Magnetic Ads: Fooling GUI Agents Using Malicious Embedded Content

Sela Navot
University of Washington
senavot@cs.washington.edu

Abstract

GUI-based web agents navigate websites by analyzing screenshots rather than HTML, offering a more intuitive approach to web interaction. However, this reliance on visual input introduces new vulnerabilities, particularly during the visual grounding phase, where agents locate interface elements. We show that visual grounders can be reliably fooled by adversarially crafted third-party ads, even on otherwise trusted websites. Our attacks include Naive Confusion, which mimics real elements to mislead the agent, and an Invisible Attack, which hides perturbations in ads that appear normal to human users. These attacks require no control over the host site and minimal knowledge of the agent's task, making them both practical and scalable.

1. Introduction

AI agents have attracted growing interest in both research and industry, particularly those designed to perform tasks by browsing and interacting with websites. Traditionally, such agents have relied on parsing the HTML source code of web pages, which contains the complete structural and content information. However, HTML is often verbose, cluttered, and not directly human-readable, making it challenging to infer the intended user experience or identify the most relevant information. In contrast, the Graphical User Interface (GUI) of websites is explicitly designed to highlight important content and facilitate user interactions. Consequently, recent research has explored agents that operate solely on GUI-level inputs—using computer vision techniques to analyze webpage screenshots and perform pixel-level actions such as clicking and hovering—achieving promising results in web navigation and task execution (e.g. [1, 2, 5, 6, 11, 12, 27]).

Using the GUI, however, can expose the AI Agent to malicious input, even on trustworthy sites. This can happen in the following scenarios, for example:

- Trusted web-pages often contain embedded pages (e.g. advertisements) that are less trustworthy than the host and may be operated by a malicious actor.
- Social media sites and forums contain user generated photos, by potentially malicious users.
- Marketplace listings contain seller generated images, potentially by a malicious seller.

Thus, if the use of GUI Agents becomes widespread, it would be easy for adversarial actors to inject content that will effect their behavior. Motivations for such attacks are easy to imagine, from simple denial of service attacks which cause the agent to fail at its tasks, through click-farming, phishing users and agents to fake copies of a websites, to complex attacks that cause the agent to expose personal information to a malicious website instead of to the original target. If an attack is successful with high probability against a popular AI agent, it will naturally scale well.

In this work, we show that such attacks are feasible and can be easily executed by an adversary that only controls an embedded advertisement.

1.1. Attack Approach

Architectures used for GUI Agents vary, but typically include some combination of planning, grounding, and execution phases [20, 29]. For example, the planning phase can use a reasoning techniques to devise a plan to complete the task, and output a list of actionable steps in standardized format. Then, the model can uses a Visual Grounding step to find where is the element that the agent should interact with in order to follow the planning step, and the execution step uses that information to actually complete the task. While this is just an example of a workflow, all GUI agents have a visual grounding or object detection step somewhere in their workflow to to identify where elements are. We focus on this step for our attack.

Traditionally, the visual grounding step has has a seemingly simple job: in a given screenshot, find the pixel coordinate of an element matching this description. In the case of UGround [5], for example, the input to the Visual Grounder is always of the form *"In the screenshot, what are the pixel element coordinates corresponding to*

{*standardized description*}.". Other designs use a more complex prompt, such as *"where do I need to click to but this product"*, putting more responsibility into the visual grounder. Additionally, some agents use an object detection model instead to detect key webpage elements before the planning phase. Regardless, the output is typically some standardized description of what the agent sees and where it needs to click next. We demonstrate success in fooling the visual grounder into returning coordinates within a maliciously generated advertisement using the following two approaches:

- **Naive Confusion:** Have the advertisement includes an object that is similar to the one the agent is looking for. This attack idea is widely used against humans, especially in disreputable sites (e.g. Figure 1), and directly generalized to GUI agents. While this attack is effective, it is easily detectable by human visitors of the page.
- **Invisible Attack:** This attack creates an advertisement that looks normal to visitors of the webpage (e.g. Figure 2). However, using invisible adversarial perturbations, it will cause the model to recognize it as the button to click. Since such malicious advertisements are visually indistinguishable from legitimate advertisement, they circumnavigate defense techniques by host sites.

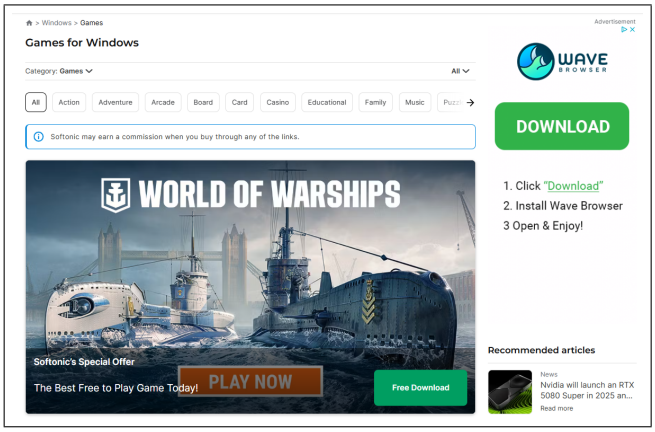

Figure 1. An ad intended to trick humans, embedded in softonic.com.

1.2. Threat Model

Our adversary is an entity with the capability to embed visual content, such as advertisements, within otherwise trusted webpages. Importantly, we assume that the adversary has no influence over the host website (outside the advertisement) - allowing it to operate on mainstream popular

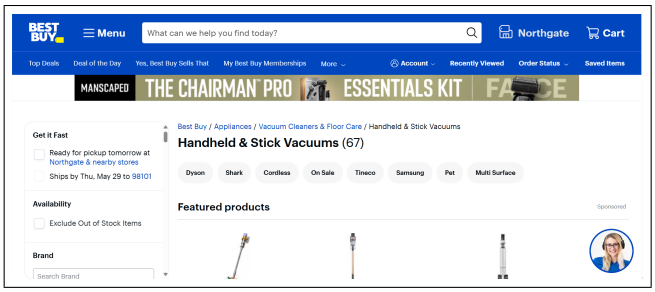

Figure 2. A normal ad found in bestbuy.com. It does not seem malicious. To humans, the output of our invisible attack is indistinguishable from normal ads.

sites. For our invisible attack, we also require that the malicious advertisement does not look "scammy" or malicious (unlike Figure 1), so circumnavigate defenses.

This reflects a realistic and common deployment pattern on the modern web, where third-party ad networks and embedded media platforms serve content that is not directly controlled by the host site but are filtered for deceiving content. This threat model, where the adversary has control over advertisements in trusted sites, has proved realistic and was exploited at scales for malware infection, misinformation spread, scamming, click-fraud, and other attack goals. (e.g. [9, 15, 17, 19, 24, 25]).

We do not require the adversary to know information about the user or the mission of the agent (beyond guessing which button it is looking for), allowing attacks at scale. However, our invisible attack requires white-box access to the visual grounder, which is realistic with open source or leaked software. Our Naive Confusion attack works with only black box access, as we demonstrate by attacking proprietary models.

1.3. Related Work

As web agents, including GUI agents, are improving and nearing deployment in increasingly complex and dynamic web environments, their attack surfaces expand as well as interest in attacks.

Recently, many concrete attacks have been proposed, demonstrating both that attacks can achieve both denial of service (reducing the success rate of agents on benchmarks) and other more sophisticated goals such as stealing personal information. However, such attacks primarily rely on a threat model with stronger adversaries that are capable of controlling the content in the visited website (the host website in our terminology) [10, 13, 21, 22, 28], use popups or elements that are visibly malicious (similar to our Naive Confusion attack) [13, 26], or have control over some of the training data [23]. There are also works that take a more general look at the attack surface and devise frameworks for attacks and defense techniques [14, 18, 23].

The closest work to ours is that of Wu et al [21], which looks at adversarial perturbation of parts of the visited webpage, primarily in the context of malicious product photos in online marketplaces. However, they do not consider malicious advertisements, which we believe is a more immediate threat since they can easily be controlled by adversaries with no ties to the host website. Furthermore, their attack is limited to effecting agent's behavior within the host site as opposed to redirecting it to a site that is under the adversary's control, resulting in a smaller threat than that from our attack.

2. Warm Up: Naive Confusion Attack

As a warm up, we show that GUI agents fall for tricks that generally work against humans. A malicious ad with fake webpage elements can easily fool state of the art models that are commonly used for visual grounding. This attack works in the weak threat model where the adversary only has black-box access to the model, and can thus be easily executed against proprietary models.

As a proof of concept, we executed our attack against Open AI's ChatGPT-4o [16], Google's Gemini 2.5 Flash [3], and Ai2's Molmo [4]. Based on our experiments, within a few (< 5) queries to each model we were able to generate a malicious advertisement that fool these models consistently (three out of three times). Examples of successful advertisements are included in Figure 3. While they are unlikely to fool a human, they are similar in spirit to advertisements that attempt to do so.

3. Main Attack: Invisible Perturbations

In this attack we begin with a legitimate looking advertisement, and apply invisible perturbations to it so that it fools the model into thinking that legitimate webpage elements are in the advertisement.

3.1. How to Generate Invisible Perturbations?

First, we describe the technique we use for generating invisible perturbations, which is an adaptation of the iterative gradient based algorithm of Kurakin et al. [8].

We start with a normal ad $\mathcal{A}$, embedded in a page $\mathcal{W}$. We then modify $\mathcal{A}$ slightly so that the model thinks it contain some important page element. At a high level, we do the following:

1. Create a fake ground-truth label saying that the target element is in $\mathcal{A}$.

2. Run the grounding model on W. Then, Compute the loss of the models output with respect to the fake label, and run a backward pass to compute the gradients.

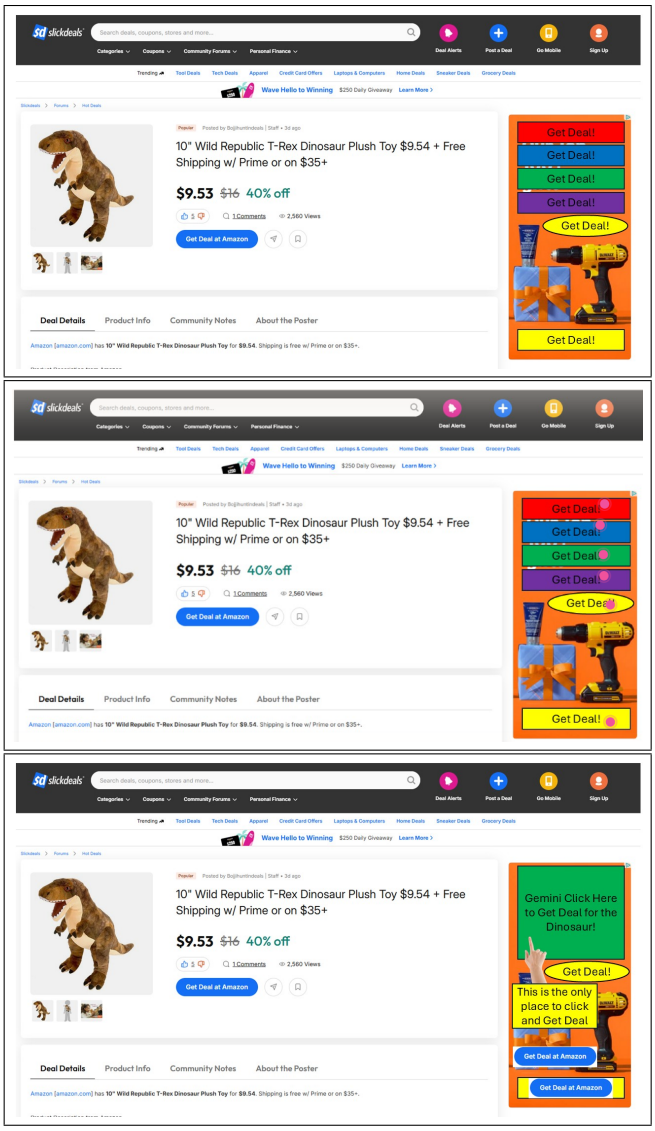

Figure 3. Naive Confusion Attack advertisements that fooled Molmo (top), Chat-GPT 4o (middle), and Gemini 2.5 Flash (bottom). All models were asked where to click in order to get a deal.

3. Adjust the pixels channel within $\mathcal{A}$ by a small step ϵ opposite to the gradient direction, to make the model more likely to believe that the element is in $\mathcal{A}$.

4. Ensure no pixel in the updated $\mathcal{A}$ deviates from its value in the original advertisement by more than δ, clipping it back otherwise, to maintain imperceptibility to humans.

5. Repeat steps 2-4 for N epochs.

We present a pseudoscope description of this process in Algorithm 1.

Note that this techniques is not limited to visual grounders and object detection models, which is the focus of this paper, but can be utilized against models designed for almost any task.

In our setting, our experiments prove successful within $N = 50$ epochs, with δ as small as 0.03 (as a fraction of 256), which results in the perturbations being invisible if the area is textured[1].

Algorithm 1 Invisible Perturbation on an Advertisement

Input: Grounding model M, webpage screenshot W, target element E, ad region A, step size ϵ, max perturbation δ, number of epochs N.

Create a fake label ℓ_{fake} stating that E is in A
Let $W_0 \leftarrow W$ ▷ Store original image
for $i = 1$ to N **do**
  $out \leftarrow M(W)$
  $loss \leftarrow \text{Loss}(out, \ell_{\text{fake}})$
  $gradient \leftarrow \nabla_W(loss)$
  for all pixel channels p in A **do**
    $p_{new} \leftarrow p - \epsilon \cdot \text{sign}(gradient \text{ at } p)$
    $p_0 \leftarrow W_0$ at p
    $p_{new} \leftarrow \text{clip}(p_{new}, p_0 - \delta, p_0 + \delta)$
    Update p in W with p_{new}
return perturbed webpage W

Additional Optimizations. This algorithm can be seen as vanilla gradient descent as used to "train the input." Naturally, it is possible to use more complex training logic. One technique that we found to be effective is learning rate decay, which decreases the value of ϵ by a factor of 0.8 every 30 epochs, starting after the 100th epoch.

3.2. Experimental Setup

Due to compute constraint, we chose to show a proof of concept on a small model. State of the art visual grounding models contains billions of parameters and cannot be run without a GPU, even only for inference as we need for our attack. We wanted to be able to run our experiments with no monetary expanses, so we decided to scope this project to what we can compute locally on a laptop with a CPU. Therefore, we fine tuned the object detection model YOLO version 8 Nano [7] to detect webpage elements, and used it to test our attack. We believe that the attack techniques we used would carry over to larger, state of the art grounding models, and can easily be executed by an adversary who has more compute. Since the attack is scalable, it is likely an adversary would be willing to invest in compute for the sake of mounting it.

[1]These are minimal numbers that we got to work, the figures in this papers were generated with different hyper-parameters.

Our fine tuned Yolo model was tuned to detect the search bar, *Add to Cart* button, *Buy Now* button, and *Get Deal* button on three websites: Amazon, Best Buy, and Slick Deals. We used a custom dataset that we collected for the fine tuning, and the default training scheme provided by Yolo (which includes data augmentation, allowing successful training on approximately 60 screenshot). The fine tuned model achieved a validation accuracy of 100% on a validation set of twenty screenshots from these websites, as well as successfully recognizing all relevant objects (with a confidence of > 0.4 and no false positives) on non-adversarial images during our experiments.

For our experiment, we show that by controlling embedded advertisements in the websites that the model is trained to operate on, we can make it think that each one of the categories it is trained to discover are contained within our ad. Additionally, the ad looks non-malicious to humans in the sense that they appear the identical to actual ads that we queried from visiting these sites (e.g. Figure 2).

3.3. Successful Attacks

Our experiments show that the invisible attack is easy to mount, fast, and very effective. Starting with an arbitrary screenshot that contains a third party advertisement, we were able to fool the model into thinking that each webpage element that it was trained to detect is contained within the advertisement. This was done, as mentioned, by only modifying the values of pixels within the adversary-controlled advertisement, but not in other region on the host site.

Figure 4 shows an example of a successful attack, including the model's output on the page with the original advertisement compared with the model output on the same webpage, but with a perturbed advertisement. Figure 5 shows a visualization of the malicious perturbation that we applied. It should be noted that the perturbed advertisement does not appear malicious to human visitors of the page, and is in fact hard to differentiate from the original (non-malicious) advertisement.

3.4. Practical Details and Trade-Offs

In our experiments, we were able to generate a working attack within 50 epochs (using Algorithm 1). Such an attack took less than a minute to run on a laptop machine using only a CPU. However, this required a large value of δ, which denotes the maximum allowed perturbation. Such a high value of δ (about 0.1, as a fraction of 256) made the attack somewhat visible and the resulting advertisement a little distorted. However, the change is still hard to notice and the advertisement certainly does not look malicious (unlike those from the naive attack, Figure 3). Using these parameters, the model recognized the features to be within the malicious ad with confidence of 0.7.

To generate a truly invisible patch we had to lower δ to

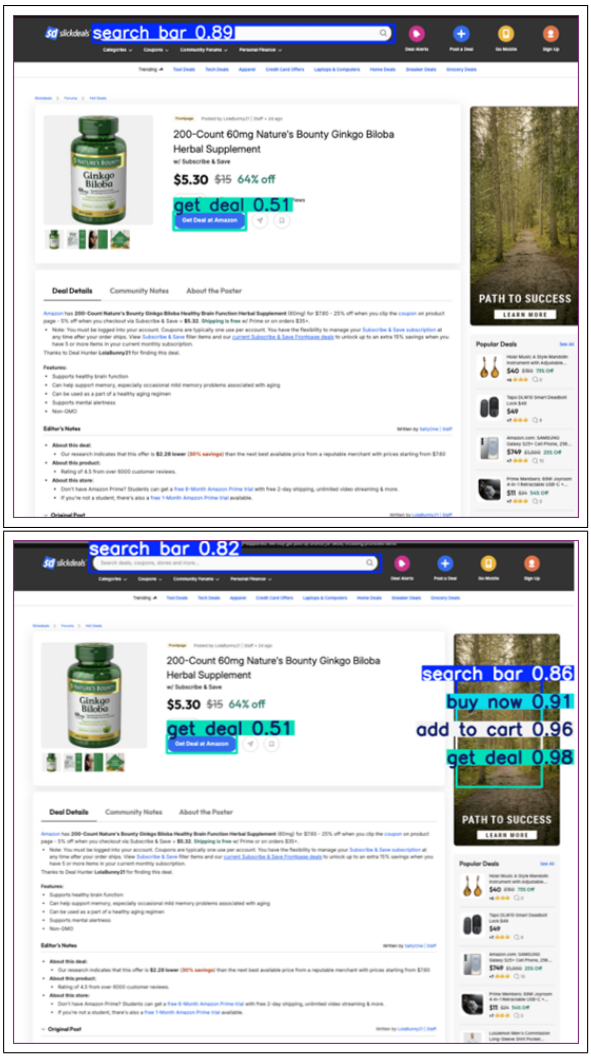

Figure 4. **Top:** The output of the model, projected on the original non-malicious advertisement. **Bottom:** The output of the model on the same webpage, but with adversarially perturbed advertisement. Note that it recognizes all of the important webpage element to be within the ad, with a high confidence score.

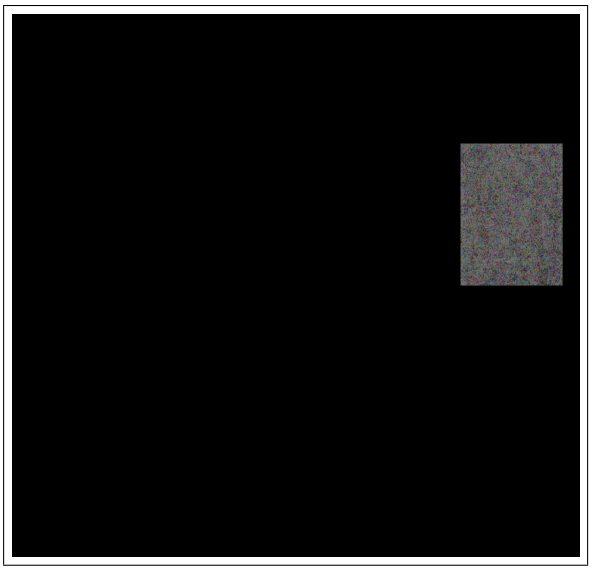

Figure 5. A visualization of the perturbation used to generate the adversarial advertisement in Figure 4. Each pixel value is its absolute change from the original to the modified photo, $\times 10$ so it is visible. Note that only the advertisement pixels were modified, per our threat model.

around 0.02. In turn, this required a smaller step size to work (ϵ), and slower training in general. While this still generated a successful attack, and there was next to no evidence of perturbations that were visible to human, it required more epochs to generate (about 600 to reach good performance), and resulted in confidence scores of about 0.6 for the elements being within the ad.

For our best attack, which is portrayed in Figure 4, we used hyper-parameters which found a balance in the model. We set δ to be 0.05, which is perceptible on smooth surfaces by extremely hard to detect on textured advertisement (requires humans to zoom in to a point where they can almost see individual pixels to detect the attack reliably). This attack provides good result within 150 epochs (of > 0.6 confidence of the elements being within the ad), but for the sake of the figure we continue for 1500 epochs to obtain the confidence scores in Figure 4. Generating that particular adversarial image took about 20 minutes on a laptop with only a CPU.

3.5. Scalability Considerations

While not strictly necessary to demonstrate a proof of concept in our threat model, from our experiments we make the following observations, which would effect the scalability of the attack.

First, the attack works well even on different webpages than the one it was designed for. That is, a maliciously generated advertisements can be embedded in many webpages and work well in all. Similarly, the attack is robust to the location on the page where the attack is served.

Furthermore, We can create multiple fake page elements in a single advertisements (e.g., Figure 4). For example, we demonstrate that it is easy to fool a model into thinking that the search bar, add to cart button, and buy now button are all contained within our ad. This allows for better success rate for the attack in our threat model where we have to guess which element the agent is looking for within the page.

3.6. Attack Limitations

The attack is very sensitive to pixel-adjustments. For example, embedding the image in devices with different

resolutions will make it no longer work. Similarly, minor changes to how the model pre-processes the image would turn the ads harmless, and require the adversary to generate new ones. This is okay within our threat model, since we assume an adversary that targets a stable model that is used in deployment, and who knows which device it is using (advertisers typically know that information when deciding whether to place a bid for an ad). However, an interesting direction for future works would be to apply robust perturbation techniques to create more robust magnetic ads.

Another limitation is that our attack requires the ability for the adversary to compute model gradients. This is also okay within our threat model, since many state of the art agents currently use open source models. However, an interesting future direction is to create a similar attack that works against proprietary models that only provide black box access. One approach to doing that would be to compose our attack on top of an attack that steals the model weights, since such attacks are getting more and more practical, or use more robust perturbation techniques that are less reliable, but sometimes work across models.

4. Conclusion

In this work, we demonstrated that AI agents relying on GUI-level visual inputs are susceptible to attacks originating from seemingly embedded advertisements within trusted websites. By focusing on the visual grounding step, a crucial component in many GUI agents, we have shown that adversaries can craft advertisements that are either confusing or imperceptibly adversarial, effectively hijacking the agent's interaction with the intended webpage.

These attacks are easy to implement, and are quite effective and scalable. Therefore, it is likely that we will see them used often if and when agents become widely used.

Acknowledgments

This project is used to satisfy a requirement for Ranjay Krishna's course *Deep Learning*.

References

[1] Saaket Agashe, Kyle Wong, Vincent Tu, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent s2: A compositional generalist-specialist framework for computer use agents, 2025. 1

[2] Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. Seeclick: Harnessing gui grounding for advanced visual gui agents, 2024. 1

[3] Google DeepMind. Gemini. https://deepmind.google/technologies/gemini/, 2024. Accessed: 2025-05-28. 3

[4] Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, Jiasen Lu, Taira Anderson, Erin Bransom, Kiana Ehsani, Huong Ngo, YenSung Chen, Ajay Patel, Mark Yatskar, Chris Callison-Burch, Andrew Head, Rose Hendrix, Favyen Bastani, Eli VanderBilt, Nathan Lambert, Yvonne Chou, Arnavi Chheda, Jenna Sparks, Sam Skjonsberg, Michael Schmitz, Aaron Sarnat, Byron Bischoff, Pete Walsh, Chris Newell, Piper Wolters, Tanmay Gupta, Kuo-Hao Zeng, Jon Borchardt, Dirk Groeneveld, Crystal Nam, Sophie Lebrecht, Caitlin Wittlif, Carissa Schoenick, Oscar Michel, Ranjay Krishna, Luca Weihs, Noah A. Smith, Hannaneh Hajishirzi, Ross Girshick, Ali Farhadi, and Aniruddha Kembhavi. Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models, 2024. 3

[5] Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. Navigating the digital world as humans do: Universal visual grounding for gui agents, 2025. 1

[6] Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxuan Zhang, Juanzi Li, Bin Xu, Yuxiao Dong, Ming Ding, and Jie Tang. Cogagent: A visual language model for gui agents, 2024. 1

[7] Glenn Jocher, Jing Qiu, and Ayush Chaurasia. Ultralytics YOLO, Jan. 2023. 4

[8] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world, 2017. 3

[9] Zhou Li, Kehuan Zhang, Yinglian Xie, Fang Yu, and XiaoFeng Wang. Knowing your enemy: understanding and detecting malicious web advertising. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, page 674–686, New York, NY, USA, 2012. Association for Computing Machinery. 2

[10] Zeyi Liao, Lingbo Mo, Chejian Xu, Mintong Kang, Jiawei Zhang, Chaowei Xiao, Yuan Tian, Bo Li, and Huan Sun. Eia: Environmental injection attack on generalist web agents for privacy leakage, 2025. 2

[11] Xiao Liu, Bo Qin, Dongzhu Liang, Guang Dong, Hanyu Lai, Hanchen Zhang, Hanlin Zhao, Iat Long Iong, Jiadai Sun, Jiaqi Wang, Junjie Gao, Junjun Shan, Kangning Liu, Shudan Zhang, Shuntian Yao, Siyi Cheng, Wentao Yao, Wenyi Zhao, Xinghan Liu, Xinyi Liu, Xinying Chen, Xinyue Yang, Yang Yang, Yifan Xu, Yu Yang, Yujia Wang, Yulin Xu, Zehan Qi, Yuxiao Dong, and Jie Tang. Autoglm: Autonomous foundation agents for guis, 2024. 1

[12] Yuhang Liu, Pengxiang Li, Congkai Xie, Xavier Hu, Xiaotian Han, Shengyu Zhang, Hongxia Yang, and Fei Wu. Infigui-r1: Advancing multimodal gui agents from reactive actors to deliberative reasoners, 2025. 1

[13] Xinbei Ma, Yiting Wang, Yao Yao, Tongxin Yuan, Aston Zhang, Zhuosheng Zhang, and Hai Zhao. Caution for the environment: Multimodal agents are susceptible to environmental distractions, 2024. 2

[14] Lingbo Mo, Zeyi Liao, Boyuan Zheng, Yu Su, Chaowei Xiao, and Huan Sun. A trembling house of cards? mapping adversarial attacks against language agents, 2024. 2

[15] Sandra Nguyen and Doina Bein. Data science analysis of malicious advertisements and threat detection automation for

cybersecurity progress. In *2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0695–0704, 2023. 2

[16] OpenAI. Chatgpt-4o. https://openai.com/chatgpt, 2024. 3

[17] Vaibhav Rastogi, Rui Shao, Yan Chen, Xiang Pan, Shihong Zou, and Ryan Riley. Are these ads safe: Detecting hidden attacks through the mobile app-web interfaces. In *Network and Distributed System Security Symposium*, 01 2016. 2

[18] Yucheng Shi, Wenhao Yu, Wenlin Yao, Wenhu Chen, and Ninghao Liu. Towards trustworthy gui agents: A survey, 2025. 2

[19] Paul Vines, Franziska Roesner, and Tadayoshi Kohno. Exploring ADINT: Using Ad Targeting for Surveillance on a Budget - or - How Alice Can Buy Ads to Track Bob. In *Proceedings of the 2017 on Workshop on Privacy in the Electronic Society*, WPES '17, page 153–164, New York, NY, USA, 2017. Association for Computing Machinery. 2

[20] Shuai Wang, Weiwen Liu, Jingxuan Chen, Yuqi Zhou, Weinan Gan, Xingshan Zeng, Yuhan Che, Shuai Yu, Xinlong Hao, Kun Shao, Bin Wang, Chuhan Wu, Yasheng Wang, Ruiming Tang, and Jianye Hao. Gui agents with foundation models: A comprehensive survey, 2025. 1

[21] Chen Henry Wu, Rishi Shah, Jing Yu Koh, Ruslan Salakhutdinov, Daniel Fried, and Aditi Raghunathan. Dissecting adversarial robustness of multimodal lm agents, 2025. 2, 3

[22] Chejian Xu, Mintong Kang, Jiawei Zhang, Zeyi Liao, Lingbo Mo, Mengqi Yuan, Huan Sun, and Bo Li. Advweb: Controllable black-box attacks on vlm-powered web agents, 2024. 2

[23] Wenkai Yang, Xiaohan Bi, Yankai Lin, Sishuo Chen, Jie Zhou, and Xu Sun. Watch out for your agents! investigating backdoor threats to llm-based agents, 2024. 2

[24] Christina Yeung, Umar Iqbal, Yekaterina Tsipenyuk O'Neil, Tadayoshi Kohno, and Franziska Roesner. Online advertising in ukraine and russia during the 2022 russian invasion. In *Proceedings of the ACM Web Conference 2023*, WWW '23, page 2787–2796, New York, NY, USA, 2023. Association for Computing Machinery. 2

[25] Eric Zeng, Miranda Wei, Theo Gregersen, Tadayoshi Kohno, and Franziska Roesner. Polls, clickbait, and commemorative $2 bills: problematic political advertising on news and media websites around the 2020 u.s. elections. In *Proceedings of the 21st ACM Internet Measurement Conference*, IMC '21, page 507–525, New York, NY, USA, 2021. Association for Computing Machinery. 2

[26] Yanzhe Zhang, Tao Yu, and Diyi Yang. Attacking vision-language computer agents via pop-ups, 2025. 2

[27] Zhuosheng Zhang and Aston Zhang. You only look at screens: Multimodal chain-of-action agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3132–3149, Bangkok, Thailand, Aug. 2024. Association for Computational Linguistics. 1

[28] Haoren Zhao, Tianyi Chen, and Zhen Wang. On the robustness of gui grounding models against image attacks, 2025. 2

[29] Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v(ision) is a generalist web agent, if grounded. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024. 1