself, can we obtain a more precise understanding of what is happening within the black box?
- Which aspects of our online behavior influence the score, and to what degree each?

During the evaluation on Website B, we registered about 13K requests in total. From these requests, we generated a dataset where each instance contains the following information:

- The duration for which actions execute.
- The hover timing.
- The press timing.
- The delay between successive requests.
- The presence of cookies.
- The frequency of requests.
- The time in the day.
- The day of the week.
- The number of different actions used.
- The total number of requests submitted in current session.
- And the score that the request got.

In order to get insights into how does reCAPTCHA v3 score requests, an intermediary step is to extract a functionally similar model that shadows the decisions of reCAPTCHA v3 [4], [22]. The extensive data collection during our experiments provides such an opportunity. We train Random Forest classifiers and regressors [27] with the aforementioned features as independent variables and the score as the target variable. On the classification task we reach 87.5% mean accuracy and on the regression task we reach 0.82 R^2. In practical terms, an R^2 of 0.82 indicates that the model cannot explain 18% of the variability in the outcome. This is anticipated as Google has access to a different set of information that in our models cannot be controlled for. Examples are the reputation of IP addresses and ranges, deviation from heatmaps generated through Google Analytics for that particular website, and spatial, temporal or entropic aspects of user activity. The above are either missing or confounding variables, so we need to be aware that a statistical analysis can indicate but does not entail a causal relationship [37].

Our goal however is not to extract a high-fidelity model of the risk analysis system behind reCAPTCHA v3, but to get a glimpse in the black box and explain how various factors can influence the score; factors that come as a result of our initial assumptions. To that end we employ a powerful framework for explainable AI called SHapley Additive exPlanations (SHAP), a game theoretic approach to explain the output of any ML model. SHAP connects optimal credit allocation with local explanations using the Shapley values, i.e. the average of the marginal contributions of the features, across all the feature permutations [28]. We note here that by following an experimental protocol, bias is inadvertently introduced in the collected data. For instance, the effect of the presence of cookies will be exaggerated; we searched for low starting sessions, something that occurs considerably more often with disabled cookies.

Figure 5 demonstrates how each variable influences the score, in descending order from most to least influential. The presence of cookies is the most influential, indicating that the risk analysis system has a clear bias between more and less privacy-aware web browsing behavior. What is interesting though is duration at the second spot and that it is positively correlated with the score. This indicates that short or instant actions are biasing the score to lower values. Time of day and weekday are also influential; their influence is most probably overestimated as we unintentionally introduce some bias by following typical (mostly) office activity. The number of requests is counter-intuitive at first look; while a few high values are correlated with low scores, indicating depletion, the feature is mostly positively correlated with score, indicating that longer and human-like activity is a strong predictor of higher scores. The number of actions has a slight and positively correlated influence, corroborating our findings that put the Abstract mode a step above Bezier. What is unexpected here though is that mouse button press, mouse hover, and the delay between requests have little to no influence to the model. The value ranges of press and hover are by our definition commensurate to what an average human would have; from the model's perspective, they might as well appear completely uncorrelated to the score.

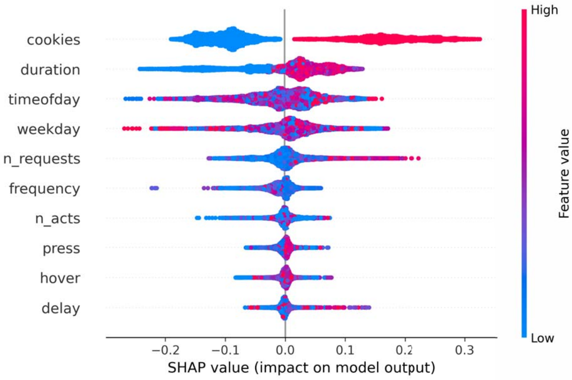

Figure 5: Shapley values for all features, depicted in descending order of importance

Computational cost. Throughout our experiments, the most demanding agent computationally was Piecewise, as it had to control actions at a rate of 50 FPS required for seamless trajectories. The computational bottleneck here is the neural network behind the agent policy, and the average inference time on an Intel i7-7700 CPU was 5×10^{-4} seconds which could theoretically support up to 2000 FPS control. Regarding the implementation and evaluation platforms, all scripts and RL agents were written in Python while a browser extension written in JavaScript was used to retrieve scores – all experiments were carried out on a dedicated desktop machine running Ubuntu 18.04.

In Table 1 we report the summary on the scores, numbers of requests, and overall traffic, across all experiments and for each of the websites. Over a period of 15 months, our RL enabled methodology successfully used the risk analysis system behind reCAPTCHA v3 to learn evasive models of web browsing, while submitting high numbers of requests commensurate to the total traffic a website attracts.

6. Discussion

In this section we critically reflect on the results and provide further insights into the topic. Captcha solutions are an essential tool to combat bot proliferation on the internet and the financial or other harm they may cause. From fraudulent transactions, to credential stuffing attacks and click fraud to generate ad revenue, an adversary would benefit immensely from general and human-like web browsing policies that evade Captcha detection. The ability to learn such a behavior by utilizing a popular, widely used Captcha service as an oracle, is a clear vulnerability that needs to be addressed as it can constitute the main component of a system that automates web attacks. Transitioning from one-shot Captcha challenges to overall behavior monitoring also has another implication. If human behavior and its imitation are indistinguishable to such a detection scheme, then this scheme – aside from being insufficient – enables the perfection of this imitation.

In January 2020 Google launched reCAPTCHA Enterprise, where this vulnerability could potentially be more severe. According to the documentation, reCAPTCHA Enterprise appears to be based on reCAPTCHA v3, with some modifications that intend it towards enterprises. Like v3, reCAPTCHA Enterprise will never interrupt users with a challenge, but it includes a pricing for every 1K requests submitted above 1M per month. Additionally

	Website A			Website B			Website C	
	Baseline	Piecewise	Bezier	Baseline	Bezier	Abstract	Baseline	Abstract
S_L	0.22	0.52	0.60	0.33	0.60	0.58	20.8%	70.1%
S_T	0.31	0.65	0.77	0.56	0.72	0.66	51.6%	90.0%
S_H	0.52	0.71	0.83	0.73	0.86	0.80	84.3%	99.6%
R_T	2.6K	21K	8K	6.3K	4.4K	2K	8.6K	12.4K
R_D	130	231	244	203	182	89	866	773
R_S	101	140	144	93	75	37	208	122
D_S	203	761	526	40	71	37	68	136
T_T	–	–	–	12.3K	9.4K	9.3K	328K	413K
T_D	–	–	–	398	390	421	33K	26K

TABLE 1: Requests per website and agent. S_L, S_T, S_H are the average score for low starting, total, and high starting sessions respectively. For Website C, S_L, S_T, S_H denote the evasion rate instead. R_T is the total amount of requests, R_D are the average requests per day, R_S are the average requests per session, D_S is the average session duration in minutes, T_T and T_D are the total and daily amount of traffic during the experiments.

potential customers have to supply personal and company information for a vetting process. A key difference to reCAPTCHA v3 is that now the scores returned include reason codes. These codes provide information on the reason reCAPTCHA Enterprise interpreted the interaction that way, in practice embedding explainability into the inference process. Such reason codes are:

- *automation*, when an automated agent is detected.
- *unexpected_environment*, when an illegitimate environment is detected.
- *too_much_traffic*, when the traffic volume from a specific source exceeds typical values.
- *unexpected_usage_patterns*, when interactions in the website diverge significantly from expected patterns.
- *low_confidence_score*, when there is insufficient traffic observed to generate a representative risk score.

These reason codes correspond to a surprising degree with several of our insights while investigating the risk analysis system of reCAPTCHA v3. An adversary with access to such reasoning behind the black-box decisions acquires a crisper signal to what effect their actions have and thus can optimize the parameters of an offensive methodology even more effectively. However, putting a premium on requests and putting customers through a vetting process can disincentivize that. This indicates a promising path for future work.

Ethics. Before deploying reCAPTCHA v3 on Websites B and C, we acquired explicit consent from the owners and administrators. Regarding the potential impact on these websites, an open question was how will our automated activity affect the scores of unrelated requests and overall user experience, even considering that a low score will not mean escalation to further verification in neither website. After all, depletion is a prominent phenomenon on Website A, so we wanted to be extra careful when we proceeded to actual websites with considerable live traffic. In contact with the website administrators, we set an initial conservative threshold on the amount of queries submitted to be in par with the daily traffic, and gradually increased it. In case of depletion we agreed to slow our requests down to a halt in order to investigate the cause and analyze the impact; something that ultimately did not occur.

Responsible Disclosure. Regarding the broader impact that our attack on Google reCAPTCHA v3 can have, we followed the standard responsible disclosure policy: we notified Google of our findings and the vulnerabilities we discovered in reCAPTCHA v3. Google responded to our detailed report and acknowledged the vulnerability. After further investigation on their side, Google has closed the issue without providing a fix and the status code “intended_behavior”. We surmise that acquiring access to backend scores and that GUI-enabled automation attacks are considered reasonable limitations on the adversary, however at the time of writing the issues described in this paper are still present.

Defenses. As our methodology does not tinker with the HTTP requests or cookies, defenses would fall primarily within the adversarial training paradigm [43]. In adversarial training, a model is trained also on data that were specifically created to fool it. The intent is to make the risk analysis system more robust towards adversarial activity. This however is a complicated matter as the domain of Captcha is considerably broader than image classification. No matter its abstraction or representation, the distribution of human behavior online is expected to be fairly diverse for adversarial examples – adversarial behavior here – to hide in the crowd. As adversarial examples are fundamentally out-of-distribution data [19], proactively modeling them can be a challenging task. There is however precedence in utilizing adversarial examples as Captcha resilient to automated solutions [36]. Finally, as long as one can obtain oracle access to the risk analysis system, defenses are limited to obfuscation and rate limiting. Adversaries can iteratively refine their attacks by using the oracle as either a discriminator to a generative adversarial network (GAN) approach or a reward function to a policy gradient method.

Limitations. The most capable agent demonstrates that it is possible to fully evade detection on a website of which it had zero beforehand knowledge or interaction with. However, in order to fully exploit the information leakage in reCAPTCHA v3, and in case the initial evasive policy is insufficient, an adversary would have to have access to the oracle for longer periods of time; something made evident by the evolution of scores in Figure 4. Furthermore, we posit that while the score returned by v3 is a signal that allows learning to take place, it is still one with noise, or as Google has indicated for v2, at times even intentionally misleading. If other heavily influencing factors are not controlled for, this innately limits the degree that learning can take place.

In many configurations, e.g. with low privacy settings, the baseline approach was sufficient to evade detection up until depletion. Beside the effectiveness and transferability of our approach, our results underline the importance of developing the scientific understanding of the strengths and limitations in the new paradigm of frictionless Captcha, specifically in the presence of adaptive attackers capable of learning from interaction with the deployed detection mechanism.

Controversies. While widely useful and an important pillar in the fight against bots and malicious users, Captcha does not come without its controversial aspects. reCAPTCHA was rightfully criticized for being a source of unpaid work to assist in transcribing text and image labeling tasks. As a competing solution, hCaptcha[5] raison d'être is to ameliorate this by returning some of the revenue made from solving the Captcha to the website owner. What can be disconcerting though is the above principle extended to frictionless Captcha such as reCAPTCHA v3, where all the interactions occurring in a website are monitored. As automated solving becomes ever more efficient and effective, Captchas are expected to transition towards behaviometrics, where there is no revenue to be made by transcribing text or labeling images. In this scenario, the users' online behavior could become the revenue, this time not to commercialize insights but as Turing tests.

Availability & Future Work. We opt for RL as a learning paradigm as it enables optimal behavior without access to gradients; the agent can learn in an immediate, online manner by interacting with the environment. Our approach and general framework, which we can make available to other researchers upon request, are modular by design to accommodate the learning of any parameters that might affect the risk analysis scoring system, in the case it adapts to observing a broader or different set of information. As a result, there are two main promising paths for future exploration: First, the possibility to train more capable agents by exploiting the reason codes of reCAPTCHA Enterprise. Secondly, to explore more effective defenses that go beyond rate limiting and security by obscurity, in the domains of adversarial ML and competitive multi-agent RL.

7. Conclusion

This work presents the first comprehensive investigation of Google reCAPTCHA v3 service. Over various websites, web configurations, attack methodologies and a period of fifteen months, we perform a black-box analysis and comprehensive evaluation of the risk scores reCAPTCHA v3 generates. We discover that it offers insufficient protection against web automation as we are able to consistently bypass it with an evasion rate up to 99.6%. We evaluate how different aspects of the activity on a website protected by reCAPTCHA v3 influence the score. Static aspects like privacy settings and IP address have a constant positive or negative impact on the score and do not change during a session. On the other hand, the score is also influenced by volatile aspects directly linked to the dynamic browsing behavior of the user. As reCAPTCHA v3 constantly monitors user interactions to differentiate between humans and bots by generating a risk score, we exploit this information to learn a general model of automated web browsing that consistently evades detection.

Our study indicates that while a transition towards Captcha services based on continuous behaviometric evaluation is unavoidable, as text and image Captchas have become more difficult for humans to solve than for AI, this inadvertently leads to vulnerabilities that dedicated adversaries can potentially exploit. The effectiveness of our approach demonstrates the need for improved bot and automation detection in zero friction, zero challenge Captcha. Nevertheless, as this type of Captcha are becoming more prominent, the competition in differentiating between human interaction and its imitation accelerates and calls for further research on proactive and reactive response to such adversarial activity.

Acknowledgment

This research is partially funded by the Research Fund KU Leuven, and by the Flemish Research Programme Cybersecurity. We want to extend our thanks to Stelios Tsampas and Vera Rimmer for the extensive discussions and insights.

References

[1] A. Acien, A. Morales, J. Fierrez, and R. Vera-Rodriguez, "Becaptcha-mouse: Synthetic mouse trajectories and improved bot detection," *arXiv preprint arXiv:2005.00890*, 2020.

[2] I. Akrout, A. Feriani, and M. Akrout, "Hacking google recaptcha v3 using reinforcement learning," *Conference on Reinforcement Learning and Decision Making*, 2019.

[3] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, and D. Dagon, "From throw-away traffic to bots: detecting the rise of dga-based malware," in *21st {USENIX} Security Symposium*, 2012, pp. 491–506.

[4] O. Bastani, C. Kim, and H. Bastani, "Interpreting blackbox models via model extraction," *arXiv preprint arXiv:1705.08504*, 2017.

[5] K. Bock, D. Patel, G. Hughey, and D. Levin, "uncaptcha: a low-resource defeat of recaptcha's audio challenge," in *11th {USENIX} Workshop on Offensive Technologies ({WOOT} 17)*, 2017.

[6] K. Boda, Á. M. Földes, G. G. Gulyás, and S. Imre, "User tracking on the web via cross-browser fingerprinting," in *Nordic conference on secure it systems*. Springer, 2011, pp. 31–46.

[7] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," *arXiv preprint arXiv:1712.04248*, 2017.

[8] E. Bursztein, J. Aigrain, A. Moscicki, and J. C. Mitchell, "The end is nigh: Generic solving of text-based captchas," in *8th {USENIX} Workshop on Offensive Technologies ({WOOT} 14)*, 2014.

[9] K. Chellapilla, K. Larson, P. Y. Simard, and M. Czerwinski, "Computers beat humans at single character recognition in reading based human interaction proofs (hips)." in *CEAS*, 2005.

[10] Y.-W. Chow, W. Susilo, and H.-Y. Zhou, "Captcha challenges for massively multiplayer online games: Mini-game captchas," in *2010 International Conference on Cyberworlds*. IEEE, 2010, pp. 254–261.

[11] Z. Chu, S. Gianvecchio, A. Koehl, H. Wang, and S. Jajodia, "Blog or block: Detecting blog bots through behavioral biometrics," *Computer Networks*, vol. 57, no. 3, pp. 634–646, 2013.

[5] https://www.hcaptcha.com/

[12] Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia, "Detecting automation of twitter accounts: Are you a human, bot, or cyborg?" *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 6, pp. 811–824, 2012.

[13] C. Cruz-Perez, O. Starostenko, F. Uceda-Ponga, V. Alarcon-Aquino, and L. Reyes-Cabrera, "Breaking recaptchas with unpredictable collapse: Heuristic character segmentation and recognition," in *Mexican Conference on Pattern Recognition*. Springer, 2012, pp. 155–165.

[14] V. Dhakal, A. M. Feit, P. O. Kristensson, and A. Oulasvirta, "Observations on typing from 136 million keystrokes," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 2018, pp. 1–12.

[15] A. Dionysiou and E. Athanasopoulos, "Sok: Machine vs. machine–a systematic classification of automated machine learning-based captcha solvers," *Computers & Security*, p. 101947, 2020.

[16] D. F. D'Souza, "Avatar captcha: telling computers and humans apart via face classification and mouse dynamics." 2014.

[17] A. Ferdowsi, U. Challita, W. Saad, and N. B. Mandayam, "Robust deep reinforcement learning for security and safety in autonomous vehicle systems," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 307–312.

[18] L. Fridman, A. Stolerman, S. Acharya, P. Brennan, P. Juola, R. Greenstadt, and M. Kam, "Multi-modal decision fusion for continuous authentication," *Computers & Electrical Engineering*, vol. 41, pp. 142–156, 2015.

[19] R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann, "Shortcut learning in deep neural networks," *arXiv preprint arXiv:2004.07780*, 2020.

[20] M. I. Hossen, Y. Tu, M. F. Rabby, M. N. Islam, H. Cao, and X. Hei, "An object detection based solver for google's image recaptcha v2," in *23rd International Symposium on Research in Attacks, Intrusions and Defenses ({RAID} 2020)*, 2020, pp. 269–284.

[21] A. Ilyas, L. Engstrom, and A. Madry, "Prior convictions: Black-box adversarial attacks with bandits and priors," *arXiv preprint arXiv:1807.07978*, 2018.

[22] M. Jagielski, N. Carlini, D. Berthelot, A. Kurakin, and N. Papernot, "High accuracy and high fidelity extraction of neural networks," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020, pp. 1345–1362.

[23] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[24] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3128–3137.

[25] T. Katerina and P. Nicolaos, "Mouse behavioral patterns and keystroke dynamics in end-user development: What can they tell us about users' behavioral attributes?" *Computers in Human Behavior*, vol. 83, pp. 288–305, 2018.

[26] X. Li, B. A. Azad, A. Rahmati, and N. Nikiforakis, "Good bot, bad bot: Characterizing automated browsing activity," in *2021 IEEE symposium on security and privacy (sp)*, 2021, p. 17.

[27] A. Liaw, M. Wiener *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[28] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proceedings of the 31st international conference on neural information processing systems*, 2017, pp. 4768–4777.

[29] K. Malialis and D. Kudenko, "Distributed response to network intrusions using multiagent reinforcement learning," *Engineering Applications of Artificial Intelligence*, vol. 41, pp. 270–284, 2015.

[30] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," dec 2013. [Online]. Available: http://arxiv.org/abs/1312.5602

[31] M. Mohamed, N. Sachdeva, M. Georgescu, S. Gao, N. Saxena, C. Zhang, P. Kumaraguru, P. C. Van Oorschot, and W.-B. Chen, "A three-way investigation of a game-captcha: automated attacks, relay attacks and usability," in *Proceedings of the 9th ACM symposium on Information, computer and communications security*, 2014.

[32] M. Motoyama, K. Levchenko, C. Kanich, D. McCoy, G. M. Voelker, and S. Savage, "Re: Captchas-understanding captcha-solving services in an economic context." in *USENIX Security Symposium*, vol. 10, 2010, p. 3.

[33] K. Mowery and H. Shacham, "Pixel perfect: Fingerprinting canvas in html5," *Proceedings of W2SP*, pp. 1–12, 2012.

[34] M. Naor, "Verification of a human in the loop or identification via the turing test," 1996. [Online]. Available: http://www.http://www.wisdom.weizmann.ac.il/~naor/PAPERS/human.pdf

[35] Z. Ni and S. Paul, "A multistage game in smart grid security: A reinforcement learning solution," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 9, pp. 2684–2695, 2019.

[36] M. Osadchy, J. Hernandez-Castro, S. Gibson, O. Dunkelman, and D. Pérez-Cabo, "No bot expects the deepcaptcha! introducing immutable adversarial examples, with applications to captcha generation," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2640–2653, 2017.

[37] B. Schölkopf, "Causality for machine learning," *arXiv preprint arXiv:1911.10500*, 2019.

[38] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[39] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[40] S. Sivakorn, I. Polakis, and A. D. Keromytis, "I am robot:(deep) learning to break semantic image captchas," in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2016, pp. 388–403.

[41] R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour *et al.*, "Policy gradient methods for reinforcement learning with function approximation." in *NIPs*, vol. 99. Citeseer, 1999, pp. 1057–1063.

[42] A. Tharwat, M. Elhoseny, A. E. Hassanien, T. Gabel, and A. Kumar, "Intelligent bézier curve-based path planning model using chaotic particle swarm optimization algorithm," *Cluster Computing*, vol. 22, no. 2, pp. 4745–4766, 2019.

[43] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," *arXiv preprint arXiv:1705.07204*, 2017.

[44] L. Von Ahn, M. Blum, N. J. Hopper, and J. Langford, "Captcha: Using hard ai problems for security," in *International conference on the theory and applications of cryptographic techniques*. Springer, 2003, pp. 294–311.

[45] D. Wang, M. Moh, and T.-S. Moh, "Using deep learning to solve google recaptcha v2's image challenges," in *2020 14th International Conference on Ubiquitous Information Management and Communication (IMCOM)*. IEEE, 2020, pp. 1–5.

[46] H. Weng, B. Zhao, S. Ji, J. Chen, T. Wang, Q. He, and R. Beyah, "Towards understanding the security of modern image captchas and underground captcha-solving services," *Big Data Mining and Analytics*, vol. 2, no. 2, pp. 118–144, 2019.

[47] L. Xiao, X. Wan, C. Dai, X. Du, X. Chen, and M. Guizani, "Security in Mobile Edge Caching with Reinforcement Learning," *IEEE Wireless Communications*, vol. 25, no. 3, pp. 116–122, jun 2018.

[48] X. Xu, L. Liu, and B. Li, "A survey of captcha technologies to distinguish between human and computer," *Neurocomputing*, 2020.

[49] R. V. Yampolskiy and V. Govindaraju, "Use of behavioral biometrics in intrusion detection and online gaming," in *Biometric Technology for Human Identification III*, vol. 6202. International Society for Optics and Photonics, 2006, p. 62020U.

[50] G. Ye, Z. Tang, D. Fang, Z. Zhu, Y. Feng, P. Xu, X. Chen, and Z. Wang, "Yet another text captcha solver: A generative adversarial network based approach," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 332–348.

Appendix A.
State and Action specifications

Piecewise agent. Our state construction ignores the absolute values of the mouse origin and destination coordinates and maintains only their initial relative position. A state extracted from a single frame with the coordinates of the mouse pointer and the destination coordinates is thus insufficient, as it does not include the speed that the pointer moves. Similar to frame stacking where RL agents play Atari games [30], the state representation here is a tuple that includes the current speed of the pointer in pixels per frame.

Abstract agent. The agent has an action space size of 4. At each step of the environment, one of three possible browsing actions can be executed: moving the mouse and clicking, scrolling, and typing. The agent selects which one to execute via action A_1, then the one selected is parameterised by actions A_2, A_3, A_4

Mouse control is identical to Bezier mode and all timings listed are in *seconds*. Scrolling is parameterised by duration d, press p, and a predefined number of scroll units n. If n is not provided, a random integer is chosen in the range 6-8. The agent pauses for p between each scroll unit and for d halfway and at the end.

Typing is parameterised by h and p, rescaled to fall in ranges pertinent to keystroke dynamics: $h \in [0.12, 0.16]$ controlling inter-key interval, $p \in [0.085, 0.125]$ controlling key press duration. Text can be provided as input, otherwise characters are typed from a dictionary at random.

Appendix B.
PPO Specifications

We follow the official implementation by OpenAI. The policy network of PPO-Clip is updated by maximizing the objective function shown in Eq. 4, taking multiple steps of mini-batch stochastic gradient ascent (SGA).

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|D_k|T} \sum_{\tau \in D_k} \sum_{t=0}^{T} min\Bigg(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} \cdot A^{\pi_{\theta_k}}(s_t, a_t), g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t))\Bigg) \tag{4}$$

where

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A, A \geq 0 \\ (1-\epsilon)A, A \leq 0 \end{cases}$$

and A is the advantage function, which is the difference between the Q value for a given state - action pair and the value V of the state: $A(s,a) = Q(s,a) - V(s)$.

ϵ is a hyperparameter that controls how far the new policy is allowed to change in comparison to the old. A similar approach is followed in approximating the value function that represents the value of each state, by minimizing Eq. 5 with stochastic gradient descent (SGD).

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|D_k|T} \sum_{\tau \in D_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t\right)^2 \tag{5}$$

Algorithm 2: PPO-Clip

```
Input: initial policy parameters $\theta_0$, initial value
  function parameters $\phi_0$;
for $k$ = 0, 1, 2, ... do do
    Collect set of trajectories $D_k = \tau_i$ by running
      policy $\pi_k = \pi(\theta_k)$ in the environment.;
    Compute rewards $\hat{R}_t$.;
    Compute advantage estimates, $\hat{A}_t$ based on
      the current value function $V_{\phi_k}$.;
    Update the policy by maximizing the
      PPO-Clip objective (Eq. 4) via stochastic
      gradient ascent with Adam.;
    Fit Value function (Eq. 5) by regression on
      mean-squared error with Adam.;
end
```

Algorithm 3: Piecewise Agent

```
Set request frequency $P$ in seconds;
Set policy update frequency $u$ in steps;
for episode $e$ = 1, 2, ... $M$ do do
    Observe the last score $s$ acquired;
    Retrieve new goal coordinates $x, y$;
    for step $t$ = 0, 1, ... $T$ do
        Select distance $d$ and angle $g$;
        Calculate new coordinates $a, b$ based on
          $d, g$ and move cursor to them;
        if $t \pmod{u} == 0$ then
            Update policy and value networks
              based on Eq. 4 and Eq. 5;
        end
        if $t$ is terminal then
            Perform left mouse button click;
            Set $r_t$ based on Eq. 3;
        end
    end
end
```

Algorithm 4: Bezier Agent

```
Set max requests $n$ per episode;
Set policy update frequency $u$ in steps;
for episode $e$ = 1, 2, ... $M$ do do
    Get initial score $S_0$;
    for step $t$ = 0, 1, ... $T$ do
        Select duration $d$, hover $h$, press $p$, and
          delay $f$;
        Perform a Bezier trajectory to goal
          coordinates $x, y$ with duration $d$;
        After hovering for $h$, press for $p$;
        if $t \pmod{u} == 0$ then
            Update policy and value networks
              based on Eq. 4 and Eq. 5;
        end
        Get score $S_t$;
        Set $r_t$ based on Eq. 2;
        Sleep for $f$;
    end
end
```

The pseudocode for PPO-Clip is shown in algorithm 2. The learning process for the Piecewise and Bezier agents are shown in Algorithms 3 and 4 respectively. Goal coordinates for Website A is the button that triggers the verification, and for Website B is the hyperlink for one of the subpages selected at random.

B.1. Architecture & Hyperparameters

For both actor and critic networks, we opted for a 2 hidden layer fully-connected neural network. The architecture and hyperparameters are shown in Table 2.

Hyperparameter	**Piecewise**	**Bezier**	**Abstract**
dense units	64, 32	64, 32	64, 32
activation	tanh	tanh	tanh
final activation	sigmoid	sigmoid	sigmoid
optimizer	Adam	Adam	Adam
action std	0.2	1	1
learning rate	0.0003	0.0003	0.0003
update	20	10	10
epochs	80	80	80
epsilon clip	0.2	0.2	0.2
gamma	0.99	0.9	0.9
max timesteps	100	100	200
max episodes	100-800	4	4

TABLE 2: Architecture and hyperparameters of PPO-Clip for Piecewise, Bezier, and Abstract agents.