An Online Gamified Learning Platform for Teaching Cybersecurity and More

Mac Malone
tydeu@cs.unc.edu
UNC Chapel Hill
North Carolina, USA

Yicheng Wang
yicheng@cs.unc.edu
UNC Chapel Hill
North Carolina, USA

Fabian Monrose
fabian@cs.unc.edu
UNC Chapel Hill
North Carolina, USA

Abstract

We present an online gamified learning platform for computer science and cybersecurity education. Exercises within the platform revolve around a custom game wherein students can demonstrate learned skills regarding password security, web security, traffic analysis, reverse engineering, cryptanalysis, and much more. We describe some key features that together make our platform novel, including its distributed infrastructure, game engine, integrated development environment, automated feedback system, and support for individualization. We demonstrate how these features assist in the learning process — both in theory and in practice — and report on the use of the platform in a cybersecurity course.

CCS Concepts

• **Security and privacy**; • **Applied computing → Interactive learning environments**; *Distance learning*;

Keywords

Learning Platform; Gamification; Cybersecurity; Distance Learning

ACM Reference Format:
Mac Malone, Yicheng Wang, and Fabian Monrose. 2021. An Online Gamified Learning Platform for Teaching Cybersecurity and More. In *Proceedings of the 22nd Annual Conference on Information Technology Education USB Stick (SIGITE '21), October 6–9, 2021, SnowBird, UT, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3450329.3476859

1 Introduction

Given our increasing reliance on cyberspace for nearly every aspect of our personal lives, building safe and secure digital ecosystems is critical. Unfortunately, even as cyberspace continues to transform the way we live, many vulnerabilities in critical infrastructure are left unchecked, exposing us to a myriad of threats. Today, cybersecurity attacks and breaches are an all too familiar events, underscoring the need for better defenses.

Sadly, the reality is that we still lack a cyber-savvy work force that can help defend against even known systemic risks. Training the next generation of cybersecurity experts requires teaching today's students *hands-on skills*. Indeed, Mason *et al.* [11] may have put it best: "educating a cybersecurity professional is similar to training a pilot, an athlete or a doctor. Time spent on the task for which the person is being prepared is *critical* for success." (Emphasis added.) However, the success of such education depends *heavily on how interesting the exercises are* [4]. Those who are not sufficiently motivated to learn new concepts or apply them on their own will gain less. Finally, recent world events have created yet another challenge for educational platforms beyond engagement – they need to be easily transitioned to a remote-learning environment.

Thus, in developing our cybersecurity curriculum, we had three goals in mind: (1) provide students with a hands-on cybersecurity education that ensures understanding of the tools and techniques needed to tackle everyday problems, (2) make the exercises interesting, thereby promoting self-study, peer instruction, and overall engagement, and (3) support distance learning.

To provide students with hands-on learning, one can leverage *student-centered* and *challenge-based learning* (CBL) techniques [14]. In student-centered approaches, learners are challenged to draw on prior learning, acquire new knowledge, work as a team, and use their creativity to arrive at solutions as part of an active learning exercise. In CBL, learners participate in some form of competition and typically learn more quickly as they are often motivated by the common goal of potentially winning a prize.

To make exercises interesting, one can transform them into games. Gamification increases motivation by providing students with clear, achievable goals and by making learning more fun, especially when competition is encouraged [12]. However, some criticize gamification as overly focussed on the provision of extrinsic rewards, possibly damaging intrinsic motivation for learning [9]. Nevertheless, the findings of Hamari *et al.* [8] and Bai *et al.* [1] are encouraging and indicate that gamification can be a winning strategy for both learning outcomes and engagement.

This is especially true in cybersecurity, where gamification often takes the form of "capture-the-flag" competitions (CTFs) or wargames [3, 16]. In a CTF, players are tasked with finding a string, called a *flag*, hidden within some system and must leverage cybersecurity techniques to discover it. In a wargame, players compete with one another to compromise some vulnerable system and whoever completes the objective first wins. While such activities can be engaging, they lack many features of a real game and are largely just cybersecurity exercises with added competitive aspects. This means they also lack many of the facets that make games so apt for cybersecurity education [17].

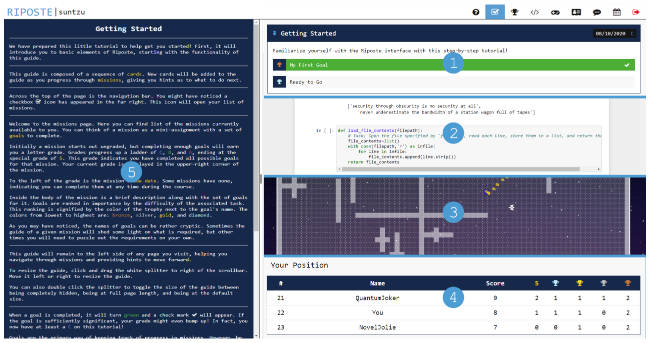
Seeking to improve upon existing methods for our goals, we created our own platform – Riposte. Riposte combines a number of industry and educational best practices into a novel package, and offers several contributions, including: *(i)* a distributed infrastructure that supports a large scale and rapid updates, *(ii)* a core game that students can interact with via both an UI and an API, *(iii)* a built-in integrated development environment (Jupyter Notebook) that allows students to begin developing right away with no machine setup or configuration, *(iv)* a thorough guide that provide explanations and hints about the task at hand *(v)* an advanced auto-grader that provided automated feedback about a students' performance and even give personalized hints for common mistakes. *(vi)* individualized assignments wrapped up in a gamified UI (*e.g.,* with trophies, a leaderboard, etc.) that allow for targeted student-centered educational approaches (*e.g.,* promoting student growth) while help to prevent academic dishonesty. We discuss each of these in the following sections.

2 Design Overview

Riposte is a learning platform designed for gamified cybersecurity education. In what follows, we discuss the design and merits of the platform from both technical and educational perspectives.

Technical Design

We leverage a modular design that allows for the easy selection and presentation of disjoint services (henceforth referred to as "modules"'). The modules include a ① **Missions** module that allows students to track their progress throughout the assignment; a ② **Code** module that provides students with a Jupyter Notebooks to develop Python online to complete the assignment a ③ **Game** module with a 2D shoot 'em up style game that contains a variety of challenges that teach various cybersecurity concepts; a ④ **Leaderboard** module that ranks students according to their achievements within the platform; and a ⑤ **Guide** module that assists students in navigating the interface and progressing through the assignment.



Getting Started

We have prepared this little tutorial to help get you started! First, it will introduce you to basic elements of Riposte, starting with the functionality of this guide.

This guide is composed of a sequence of cards. New cards will be added to the guide as you progress through missions, giving you hints as to what to do next.

Across the top of the page is the navigation bar. You might have noticed a checkbox ☑ icon has appeared in the far right. This icon will open your list of missions.

Welcome to the missions page. Here you can find list of the missions currently available to you. You can think of a mission as a mini-assignment with a set of goals to complete.

Initially a mission starts out ungraded, but completing enough goals will earn you a letter grade. Grades progress up a ladder of C, B, and A, ending at the special grade of S. This grade indicates you have completed all possible goals for that mission. Your current grade is [illegible]layed in the upper-right corner of the mission.

To the left of the grade is the mission [illegible]de date. Some missions have none, indicating you can complete them at any time during the course.

Inside the body of the mission is a brief description along with the set of goals for it. Goals are ranked in importance by the difficulty of the associated task. This ranking is signified by the color of the trophy next to the goal's name. The colors from lowest to highest are: bronze, silver, gold, and diamond.

As you may have noticed, the names of goals can be rather cryptic. Sometimes the guide of a given mission will shed some light on what is required, but other times you will need to puzzle out the requirements on your own.

This guide will remain to the left side of any page you visit, helping you navigate through missions and providing hints to move forward.

To resize the guide, click and drag the white splitter to right of the scrollbar. Move it left or right to resize the guide.

You can also double click the splitter to toggle the size of the guide between being completely hidden, being at full page length, and being at the default size.

When a goal is completed, it will turn green and a check mark ✓ will appear. If the goal is sufficiently significant, your grade might even bump up! In fact, you now have at least a C on this tutorial!

Goals are the primary way of keeping track of progress in missions. However, be

Getting Started 08/10/2020 C

Familiarize yourself with the Riposte interface with this step-by-step tutorial!

🏆 My First Goal ✓

🏆 Ready to Go

```
['security through obscurity is no security at all',
  'never underestimate the bandwidth of a station wagon full of tapes']

In [ ]: def load_file_contents(filepath):
            # Task: Open the file specified by `f[illegible], read each line, store them in a list, and return th
            file_contents=list()
            with open(filepath,'r') as infile:
                for line in infile:
                    file_contents.append(line.strip())
            return file_contents
```

Your Position

#	Name	Score	S	🏆	🏆	🏆	🏆
21	QuantumJoker	9	2	1	1	1	2
22	You	8	1	1	1	0	2
23	NovelJolie	7	0	0	1	0	2

Figure 1: A collage of the modules of Riposte: (1) missions, (2) code, (3) game, (4) leaderboard, (5) guide.

Students access Riposte through a single login portal that supports both custom login credentials and federated authentication (*e.g.,* with Google Account Services). Once logged in, the web client connects with the endpoint(s) assigned to the student. Each student is assigned their own VM which contains their instances of relevant modules (*i.e.,* the game, Jupyter Notebook, etc.). These modules and Riposte as a whole are deployed through Docker containers, enabling fast setup and easy scalability. To add a new student machine, we provision a new VM, register the endpoint with the main Riposte server (through its RESTful API), and boot up the Docker containers. Riposte also supports a continuous integration / continuous deployment (CI/CD) pipeline that allows us to quickly update and adapt the platform as a semester progresses.

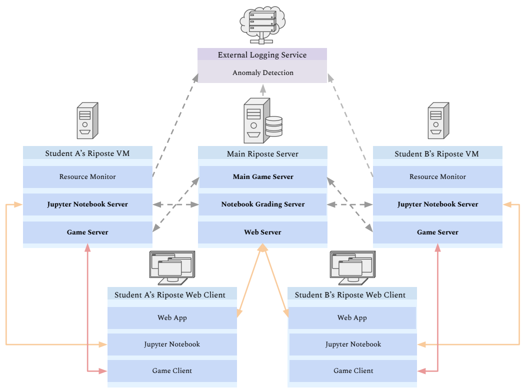


Figure 2: *Overview of the architecture.* **The bolded services are run within isolated Docker containers. Arrows represent communications between services. Red solid ones are those that the students can monitor and are encouraged to explore and exploit, Yellow solid ones can technically be seen but are not the focus, and gray dashed ones are invisible to students.**

The containerization also isolates the students' environments from one another. This is especially important in the context of cybersecurity education, as having hands-on experience with offensive security in a risk-free environment is paramount. However, there still remains the potential for students to break their own machines. To help mitigate this, we set up services that run both inside and outside of the docker container to immediately restart the modules should they crash. In addition, Riposte keeps detailed logs of its activity and forwards them to a logging service to allow diagnosis of complicated failures. These logs also serve as another way to keep tabs on students, being used both to track their performance and to detect suspicious activities (*e.g.,* potential malicious cyber attacks and/or academic dishonesty).

Educational Design

In his seminal work on principles of instructional design [6, 7], Gagne *et al.* outlined nine steps of the learning process that all learning system should incorporate [5]. They are 1) Gain the attention of the learner. 2) Inform the learner of the objectives. 3) Stimulate the recall of prerequisite learning. 4) Present the educational material. 5) Provide learning guidance. 6) Elicit performance. 7) Provide feedback about the performance. 8) Assess performance. 9) Enhance retention and transfer.

Our goal is to support each step and facilitate learning. Step 1 is about motivating students to engage with the learning process. This is where gamification comes in. Step 2 is the the domain of the Missions module: assignments and learning objectives are disseminated in a gamified form to students, leveraging the gaming notion of trophies (achievements) to denote concrete tasks. Step 3 is accomplished through "Field Exercises", which are special pre-assignment exercises designed to test students on their prerequisite knowledge. These low stakes pretests verify if learners have the necessary background knowledge for upcoming assignments, and, in the case of the first exercise, whether or not the course is right for them. To motivate students to complete the exercise but not panic over it, doing sufficiently well on the field exercise rewards students with a minimum passing grade on the upcoming assignment, but failing the exercise has no negative grade consequences.

As we utilize Riposte within an existing academic course, Steps 4-5 are mainly provided through the course itself (*e.g.,* via lectures, office hours, etc.) rather than Riposte. However, we still augment this with the Guide module – the guide progresses through step-by-step instructions as students complete tasks and can even provide hints if the system detects that a learner is getting lost. We created a fully integrated tutorial on the functionalities of the Riposte platform within the guide that can can be completed by the students on their own. Thus, Riposte can support being used as a completely standalone learning platform without an academic course backing it. Steps 6-8 are handled through a combination of the Game, Code, and Missions modules that we will delve into deeper in §3 - §5.

A common challenge that arises with Step 8 of the learning process is striking a balance between assessing a student's growth versus their proficiency at a set of tasks. Ideally, education grows a student to a requisite knowledge of proficiency in a field, but this is hardly universal. Assignments are generally graded based on whether or not students completed certain learning objectives. That is, it measures whether they are *proficient* in a task). However, different students may need very different amounts of growth to achieve such a level of proficiency. The same task can require only a small improvement from one student, but a vast jump for another. If the latter does improve (and even potentially improve more, in relative terms, than the other student), but not enough to fully reach proficiency, they will still be ranked low and graded harshly under a pure proficiency model. This can discourage them and turn them away, despite their promising learning potential, because the fruits of their labor were not rewarded. Individualizing assignments can help with this (and more) and is the focus of §6.

Finally, Step 9 concerns retaining what is learned and transferring it to real world situations. We do this by designing our assignments in a cumulative manner, making sure relevant concepts are continually repeated and built upon in order to assist retention. We also try to make real world connections in each of our assignments and developed the Riposte learning platform to closely mirror aspects of real world tools and techniques. Some examples of these assignments will be presented in §7.

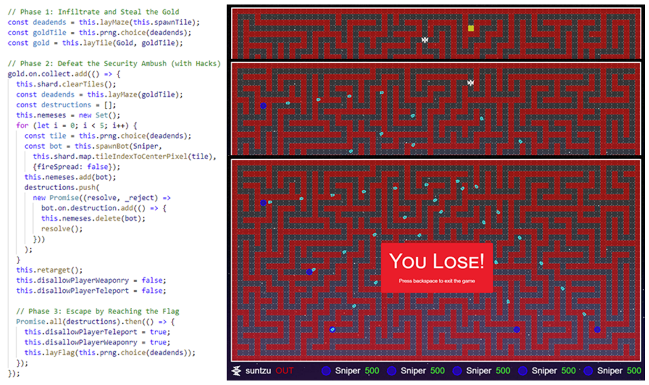
3 Game Engine

Riposte, as a gamified learning environment, contains a 2D shoot 'em up game at its core. The core gameplay loop involves controlling an avatar that can move around a 2D game board and fire projectiles to complete specific objectives. Hacking the game allows student to augment their avatar with more advance capabilities such as teleporting, moving through walls, and laying mines. The game supports both player versus player (PvP) and player versus environment (PvE) game modes, and also supports team play in both modes. In PvP, the goal is reduce the opposing players' (or the opposing teams') life total to zero. In PvE, players face off against a variety of AI challenges with differing goals: solving puzzles, navigating mazes, defeating bosses, surviving waves of enemies, etc. The challenges themselves can be locked behind various authentication mechanisms such as vulnerable cryptographic ciphers that players may need to exploit.

Thus, despite its presentation as an ordinary game, it can be used as a vector to teach a wide range of computer programming and computer security concepts. For example, learners log into into the game using a separate set of credentials from the main Riposte portal, which forms the basis for an online password cracking exercise where students have to crack the instructor's passwords and defeat them at PvP. We also taught various cryptanalysis techniques (such as CBC byte flipping) by having students forge challenge unlock codes. In addition, the communication protocol of game is similar to that of many common web apps (JSON encoded messages over a WebSocket connection to perform remote procedure calls). Therefore, by teaching students how to hack the game, they get hands-on experience with real-world technologies while learning the basics of threat analysis and reverse engineering.

The game is modular in that challenges and game modes are heavily encapsulated, which makes customization straight forward. Anticipating the flexibility needed for making changes to the platform to adjust our lessons each semester, we took special care to make it easy to design new challenges and enemy AIs. In developing the challenges, we also looked for ways to measure varying levels of skill — *i.e.,* different challenges need to enable and disable different kinds of exploits to accurately assess the specific learning objective. For example, a common exploit used in many challenges allows a player to teleport freely around the board. However, we disable this exploit in assignments where the learning objective demands that the students develop step-by-step maze-solving algorithms, keeping the completion of the assignment in-line with achieving the learning objective. Riposte supports this flexibility via per-challenge configurations. An example of how the API is used can be seen in Figure 3.

The distributed design of Riposte also factors into game. In our setup, each student connects to their own instance of the game server hosted in a restricted section of their VM (known as the *satellite* server). This allows us to design assignments that require interaction with the game server binary and prevents students from interfering with others' games. At the same time, we want to support teamwork and a communal leaderboard, so a main server exists which manages the shared state of the satellite servers. Results from games played on each satellite are communicated securely to the main server, and each satellite can query the main server for information about other satellites, allowing for team play. In a team play session, one student hosts the game on their server and every other player joins that game through an in-game lobby interface. The other players' clients know who to connect to by query the main server and the host's server can verify those players' logins by querying the main server as well.



```
// Phase 1: Infiltrate and Steal the Gold
const deadends = this.layMaze(this.spawnTile);
const goldTile = this.prng.choice(deadends);
const gold = this.layTile(Gold, goldTile);

// Phase 2: Defeat the Security Ambush (with Hacks)
gold.on.collect.add(() => {
  this.shard.clearTiles();
  const deadends = this.layMaze(goldTile);
  const destructions = [];
  this.nemeses = new Set();
  for (let i = 0; i < 5; i++) {
    const tile = this.prng.choice(deadends);
    const bot = this.spawnBot(Sniper,
      this.shard.map.tileIndexToCenterPixel(tile),
      {fireSpread: false});
    this.nemeses.add(bot);
    destructions.push(
      new Promise((resolve, _reject) =>
        bot.on.destruction.add(() => {
        this.nemeses.delete(bot);
        resolve();
      }))
    );
  }
  this.retarget();
  this.disallowPlayerWeaponry = false;
  this.disallowPlayerTeleport = false;

  // Phase 3: Escape by Reaching the Flag
  Promise.all(destructions).then(() => {
    this.disallowPlayerTeleport = true;
    this.disallowPlayerWeaponry = true;
    this.layFlag(this.prng.choice(deadends));
  });
});
```

Figure 3: Code for an example challenge, Heist, that demonstrates the challenge creation API. Heist has three phases: (1) solve a maze to find the gold, (2) defeat the adversaries that appear when collecting it, and (3) solve another maze to escape. Players don't know about these phases initially and are tasked with completing the challenge in the fewest attempts possible.

4 Web-based Development Environment

Hands-on learning in computer science demands coding. While students could code on their own machines, doing so has disadvantages. For one, it assumes that learners already know how to properly setup and manage their development environments, which is not always the case. This is especially true in computer security, where topics like binary reverse engineering and traffic analysis can require students to setup and run VMs and complex software. Much of the common tooling is written in low-level languages and is frequently complex enough that even experienced computer scientists outside the domain have a hard time managing it. As such, although setting up and managing a development environment is an arguably essential skill for cybersecurity practitioners to learn, it is often beyond the scope of an introductory course.

To minimize frustration and lower the barriers to entry, we integrated the necessary development environment tools into Riposte in a manner that required little to no setup on the part of the student. A key advantage of this integration is that it allows us to track student interactions with the tools and use this information to advance the learning process. It also allowed us to more easily combine the development environment with other elements of Riposte (*e.g.,* Missions, Leaderboard, Game, and Guide modules).

Specifically, we used the Chrome Developer Tools to teach students how to perform traffic analysis, reverse engineer websites, and alter the game client. While the use of browser developer tools might not be appropriate for an advanced security course — where hands-on experience with lower level tools is part of the learning objective — it worked well for our introductory course, where the concepts being taught, rather than the tools, are the focus. Additionally, we provided each student with their own Jupyter notebook *hosted on the Riposte infrastructure*. Each student's Jupyter notebook server is simply a container running on their assigned VM and linked up to the main Riposte site. Thus a learner only needs to log into the site to access their notebook.

Jupyter itself is a web client for writing Python code. Code in Jupyter is organized into *notebooks* which are made up of many *cells*. A cell can either be a snippet of Python code or some prose written in a markup language, allowing users to freely interweave formatted text and code. Python cells can be run individually and print their output below them, allowing for quick feedback on how a piece of code functions. A cell can also support other output formats (*e.g.,* graphs). These features make Jupyter notebooks a very popular way of sharing code online, especially in the data science communities. Thus, it is a great IDE to integrate into Riposte, as it is both easy-to-use and practical. Jupyter has also been tested in academia before, with Cardoso and Leitão [2] showing that it can be an excellent tool to support the learning process.

5 Automated Feedback

The design of our platform allows us to track student progress and provide feedback to students in a number of ways via different modules. In the Game module, players are presented with statistics about their play, including metrics such as damage taken, bullets fired, and games lost. The server also privately tracks other information about the player's activity, such as login attempts, challenge unlocks (and unlock failures). That information is relayed to the main Riposte server, which uses it to update the learners progress within missions and award trophies. Instructors can use this information to track student progress.

The Code module (*i.e.,* the Jupyter notebook environment) contains a notebook template for each assignment. The template provides students with an automated checker which verifies that aspects of their solutions work correctly and monitors their activity. This allows instructors to see when students are working, for how long, and whether they are having any success. If we discover that they are spinning their wheels, we can intervene to assist. Riposte itself can also provide hints to help students recover through either the Guide or the checkers. Once a student has completed all the requisite tasks in a notebook, they can then submit it to a grading service (through the notebook UI). The grader checks that their solutions satisfy the constraints imposed and reports their performance. It can also provide hints on incorrect answers. Since we allow students to submit their work multiple times on most assignments, they can leverage this feedback to refine their answers. As with the Game module, all of the information is aggregated by the main Riposte server and saved in the database for analysis.

While the results from the Game and Code modules are used to automatically grade students, the grades produced are not final. As our assignments are semi-structured, we manually review every assignment and award partial credit for exploration and failed, but noteworthy, attempts. Also, extra credit is given for novel solutions. Initially, we awarded grades only after this manual review, but found that many students were stressed by not having a concrete rubric that provided expectations for their grade. We designed the Mission module trophies and the grader to remedy this problem. They help give students a sense of where they are on the grading scale and indicate what they need to do to improve. Thus, our grader reports an expected minimum grade.

The Leaderboard module ranks students based on their performance in missions (primarily by the number of trophies they have received). We report these rankings to students in two ways: a class-wide listing of the *n* top-ranked students and a private per-student listing of their relative position. Students are listed by a handle, as opposed to by real name, to allow students to remain somewhat anonymous should they so desire. We present a student's relative position by listing their current position, along with who is directly above and below them. We do this to mitigate the negative upward social comparisons that can be engendered by seeing oneself placed low on a large leaderboard, while still providing competitive learners with motivation (and bragging rights) for reaching the top. The top *n* ranking gives learners an idea of what level of performance is needed to climb high up the leaderboard, while the relative ranking informs them what they need to do to reach the next step up. This allows students to easily set both short-term and long-term goals.

Assignments also have stretch goals, which go beyond what is necessary to earn an "A" grade. To implement these, each assignment is designed with multiple objectives that measure varying levels of skill [1] and have many different solutions. The semi-structured nature of the game along with the opportunities presented by being able to hack it in various ways facilitates this greatly. As such, most assignments come with some form of stretch goal that requires students to evolve their approach beyond what is taught in class to complete (*e.g.,* by synthesizing their approach with information from other domains or by inventing something new).

6 Individualized Learning

One of the benefits of computerized learning platforms is that they provide the ability to automatically individualize the learning experience for each student. In our platform, individualization comes in a number of forms. First, we have the aforementioned individualized feedback within the game and IDE. Students can be given hints, feedback, and even objectives on an individual basis according to their performance and pedagogical needs. Second, exercises can have their content randomized to prevent academic dishonesty. For example, in one of our assignments students are required to crack an encrypted trace and reverse engineer a path through an in-game maze. We leveraged Riposte to provide each student with an different random trace and adjust the in-game maze accordingly to match this trace when the student plays. The level of individualization can also go beyond simple randomization: in our password cracking assignment, we now provide each student with a randomly selected password (and associated hint) from the same class of passwords.

We also use individualization to promote growth-based learning. By measuring initial student proficiency through pretests like our "Field Exercises", we can get a handle on where a student is initially. By tracking how far they have advance in future assignments, we can provide targeted feedback highlighting their success from within Riposte. Just as we reward students who complete proficiency stretch goals with extra trophies that place them higher up on the leaderboard, we do the same for growth stretch goals. This allows growers to see that their efforts are valued and motivate them to continue. Growth-based assessment was factored into a student's final grade for the course.

7 Case Study

Two iterations of Riposte have been employed for teaching the undergraduate "Introduction to Computer Security" class at our university. A simpler version, including only the Game and Leaderboard modules, was used in the Fall of 2019. We then developed the full version for the Fall of 2020. The 2019 course was taught completely in-person, but, due to world events, the 2020 version was taught in a distance learning setting. As result, we have been able to evaluate how Riposte performs in each setting.

In both interventions, the class consisted of exercises/missions related to online and offline password cracking, web client modification, network traffic inspection, SQL injection, and cryptanalysis. After each exercise, students were asked to complete a Likert-scale questionnaire regarding how gamification affected their interest in the assignment and perceived learning outcomes. Comparatively, we measured hours spent on the platform and the students' performance as proxy metrics for actual engagement and learning outcomes. Students were also asked to provide qualitative feedback about their experience.

In the 2019 iteration, we found consistent, statistically significant positive correlation between student's interest in gamification and their perceived learning outcomes, both in general and specific to the topic of each assignment (see, statistical analysis of learning outcomes in our previous paper [10]). We also found that many students continued to play the game (up to more than 100 rounds over 4-5 hours) despite having met the basic requirement for the exercises. They did so even if it did not significantly improve the quality of their solution (and even hurt it) [10]. We also received a lot of positive qualitative feedback from students at the end of the semester, regarding how interesting and engaging the assignments were – with some even raving that this was by far the most fun class they had taken within the CS department. These indicated to us that gamification was very effective in keeping students engaged.

That said, there was room for improvement in the 2019 iteration. For instance, many students expressed that they struggled with parts of exercises that were irrelevant to the security concepts we were trying to assess: 11/49 students wrote that they struggled with writing asynchronous code in JavaScript and therefore had trouble with some of the cryptanalysis assignments, and 6/49 students wrote that they had issues with one of the cryptanalysis assignments because they were unaware that their editor was surreptitiously inserting newlines (thereby leading to incorrect hashes). Additionally, students often times found the lack of structure around grading intimidating and frustrating. To address those limitations, we (a) integrated more opportunities for learning into the Riposte Web UI, added in "Field Exercises" to assess prerequisite knowledge, and displayed the student's current grade in real time, (b) integrated the Jupyter Notebook IDE into Riposte to standardize student's development environment, and (c) added automated feedback to diagnose common, operational mistakes.

These changes correspond to the Jupyter Notebook and Web Server components of the Riposte platform (Fig. 2). The impact of

[1] For more details on how we rank skill, see our previous paper [10].

these changes was evident in the 2020 iteration of the class, where students did not report any major operational issue being a roadblock to completing an assignment. More important, we saw similar qualitative feedback regarding engagement as that in 2019 (*e.g.,"I think the game format really helped me understand the concepts better, and it was a lot more engaging than typical written homeworks; This class is the most fun CS class I had at UNC, and I have learned so much by working through all the challenges; The module-based, gamified approach to learning was really great. I really enjoyed this class, and it was very different when compared to other CS courses"*), and no negative ones concerning frustration and uncertainty. Furthermore, the 2020 iteration demonstrated Riposte's capabilities as a distance-learning platform, aptly filling a void that existed in the current educational landscape.

8 Other Related Work

In gamified settings, the STEAMiE Education Engine [13] provides instructors with the tools to develop 3D educational games for STEM courses. Its primary focus is on teaching engineering and physics through designing and interacting with simulations. On the other hand, the Simple Academic Game Engine (SAGE) [15] is designed to teach computer science to students by having them design and implement games within SAGE. As such, like Riposte, the developed game itself is not educational per-se, but serves as a vector for teaching the student programming concepts. Riposte, unlike SAGE, is not a game engine that allows students to develop their own games, but rather a game that allows students to exploit and automate it. This is a better fit for our cybersecurity use case as it avoids the need to teach unnecessary programming concepts, while still utilizing a game properly focused on engaging gameplay rather than education (as STEAMiE would create).

9 Conclusion and Future Work

We presented Riposte, an online gamified learning platform for computer science and cybersecurity education. We outlined key features of the platform and discussed how each helps facilitate learning and, in particular, assists in each of Gagne's nine steps of the learning process. Our overall goals were to create a platform for hands-on cybersecurity education, make exercises interesting by applying them to a game, and support distance learning. We evaluated how well our design worked in practice by utilizing it in the Fall 2020 semester and found that, with Riposte, we could teach a wide variety of cybersecurity concepts via hands-on exercises in a distance learning environment while motivating students to engage with the subject matter through the use of gamification.

Despite the success, there is still room for improvement. On the technical side, Riposte utilizes a single central server with a single central database for state tracking. This limits Riposte's scalability as having hundreds of student VMs would quickly overrun the central server. Thus, ideally, further decentralization (*e.g.,* via sharding) is needed to remove this single point of failure. On the educational front, more rigorous A/B testing is needed of the various aspects of the platform along with further integration of per-student individualization. Specifically, we intend to compare a gamified course using Riposte to a gamified course without it (*e.g.,* one that just uses CTF exercises), and compare both to a non-gamified course. On individualization, we need to better adapt the platform to the many different ways players engage in games (such as goal-seeking versus exploring). There is active research on designing and testing AIs to personalize the gaming experience to these different styles [18]. Riposte could leverage such techniques to help in individualizing the gamified learning experience to best appeal to the student's style and thereby better promote engagement. These limitations notwithstanding, we hope that the ideas described herein will motivate similar approaches elsewhere in education.

10 Acknowledgments

We thank Jan Werner and Kedrian James for their help testing and deploying various aspects of the Riposte platform.

References

[1] Shurui Bai, Khe Foon Hew, and Biyun Huang. 2020. Does gamification improve student learning outcome? Evidence from a meta-analysis and synthesis of qualitative data in educational contexts. *Educational Research Review* 30 (2020).

[2] Alberto Cardoso and César Leitão, Joaquimand Teixeira. 2019. Using the Jupyter Notebook as a Tool to Support the Teaching and Learning Processes in Engineering Courses. In *International Conference on Interactive Collaborative Learning*, Michael E. Auer and Thrasyvoulos Tsiatsos (Eds.), Vol. 2. 227–236.

[3] Martin Carlisle, Michael Chiaramonte, and David Caswell. 2015. Using CTFs for an Undergraduate Cyber Education. In *USENIX Summit on Gaming, Games, and Gamification in Security Education.*

[4] Melissa Dark. 2014. Advancing Cybersecurity Education. *IEEE Security & Privacy* 12, 6 (Nov 2014), 79–83.

[5] Fadi P. Deek, Ki-Wang Ho, and Haider Ramadhan. 2000. A critical analysis and evaluation of Web-based environments for program development. *The Internet and Higher Education* 3, 4 (2000), 223–269.

[6] Robert M Gagne and Leslie J Briggs. 1974. *Principles of instructional design.*

[7] Robert M Gagne, Walter W Wager, Katharine C Golas, and John M Keller. 2004. *Principles of instructional design* (5th ed.).

[8] Juho Hamari, Jonna Koivisto, and Harri Sarsa. 2014. Does Gamification Work? – A Literature Review of Empirical Studies on Gamification. In *Hawaii International Conference on System Sciences.* 3025–3034.

[9] Christoph E. Hollig, Andranik Tumasjan, and Isabell M. Welpe. 2020. Individualizing gamified systems: The role of trait competitiveness and leaderboard design. *Journal of Business Research* 106 (2020), 288–303.

[10] Mac Malone, Yicheng Wang, Kedrian James, Murray Anderegg, Jan Werner, and Fabian Monrose. 2021. To Gamify or Not? On Leaderboard Effects, Student Engagement and Learning Outcomes in a Cybersecurity Intervention. In *ACM Technical Symposium on Computer Science Education.* 1135–1141.

[11] Daniel Manson and Ronald Pike. 2014. The Case for Depth in Cybersecurity Education. *ACM Inroads* 5, 1 (Mar 2014), 47–52.

[12] Cristina Ioana Muntean. 2011. Raising engagement in e-learning through gamification. In *International Conference on Virtual Learning*, Vol. 1. 323–329.

[13] Scott Nykl, Chad Mourning, Mitchell Leitch, David Chelberg, Teresa Franklin, and Chang Liu. 2008. An overview of the STEAMiE Educational game Engine. In *Annual Frontiers in Education Conference.* F3B–21–F3B–25.

[14] Geraldine O'Neill and Tim Mcmahon. 2005. Student-centred learning: What does it mean for students and lecturers? *Emerging Issues in the Practice of University Learning and Teaching* 1 (01 2005).

[15] Ian Parberry, J Nunn, Joseph Scheinberg, Erik Carson, and Jason Cole. 2007. SAGE: a simple academic game engine. In *Proceedings of the 2nd Annual Microsoft Academic Days on Game Development in Computer Science Education.* 90–94.

[16] Giovanni Vigna, Kevin Borgolte, Jacopo Corbetta, Adam Doupé, Yanick Fratantonio, Luca Invernizzi, Dhilung Kirat, and Yan Shoshitaishvili. 2014. Ten Years of iCTF: The Good, The Bad, and The Ugly. In *USENIX Summit on Gaming, Games, and Gamification in Security Education.*

[17] Brad Wolfenden. 2019. Gamification as a winning cyber security strategy. *Computer Fraud & Security* 2019, 5 (2019), 9–12.

[18] Jichen Zhu and Santiago Ontañón. 2020. Player-Centered AI for Automatic Game Personalization: Open Problems. In *International Conference on the Foundations of Digital Games.* Article 6, 8 pages.