A Red/Blue Video Game for Hide&Seek Analysis for Hardware Trojans and the Seeker's Dilemma

Peter Jamieson†, Obed Amaning-Yeboah†, Jared Butler†, Deniz Misirlioglu†, Riley Taylor†, Suman Bhunia*,
* Department of Computer Science and Software Engineering, Miami University, Oxford, Ohio, USA
† Department of Electrical and Computer Engineering, Miami University, Oxford, Ohio, USA
Email: {jamiespa,amanino,butler57,misirld,taylo550,bhunias}@miamioh.edu

***Abstract*—This paper describes our work in creating a video game to analyze the human dynamics of a Hide&Seek game. By harnessing human computation, we use this approach to get insights into how humans in a game Hide&Seek help understand how this relates to other problems, such as how Hardware Trojans might be hidden in an Integrated Chip. Hide&Seek, as delineated by Chapman *et. al.* [1], can be directly related to cybersecurity, and Sarihi *et. al.* [2] further describe Hardware Trojans in the perspective of The Seeker's Dilemma (an extension of Hide&Seek on a graph). Within the cybersecurity context, Hide&Seek is analogous to a continuous game of cat-and-mouse: the red player (the cat) assumes the role of the seeker, while the blue player (the mouse) acts as the hider. This study aims to model these games with human players and mathematical representations of Hide&Seek on a graph to collect data to better understand the hiding and seeking problems for a given graph. We describe the video game and provide results for a small sample of players hiding and seeking on three levels/graphs.**

***Index Terms*—Hardware Trojan, Human Computation, Hide&Seek**

I. Introduction

This work explores the dynamics of a unique Hide&Seek video game as a human-based simulation for cybersecurity practices, as defined by Chapman et al. [1]. Hide&Seek, within the context of cybersecurity, can be seen as an ongoing game of cat-and-mouse. The red player, the cat, takes the role of the seeker, while the blue player, the mouse, is the hider. This project aims to emulate a game theory model, "The Seeker's Dilemma," a mathematical representation of Hide&Seek on a graph [2] [3]. This problem, as related to adding Hardware Trojans (HTs) to Integrated Chips, is challenging. We seek to use our video game to analyze human approaches within this space to understand better how an intelligent agent might hide or seek.

In our model, a seeker is constrained by limited time resources, introducing a significant challenge in the game. The interplay of exploration versus exploitation is central to the game's strategy, requiring the seeker or hider to decide whether to exploit known information or explore for new information and potentially better alternatives. An essential question is whether the optimal strategy for the hider is to remain in a fixed location, and if they do explore, where would they go? Similarly, on the other side of the game, the hider chooses where to hide, and does this provide insight into where we might expect HTs to be hidden? To test this approach, we developed a two-player online Hide&Seek video game. Throughout gameplay, the system continuously records the positions of both players. This data generates a heatmap, visually representing the locations and durations of the hider and seeker at each spot. Analyzing these heatmaps can reveal insights into players' strategic behaviors when assuming the roles of seeker or hider.

We show the results of a small population of players who have played our game's three levels (each level is a graph). In this way, we are harnessing human computation [4] [5] to analyze behavior such that we can use the players' behavior to understand how we might create agents to solve the HT detection problem algorithmically.

The contributions of this paper are as follows:

- Development of a novel Hide&Seek video game to emulate cybersecurity scenarios.
- Implement a two-player online game to collect player positions and strategies data.
- Generation of heatmaps to visualize and analyze the behavior of seekers and hiders.
- Insights into the strategic decision-making processes involved in exploration versus exploitation within the game.

The rest of this paper is organized as follows: Section II reviews related work on game theory applications in cybersecurity and human-based simulations. Section III details the design and implementation of the Hide&Seek game, including the rules and mechanics, and data collection methods. Section IV presents the analysis of the gameplay data, including the generation of heatmaps and the insights gained regarding player strategies. Finally, Section VI concludes the paper and suggests potential directions for future research.

II. Background and Related Work

The creation of our Hide&Seek video game to understand human hiding and seeking behavior and its relation to Hardware Trojans (HTs) is an important link between what we call harnessing human computation and game theory in HT detection and insertion. We cover both of these ideas briefly in this section.

A. Harnessing Human Computation with Games

Harnessing Human Computation [4] and other types of productive play fall under the greater domain of games with a purpose (GWAP) (defined by von Ahn) and the general and contested term serious games. The first of these games was

the ESP Game [6], created by Luis von Ahn. Von Ahn and his colleagues and students have research articles on GWAPs [7], [8], [9], [10], [11]. There is a huge variety of Human Computation Games (HCGs), with popular examples being games such as Foldit, which has had tremendous success [12].

Focusing on the types of computational problems and algorithms in the graph-based meta-heuristic problems, this work focuses on games based on research such as Viglietta *et. al.* [13] [14] have looked at classic video games and have shown how complex these algorithms are in terms of complexity theory. Regardless of the problem's complexity classification, we confidently hypothesize that meta-heuristic algorithms (as surveyed and classified by Blum *et. al.* [15]) are an excellent interface point to think of as good game isomorphs that humans can work with computers to solve real-world problems. The links between graphs, metaheuristics, and games include a survey by Siu *et. al.* [16]. We have done previous explorations in this space with landscape generation [17] and the FPGA placement problem [18]. As is the focus of harnessing insight here, this is more related to the recent thesis by Gundry [19], which focuses on games to collect data.

B. Game Theory link for HT Detection

The Seeker's Dilemma [2] was established by linking directly to Chapman *et. al.* [1] definition of cybersecurity and Hide&Seek. HTs are unwanted modifications in the design or manufacturing of an Integrated Chip (IC) such that the IC's expected behavior is altered. Such modifications follow malicious goals such as denial-of-service or information stealing [20].

Kamhoua *et. al.* [21] pioneered the study of the HT dilemma as a zero-sum game between an attacker and a defender, aiming to find the optimal test batch that reveals HTs in the presence of an intelligent attacker using the Nash Equilibrium (NE). Their study claims robustness against irrational attacker scenarios as well.

Saad *et. al.* [22] incorporated uncertainty and risk in decision-making for both the attacker's and defender's behaviors by applying principles from prospect theory (PT) [23]. Das *et. al.* [24] also employed PT, but with a modification in the game, where the defender learns the attacker's strategies in the "learning stage" and utilizes this knowledge in the "actual game", enabling the attacker to "play dumb" and deceive the defender in the learning phase.

Brahma *et. al.* [25] investigated HT testing with a budget constraint on the testing process, considering single and multiple HT types. Nan *et. al.* [26] built upon this work by introducing human errors and biases in the insertion and detection processes. Gohil *et. al.* [27] examined attack and defense strategies in a split-manufacturing environment.

Regarding the Seeker's Dilemma [2] [3] as an HT detection game, our video game (as described below) focuses on creating planar graphs with a Hide&Seek gaming environment. The goal is to understand hiding and seeking human tendencies on a graph, and both the dynamic aspect is not present in actual HT detection, and the video game is a strict presentation of Hide&Seek where $k == 1$. We discuss these weaknesses later.

III. The Hide&Seek Game and Related System

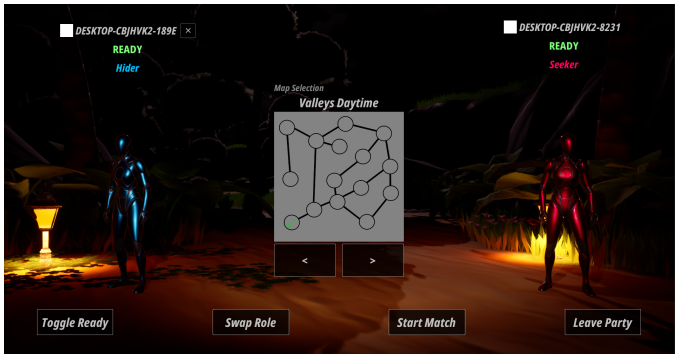

Fig. 1: Start screen of the game with the Valleys graph displayed in the center.

We created a video game where we can translate graphs (related to digital circuits) into maps on which a hider and seeker can play Hide&Seek. Figure 1 shows the screen before the game starts where the hider and seeker will see the graph (converted into a 3D map) on which they will play the game. In this section, we describe the game, some technical aspects of the game system, and the data collection capabilities built into the system.

A. Gameplay Framework

Our game is designed using the Unreal Engine. The game is run on a server, and players connect as clients. The gameplay framework in Unreal Engine is akin to classes in C++, where various class aspects facilitate shared information and communication between the server and client. Understanding how each class operates in a multiplayer setting is crucial due to replication challenges as gameplay scales. Adding new features increases complexity, and these classes help in maintaining modularity.

In Unreal Engine, the game mode defines the rules governing the game. These rules encompass player joining mechanisms, character selection, and game logic. This class is replicated only on the server, ensuring clients cannot alter the game rules unexpectedly. It dictates game operation and sets timers stored in the game state. Upon meeting certain conditions, it executes client replication events such as enabling and disabling movement and power-ups. This class also allows for game and variable resets.

The game state monitors the current game mode and the players' connections. This class is replicated for all clients, making functions and variables accessible to all users. It controls player spawn points, role selection, and movement definitions per the game's rules. Time-related variables are also defined here to ensure proper client updates.

The player state is vital for obtaining specific player information and their roles. This class is replicated for both servers

and clients. It produces player coordinate arrays that generate hider and seeker heatmaps (which we will describe below).

B. Creation of Heatmaps as Data

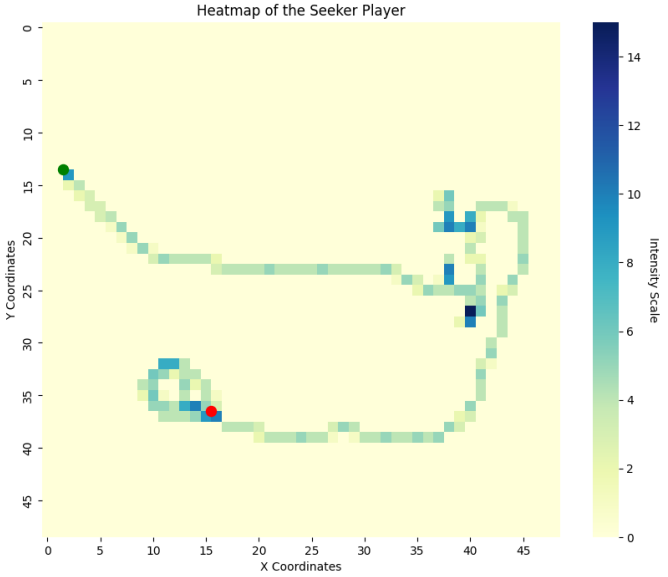

Fig. 2: Example 2D Heatmap of the seeker

Player coordinates are recorded in hider and seeker arrays stored in the player state on the clients at a rate of ten recorded positions per second. For simplicity, the z-axis (of the 3D game) is not recorded. The data is saved as comma-separated values (CSV), ensuring manageable file sizes and ease of post-data analysis. Python scripts provide graphing capabilities to create a visual heatmap representation from these coordinates. The Python script renders player movement within a game by generating a heatmap that visualizes player presence frequency and duration in various map areas. This heatmap aids in the following:

- Identifying strategic locations within the game.
- Balancing the game by modifying overused or underused areas.
- Enhancing player experience by understanding player behavior and preferences.

Figure 2 shows a seeker player's heatmap using a 'YlGnBu' colormap. Yellow represents the low density of the player position, green represents the medium density of the player position, and blue represents the high density.

To improve accessibility, the final heatmap was enhanced to a three-dimensional version, allowing intensity visualization by colors and elevation (Figure 3). This heatmap provides a more visually appealing and informative representation of seeker player positions in the Valleys map, where the z-axis allows for easy visualization of the player's high-density positions.

C. The Basics of our Game

In our final video game, we created three graphs Valleys, Garland, and GPIO as levels a hider and seeker can play on.

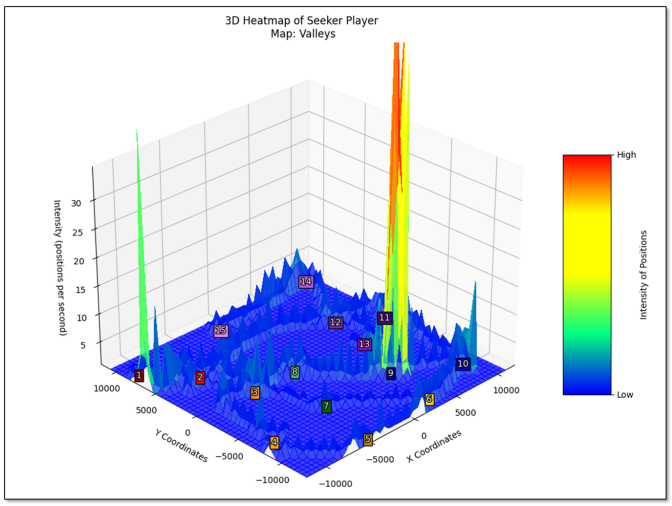

Fig. 3: An example 3D Heatmap of the seeker player in the Valleys map

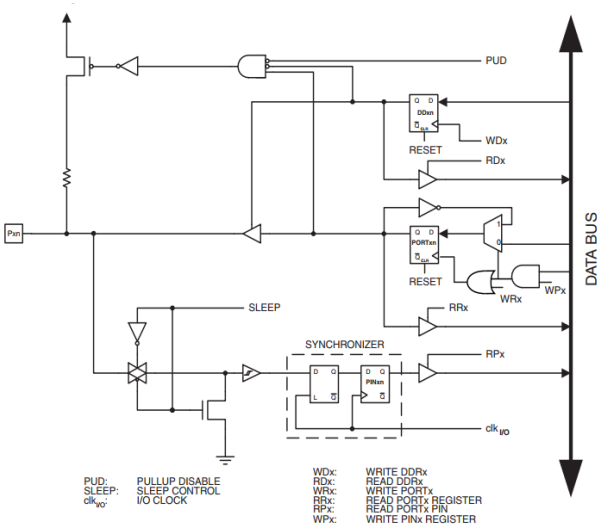

Fig. 4: GPIO circuit diagram from [28]

Valleys were the initial level that we developed our game with, and it is shown with colored nodes in Figure 5. The checkmark in node 1 shows the seeker's location to start the game. Level 2 is based on Miami's Garland Hall floor plan with faculty offices and classrooms. It is shown in Figure 8. Finally, to illustrate how a circuit can be implemented as a graph, we take the circuit diagram of the Atmega 328P microcontrollers GPIO circuitry (shown in Figure 4 and convert that circuit into the graph in Figure 11. The coloring of the nodes in each of these graphs will be used in our bar charts to make analysis easier.

Once a level has been selected and both the hider and seeker are connected to the server, a hide&seek game begins. Initially, the hider is given a few seconds to travel the 3D map, and then a countdown timer is started for the seeker to search for the hider. The game ends when the seeker enters the same node as

the hider or when the countdown timer reaches zero. During gameplay, both player locations are recorded.

IV. Results and Analysis

After data collection from a small player population of 10 pairs of seekers and hiders, in this section, we provide our aggregate data results and a basic analysis of player behavior. One aspect of our game, which is not as related to the HT detection problem, is that players' behavior adjusts with knowledge of other players' tendencies, leading to dynamic and strategic gameplay. Each player freely traverses the virtual environment, navigating the nodes and edges of the graph representation of the game map. The traversal process poses challenges for analysis, as players can reroute themselves while on an edge and potentially reverse their direction to a previous node. Despite these complexities, discernible patterns and strategies emerge, revealing the dynamic nature of the game's graphical representation.

For each level, we will briefly describe the level (with an associated graph), and then provide aggregate data for the hiders and seekers as bar graphs. For example, Figure 6 shows the aggregated data of the hiders on the level of the valleys. The x-axis shows each node (numbered and colored), and the y-axis shows the percentage of time (aggregated over all games) the hiders spent at that node.

A. Graph of Valleys

The Valleys graph is an abstract planar graph created with 15 distinct nodes, each featuring unique attributes (e.g., a mansion with two floors at node nine and a hideable bush at node fifteen). These features influence player position data, as players may take longer to traverse specific nodes.

Figure 5 depicts the Valleys map. Figure 6 shows the steady-state distribution for hiders, indicating a prevalent strategy of remaining near the initial nodes and frequently returning to the starting node. Node 9, with the most assets, is a popular hiding spot. Low-value nodes are typically transitioned through quickly.

Figure 7 shows the steady-state distribution for seekers who inspect each node. Time spent at each node scales with the number of remaining assets to be checked, with node 9 requiring more thorough searches.

B. Graph of Garland Hall

Garland Hall, representing a building at Miami University, features a non-planar graph with intersecting edges that include more nodes and longer edges than the other two maps.

Figure 8 shows the Garland Hall map. Figure 9 indicates that hiders often remain stationary at nodes furthest from the hider's starting location, contrasting with the Valleys strategy. For this level, we might hypothesize that the complexity of the graph and the lack of loops suggests that a hider can hide in complexity and hopefully wait for the seeker. Node 4, however, has the highest occupancy rate and is relatively close to the seeker's starting location. This shows that hiders have taken various strategies, sometimes hiding in the depths and trying to

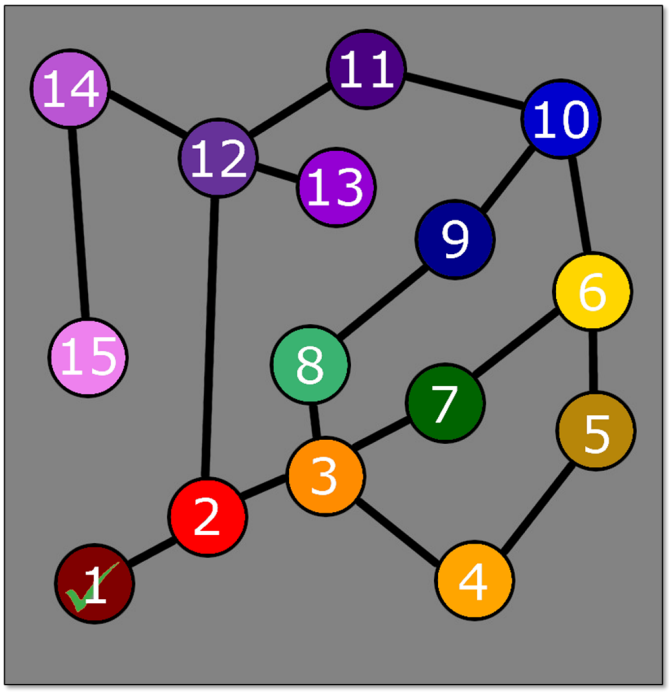

Fig. 5: Graph of the Valleys map.

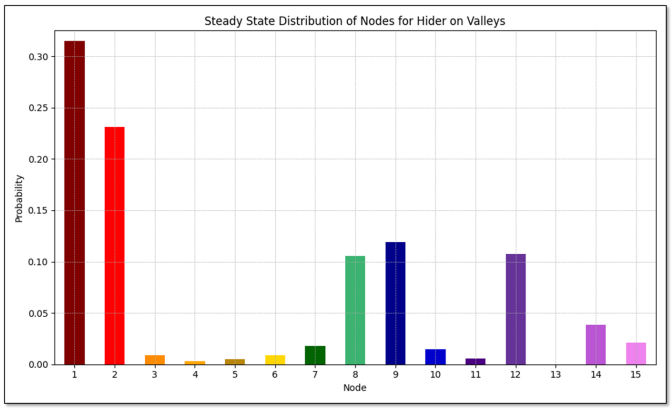

Fig. 6: Steady-state distribution of nodes for hider on Valleys.

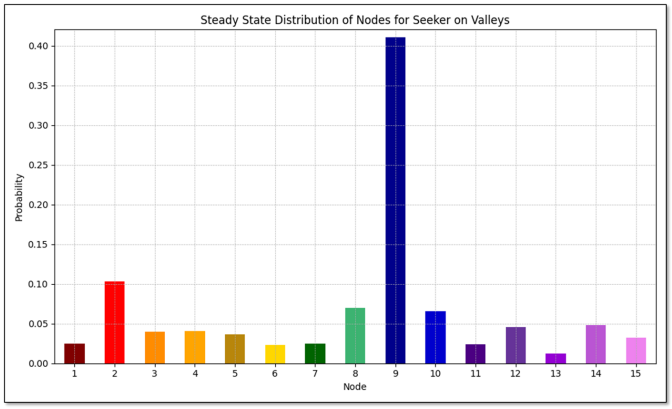

Fig. 7: Steady-state distribution of nodes for seeker on Valleys.

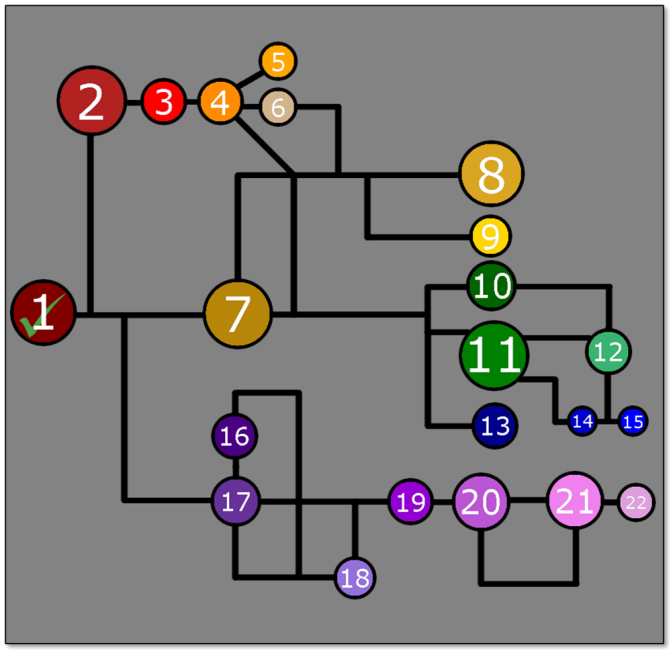

Fig. 8: Graph of Garland Hall at Miami University.

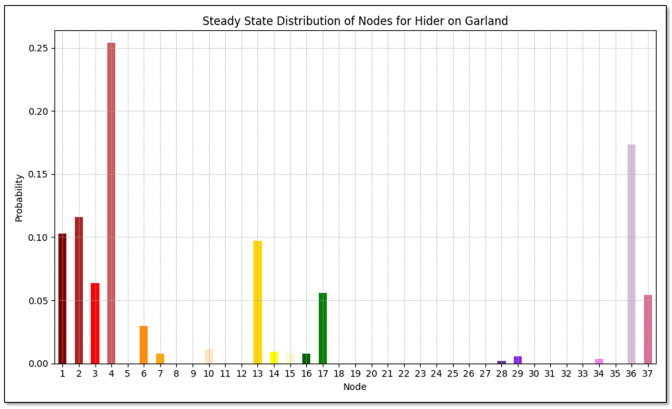

Fig. 9: Steady-state distribution of nodes for hider on Garland.

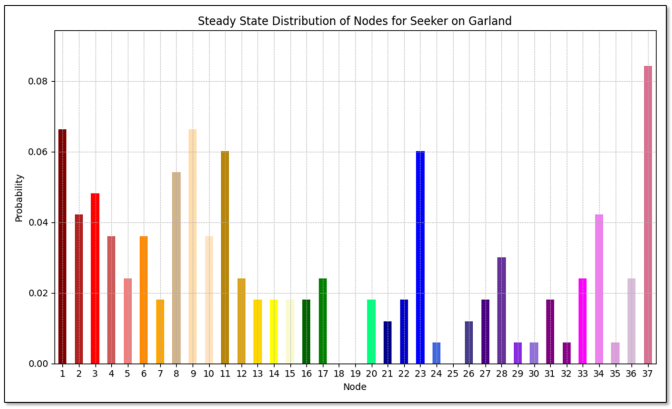

Fig. 10: Steady-state distribution of nodes for seeker on Garland.

circle the seeker. The hider tends to hide in either the circular points or far away isolated nodes.

Figure 10 shows the more evenly spread seeker distribution, emphasizing nodes furthest from the start. Time spent at nodes scales with node size, from small offices to larger classrooms. As shown in each result, the seeker must emphasize an exploration phase in almost all its approaches. This results in a much broader distribution of visits.

C. Graph of a GPIO Circuit Diagram

The GPIO circuit graph resembles a tree structure with three branches, representing a GPIO circuit from an 8-bit AVR microcontroller [28]. This map's analysis can indicate potential hotspots for HTs as this is a realistic circuit. However, this circuit is still relatively simple compared to realistic circuits in which HTs might be hidden.

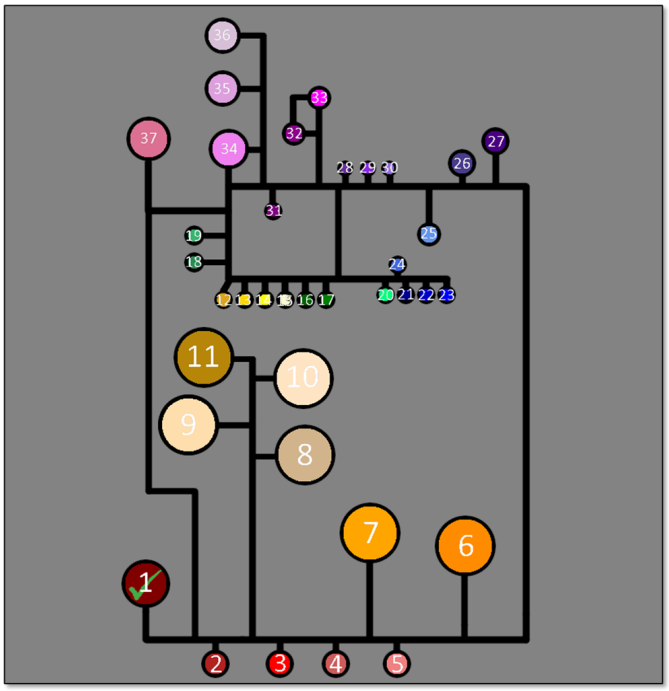

Fig. 11: Graph of a GPIO circuit from an 8-bit AVR microcontroller.

Figure 11 shows the GPIO map. Figure 12 reveals that hiders select a single branch and occupy the furthest node, with nodes 8, 15, and 22 showing the highest occupancy. Returning to the starting node is also a strategy we identified in the previous level that is common to the dynamic nature of our video game.

Figure 13 depicts the seeker distribution, where the tri-branch structure gives seekers a 33% initial chance of locating a hider. Backtracking consumes time, increasing the likelihood of finding the hider in subsequent branches.

V. Discussion

Our analysis provides insights into player strategies and behaviors, which are applicable in fields like cybersecurity

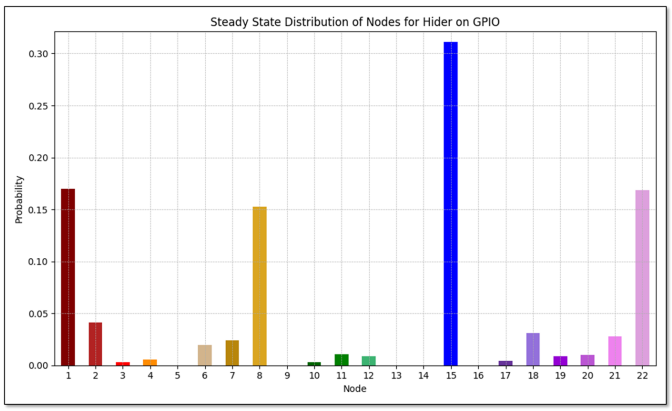

Fig. 12: Steady-state distribution of nodes for hider on GPIO.

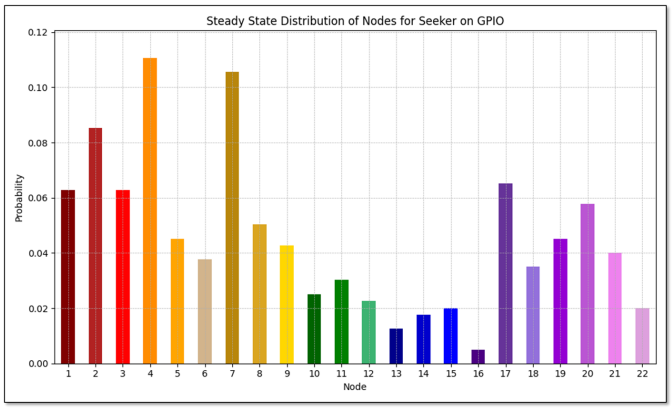

Fig. 13: Steady-state distribution of nodes for seeker on GPIO.

to understand potential threat actor locations. Our approach of using graph theory to map physical architectures, such as GPIO circuits, can be extended to other hardware circuits, noting that the balance between graph size and interesting gameplay becomes more challenging as the graph grows.

This work provides some ideas for understanding where HTs might be hidden in a circuit but does not fully connect the two problems. The disconnect between the video game and the real HT detection problem is that the game has a dynamic hider, while the HT is a static hider. However, we could still use this approach to get insights, as described below.

First, levels in this game could be snapshots of a larger circuit. This would allow larger circuits to be partitioned so that each level would give us insight into pieces of the overall circuit. Second, capturing moments of hiding and recording the aggregate of the seekers' detections when the hider is at a specific node might provide insight into the static HT problem.

VI. Conclusion

In this work, we created a video game that allows humans to play Hide&Seek. Our game system is designed to record positional data of the hider and seeker so that we can analyze their strategic behavior. We showed aggregate results of three levels of this game to illustrate how the data can be used to understand the hiders and seekers' behavior, which then can be used to understand the nature of HT being hidden. Our results show that this type of system can create and describe how the data collection techniques might provide insight into designing agent heuristics for real-world Hide&Seek problems like HT detection.

References

[1] M. Chapman, G. Tyson, P. McBurney, M. Luck, and S. Parsons, "Playing hide-and-seek: an abstract game for cyber security," in *Proceedings of the 1st International Workshop on Agents and CyberSecurity*, pp. 1–8, 2014.

[2] A. Sarihi, A. Patooghy, A.-H. A. Badawy, and P. Jamieson, "The seeker's dilemma: Realistic formulation and benchmarking for hardware trojan detection," *arXiv preprint arXiv:2402.17918*, 2024.

[3] A. Sarihi, A. Patooghy, P. Jamieson, and A.-H. A. Badawy, "Hiding in plain sight: Reframing hardware trojan benchmarking as a hide&seek modification," *arXiv preprint arXiv:2410.15550*, 2024.

[4] P. Jamieson, L. Grace, and J. Hall, "Research directions for pushing harnessing human computation to mainstream video games," in *Meaningful Play*, 2012.

[5] P. Jamieson, L. Grace, J. Hall, and A. Wibowo, "Metaheuristic entry points for harnessing human computation in mainstream games," in *Online Communities and Social Computing: 5th International conference, OCSC 2013, Held as Part of HCI International 2013, Las Vegas, NV, USA, July 21-26, 2013. Proceedings 5*, pp. 156–163, Springer, 2013.

[6] ESP Game, "http://www.gwap.com/gwap/gamesPreview/espgame/." 2008.

[7] S. Hacker and L. von Ahn, "Matchin: eliciting user preferences with an online game," in *International conference on Human factors in computing systems*, pp. 1207–1216, 2009.

[8] E. Law, A. Mityagin, and M. Chickering, "Intentions: A game for classifying search query intent," in *International conference extended abstracts on Human factors in computing systems*, pp. 3805–3810, 2009.

[9] L. von Ahn and L. Dabbish, "Designing games with a purpose," *Commun. ACM*, vol. 51, pp. 58–67, August 2008.

[10] L. von Ahn, S. Ginosar, M. Kedia, and M. Blum, "Improving image search with phetch," in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, pp. IV–1209 –IV–1212, 2007.

[11] L. von Ahn, R. Liu, and M. Blum, "Peekaboom: A Game for Locating Objects in Images," in *ACM Conference on Human Factors in Computing Systems*, pp. 55–64, 2006.

[12] FoldIt, "http://fold.it/portal/." 2008.

[13] G. Viglietta, "Gaming is a hard job, but someone has to do it!," in *FUN*, pp. 357–367, 2012.

[14] G. Aloupis, E. D. Demaine, A. Guo, and G. Viglietta, "Classic nintendo games are (np-)hard," *CoRR*, vol. abs/1203.1895, 2012.

[15] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Comput. Surv.*, vol. 35, pp. 268–308, September 2003.

[16] K. Siu, M. Guzdial, and M. O. Riedl, "Evaluating singleplayer and multiplayer in human computation games," in *Proceedings of the 12th International Conference on the Foundations of Digital Games*, pp. 1–10, 2017.

[17] A. Ehret, P. Jamieson, and L. Grace, "How to use combinatorial optimization problems (traveling salesman problem) for procedural landscape generation," in *GAMEON'2015*, 2015.

[18] L. Terry, V. Roitch, S. Tufail, K. Singh, O. Taraq, W. Luk, and P. Jamieson, "Harnessing Human Computation Cycles for the FPGA Placement Problem," in *The international conference on Engineering of Reconfigurable Systems and Algorithms*, 2009.

[19] D. E. Gundry, *Designing Games to Collect Human-Subject Data*. PhD thesis, University of York, 2022.

[20] S.-Y. Yu, R. Yasaei, Q. Zhou, T. Nguyen, and M. A. Al Faruque, "Hw2vec: A graph learning tool for automating hardware security," in *2021 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 13–23, IEEE, 2021.

[21] C. A. Kamhoua, H. Zhao, M. Rodriguez, and K. A. Kwiat, "A game-theoretic approach for testing for hardware trojans," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 2, no. 3, pp. 199–210, 2016.

[22] W. Saad, A. Sanjab, Y. Wang, C. A. Kamhoua, and K. A. Kwiat, "Hardware trojan detection game: A prospect-theoretic approach," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 9, pp. 7697–7710, 2017.

[23] D. Kahneman and A. Tversky, "Prospect theory: An analysis of decision under risk," in *Handbook of the fundamentals of financial decision making: Part I*, pp. 99–127, World Scientific, 2013.

[24] T. Das, A. R. Eldosouky, and S. Sengupta, "Think smart, play dumb: Analyzing deception in hardware trojan detection using game theory," in *2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, pp. 1–8, IEEE, 2020.

[25] S. Brahma, L. Njilla, and S. Nan, "Game theoretic hardware trojan testing under cost considerations," in *International Conference on Decision and Game Theory for Security*, pp. 251–270, Springer, 2021.

[26] S. Nan, L. Njilla, S. Brahma, and C. A. Kamhoua, "Game and prospect theoretic hardware trojan testing," in *2023 57th Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–6, IEEE, 2023.

[27] V. Gohil, M. Tressler, K. Sipple, S. Patnaik, and J. Rajendran, "Games, dollars, splits: A game-theoretic analysis of split manufacturing," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 5077–5092, 2021.

[28] "ATMEL 8-bit AVR Microcontroller." https://ww1.microchip.com/downloads/aemDocuments/documents/MCU08/ProductDocuments/DataSheets/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061B.pdf, 2011. Accessed: 07-07-2019.