Disturbed playing: Another kind of educational security games

Sebastian Koch, Joerg Schneider, Jan Nordholz
Technische Universitaet Berlin
{seb,komm}@cs.tu-berlin.de, jnordholz@sec.t-labs.tu-berlin.de

Abstract

Games have a long tradition in teaching IT security: Ranging from international capture-the-flag competitions played by multiple teams to educational simulation games where individual students can get a feeling for the effects of security decisions. All these games have in common, that the game's main goal is keeping up the security. In this paper, we propose another kind of educational security games which feature a game goal unrelated to IT security. However, during the game session gradually more and more attacks on the underlying infrastructure disturb the game play. Such a scenario is very close to the reality of an IT security expert, where establishing security is just a necessary requirement to reach the company's goals. By preparing and analyzing the game sessions, the students learn how to develop a security policy for a simplified scenario. Additionally, the students learn to decide when to apply technical security measures, when to establish emergency plans, and which risks cannot be covered economically.

As an example for such a *disturbed playing* game, we present our distributed air traffic control scenario. The game play is disturbed by attacking the integrity and availability of the underlying network in a coordinated manner, i.e., all student teams experience the same failures at the same state of the game. Beside presenting the technical aspects of the setup, we are also discussing the didactic approach and the experiences made in the last years.

1 Introduction

Capture-the-flag (CTF) contests are well known games to train and test security knowledge. Although CTFs were first developed by practioners, a number of universities developed teaching concepts based on CTFs and derived games.

A disadvantage of CTF contests is the typical limited duration of each game which leads the participants to use ad-hoc methods and strategies. Although availability aspects play a big role in real-world scenarios, CTF games have problems to address this kind of threats and often suppress these aspects by the rule set. So the game participants don't learn to handle an intentional loss of availability in a CTF contest.

In this paper, we introduce with *disturbed playing* another game concept to be used in class. Disturbed playing is based on the idea of CTF contests in analyzing and practically dealing with security topics. Teamwork is also an aspect which should be applied in this game.

However, while in a CTF the students are directly awarded score points for security measures and lose points for undetected vulnerabilities, our approach is based on a game with a regular game goal unrelated to security. As the students play this game, the system used to play is attacked by the trainers based on a known threat model.

Our proposed game concept takes the advantages of the CTF contest, i.e. practical experience with security topics, team work, and short-term feedback, yet prevents the disadvantages of ad-hoc methods by introducing multiple learning cycles which consist of preparation, game-play and analyzation. This enables the participants to develop a well conceived security policy. Additionally, the inclusion of availability aspects broadens the scope in comparison to other security games.

After discussing some related work, we introduce the didactic concept and the necessary requirements needed to prepare a class using disturbed playing. Afterwards, we describe the details of our implementation of the proposed concept based on an air traffic control (ATC) scenario. We report on our experiences using the concept in class for the last years, especially the strategies the teams decided to apply. Based on our experiences, we conclude with some guidelines for implementing a game with disturbed playing.

2 Related Work

Game based security education is widely discussed in the literature. Many games utilize the capture-the-flag (CTF) concept. In a CTF game, each student or group of students is in charge of a vulnerable system. During the game the students have to secure their own system and at the same time they have to break into the other's system to locate and disclose a virtual flag.

How CTF like games can be used in the classroom is discussed in great detail by Vigna in [6]. For pure CTFs as well as three similar games, he shares implementation details and experiences. Additionally, Vigna argues that balancing the student teams is important for the motivation and the learning outcome.

During a summer school, Mink and Greifeneder performed a study to compare offensive and defensive oriented training. [1] Two groups of students learned the same content. One group did offensive and the other defensive exercises. By comparing the learning outcome, they concluded that the offensive team had made better results. Although our experience with students points also in this direction, in our disturbed playing approach the students have to take the defensive perspective, albeit with a twofold focus as both risk assessor and actual ATC operator.

Näf and Basin introduce in [2] the conflict based and the review based game. The conflict based game is a real-time CTF. In the review based game, the students first protect their system and hand it over to the attacking students. Each student has first the role of the defender and is then in the second phase the attacker of another student's system. In both phases, the students have plenty of time to implement even complex defense and offense mechanisms. Therefore, Näf and Basin conclude that the new review based approach can also be used to teach more theoretical concepts. By introducing multiple game sessions, we use this idea in our disturbed playing concept, too: While the students do not change their role, they have plenty of time in-between the game sessions to prepare their machines and strategies. Furthermore, by repeating the sessions the students can also adjust their strategies and can experiment more.

There are many reports on how to set up a security lab with sandboxes and configurable networks like the one by Romney and Stevenson. [3] For most game concepts such a lab is required. In our disturbed playing concept, the sandboxing of the machines does not mainly aim at protecting the environment against the students, as most of the machines are under the students' control anyway. The sandboxing is rather used to guarantee each team an identical game setup which is free from influences created by other teams (whether accidental or malicious) or the surrounding infrastructure.

There are also other approaches to use games as a teaching method. For example, in the CyberCIEGE system realistic scenarios are simulated with a number of users interacting with each other in a 3D world. [5] The students can now experiment by activating and configuring security measures and the simulated user will visualize the positive and negative effects of the students' action.

Testing a security policy, its implementation, or how well its measures are mastered by the involved persons by simulating various scenarios is also a well established technique. These simulations range from flight simulations for single pilots to large realistic disaster exercises involving many workers from different emergency services. Such exercises have been also held in the area of IT security, e.g., the Cyber Storm[1] exercises in the US or the LÜKEX[2] exercise in Germany. However, such exercises are designed to test established security policies already setup by experts. Our game concept aims at building up the expertise to develop such security policies. Therefore, the students will be in both roles—designer of the policy and user who is affected by the positive and negative effects of his policy.

3 Disturbed Playing

The concept of *disturbed playing* tries to enhance the realism of case study or game-based security education. Beside the technical detail work, the most challenging task for new IT security specialists is deciding which action to take first, which problems have to be solved with technical or organizational methods, and which problems cannot be solved economically. Furthermore, the desires and established procedures of the users make the problem even more complex.

According to Bruce Schneier, everyone is doing risk management all the time. [4] The question is

[1] `http://www.dhs.gov/files/training/gc_1204738275985.shtm`

[2] `https://www.denis.bund.de/luekex/`

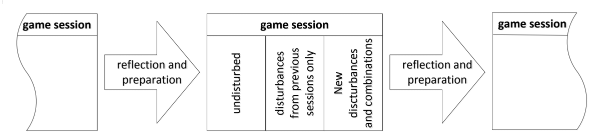

game session
reflection and preparation
game session
undisturbed
disturbances from previous sessions only
New disturbances and combinations
reflection and preparation
game session

Figure 1: Each session consists of three phases with gradually more disturbances. There are multiple sessions and the students can change their strategie in-between them.

whether the security decisions are based on objective data or reactions based on emotions, hunches, and anecdotes. The problem in risk management is the assessment of the extent of damage and of the probability of occurrence. Especially for very rare events associated with high damage an objective risk analysis is difficult. The idea of *disturbed playing* is to increase the probabilities associated with damages clearly specified as part of the rules. This gives the player an objective data base for the risk analysis and thus provides the ability to evaluate the impact of security measures.

At the same time, discussing strategies to develop security policies and the non-technical impact of security measures in class is not very attractive for computer science and engineering students. Still it is one of the most basic knowledge of an IT security expert.

We see disturbed playing as a form of problem based learning. The main didactic goal is to develop strategies to create and evaluate security policies. Additionally, students playing the game can also learn the use of security technologies and team management.

3.1 Basic setup: Game, Game sessions, and Disturbances

The general setup requires a game which is exciting and easy to learn as well as a number of undisturbed and disturbed game sessions. Students should be able to play the game alone or in a team with only few hours of training for the game itself. The training process can be enhanced by providing the undisturbed game and the game rules before the first session.

As the game is played by programmers, it is very important that the game play itself cannot be automated. Else, the students will spend a lot of effort in programming bots to play the game. Using the bot, the students will hardly experience the impact of the disturbances. Additionally, in our experience students will spend much more time in automating the game than in securing their system. Uncertain game rules and not well-formed, but still human readable communication help to make automations very unlikely.

A game in which students play against each other will not lead to comparable and repeatable results as discussed in [6]. Simulations or real-time games against computer players are much more suitable for this purpose and provide adjustable and predictable game situations that cause an intended stress level. The game should also feature a single score value, e.g., high score points combining all achievements. These score points will be the only asset values in the students' security police.

To allow the refinement of the security policy and to give feedback for different approaches, the game has to be played in multiple sessions. Between each session, the students can change their security policy, enhance their emergency plans, and even implement technical security measures. Each game session consist of three phases (see Figure 1): An undisturbed phase, a phase in which only known attacks at the same level of previous sessions are repeated[3], and a last phase containing also so far unknown attacks and not-yet experienced combinations. Throughout the whole session, the complexity of the game itself should be manageable, i.e., real stress is only introduced by the disturbances.

The first phase of undisturbed game play is used to settle down but also to verify that all technical and organizational changes do not impact the normal operation, i.e., the game play. In the next phase, the students can see how much their changes help to reduce the impact of the disturbances. The last phase addresses the main problem in policy design: Protecting against the unknown.

In order to avoid confusions of the students it is important to specify the threat situation exactly. The whole game environment should be designed to be friendly to the players, except for the attacker instance implementing the disturbances, i.e., any hidden vulnerabilities should be avoided. How the game will be disturbed should be disclosed at the beginning of the class. The system should be clearly (ideally formally) specified including the architecture, used protocols, and timings. Furthermore, the accesspoint where the attacker will apply the disturbances should be well defined.

Defining on broad categories, which part of the system is under attack and which can be assumed as reliable helps to perform the risk analysis and also focuses the students' effort on a small subset of

[3]except for session 1 in which no disturbances are known

the general problem. Disturbances or attacks can be realized for all typical security goals: An attacker extracts unprotected confidential information and discloses it to the computer player; the information exchanged between the students' systems and the computer player is modified; communication links or systems are shut down during the game.

All the potential disturbances have to be planned during the design phase of the game to include uncertainties in the protocol, to hide otherwise obvious attack vectors, and to provide hooks for possible solutions. Additionally, the game has to be specifically designed to prevent technical solutions to automate the game play.

3.2 Didactic dimensions

The main didactic goal as explained before is to develop strategies for creating security policies. By reducing the variables in the system the students can perform a risk analysis before each game session. The valuables are defined by the in-game properties leading to a higher final score. Each change of the environment leading to a loss in points is a vulnerability. As the game rules specify how the final score is calculated, the actual impact of the vulnerabilities can be computed. The threats are well defined beforehand, too. For the second phase, not only the attacks but also their likelihood are known from previous sessions and can be directly integrated in the policies as threat probabilities. Using this information, the students are even able to predict to some degree the potential risks during the last phase of the game session.

The investment costs of the security policy are covered by the workload the students will invest between the game sessions. The operational costs of the security policy, i.e., slower computations due to additional checks or organizational overhead, become visible during the game session. The effectiveness of the students' preparations is also assessed during the game session. The final score represents roughly how good the students' policy is, i.e., an ideal policy and perfect security measures should lead to score values close to an undisturbed game session.

The students are now not only in the role of a security expert who defines a policy but also the user who feels the negative and positive effects of the policy. This direct feedback on the technical and non-technical properties of a policy leads to a good learning experience.

Additional technical topics can be integrated, too. To realize the policy, the students have to implement security measures. By designing the vulnerabilities, the trainers are able to force the students to experiment with certain technologies. For example, by injecting forged packets into the network the students are led to solutions involving cryptographic signatures.

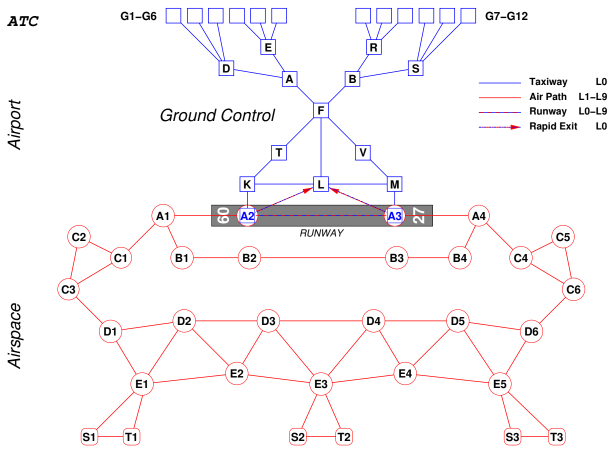


Figure 2: In the ATC game the students had to guide airplanes on this discrete map of an airspace. Rectangles represent ground nodes and circles airborne nodes. Sx and Tx are the airborne handover nodes.

We implemented our concept using an air traffic controller scenario. Another possible game scenario we have discussed is a distributed industrial control system like in a nuclear power plant or in an oil refinery. In both scenarios processes may be defined that cannot be stopped abruptly. If these processes get out of control a great damage is caused. The basic game is to maintain the operation of the plant or the refinery while external events disturb the underlying IT systems.

4 Example game: Air Traffic Controller

We tested our idea of *disturbed playing* as part of a practical security class. The class was developed in 2007 and the air traffic control game was played every year since then except for 2010 and 2011. There were five game sessions of two hours each spread over the whole semester.

In the air traffic control (ATC) game, each team of three students was given control over three virtual machines acting as ATC terminals. In the game, the students were responsible for a small airport and its surrounding airspace. The detailed rules of the game will be discussed in the following section.

We reduced the potential attack vectors to problems with the integrity and availability of the base infrastructure, i.e., power and network. Possible disturbances are modifications of network packets, injection of packets, dropping of packets, and loss of power for up to two virtual machines. The SSH control connection is guaranteed to remain safe and secure and the same holds for the hard drive and memory content.

After describing the detailed game rules in the next section, we will present the technical details of the game setup before discussing our experiences.

4.1 The ATC Game

After long discussions with our students of the CTF team, we developed the rule set for the air traffic control (ATC) game. Basically, a team of three students is responsible to guide incoming and outgoing airplanes of a small airport.

4.1.1 Rules

To reduce the complexity, the airspace and the ground area of the airport were reduced to a discrete map made up by a sparsely connected network of nodes (see Figure 2). A plane can only fly or taxi along the edges of the network. While a plane traverses an edge, the students can give an order what the plane should do when it reaches the next node, i.e., changing the direction or the flight level. There are 9 discrete flight levels and a plane can only move one level up or down per edge.[4] If a plane did not receive a command before its arrival at the next node it normally continues on the same flight level and on the edge most closely to a straight line.

There are six special handover nodes in the airspace, where outgoing planes had to be guided to and where incoming planes arrive in the system. Similarly, there are 12 gates on the ground where flights start and end. A separate flight plan specifies all flights. The plan is delivered for each flight as a single record to the virtual machine of the students. Each flight record specifies the callsign, the source and destination nodes as well as the estimated time of the flight in the source node (ETS). The snippet is delivered to the teams' VM a couple of minutes before ETS.

4.1.2 Scoring

The scoring is composed of positive points for successfully handling a flight, saving fuel and time as well as negative points for plane crashes, unsafe situations, time delays, and wasting fuel.

[4]In case of an emergency, descending by two flight levels is allowed.

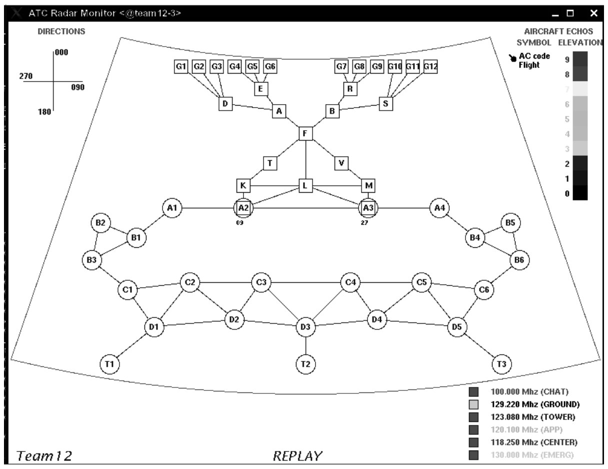


Figure 3: The ATC client application visualizes the radar information and features a chat-like radio simulation.

To balance the game, a plane crash is very unlikely. An automatic traffic alert and collision avoidance system like the real TCAS is implemented and planes with too little fuel complain about it very noisy. However, a crash is still possible and avoiding it should be considered as the highest asset by the students.

4.1.3 Playing

To play the game, each student has a virtual machine running a graphical ATC client application (see Figure 3). Additionally, the flight plan is delivered using a web service hosted on one of the students' systems. The ATC application features a radar image of the air space together with transponder data for each airplane. The actual game play was done using a chat-like radio simulation using the control application. The available commands on the simulated radio link are defined as a formal language. However, the responses of the airplanes are not restricted to a single possible phrase. The radio answers are always human readable, but are randomly chosen from a wide range of answer patterns. Using these hardly predictable answers, the students are very handicapped in implementing an automatic pilot system—and thus solving the game by software.

We use four virtual machines for each team (see Figure 4 for an overview). Three are handed to the students, who get full administrative control of the machines. These machines are equipped with a Debian Linux distribution and some ATC specific ap-

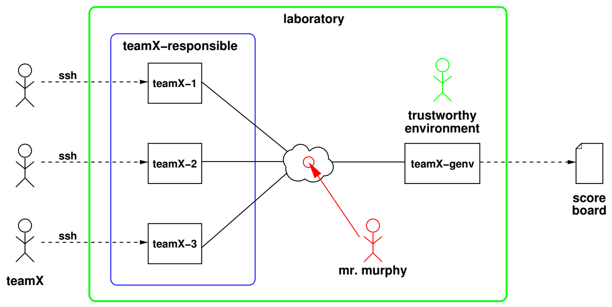


Figure 4: Three virtual machines are completely handed to the students, one additional simulates the game environment. Mr. Murphy is the incarnation of real world problems and the only source of disturbances

plications, e.g., the ATC client. The ATC specific applications are installed in binary format as well as in source code, so the teams are able to fix or improve the ATC specific applications.

After the hand over of the virtual team machines, they are not touched any more by the trainers, i.e., the students can rely on their integrity. The fourth machine handles the *game environment* (genv), i.e., simulates the state of the students' airspace, simulates the airlines handing in flightplan records, answers on radar echo requests, and mimics the radio answers of the airplanes. Additionally, the scoring is counted using the data collected there.

A virtual bridge connects all four virtual machines. The actual communication disturbances are realized in this bridge on the underlying virtualization host (see Mr. Murphy in Figure 4).

We used qemu/kvm as the virtualization platform to run 32 virtual machines with Linux on a host to serve eight teams. For each team, a separate VLAN was used to separate the teams. We used up to two such physical machines to play the game with a maximum of 14 teams.

The flight plan service is realized as a normal webservice which can receive signed messages and the students can migrate the webservice between all three systems. The radar is implemented using UDP: After sending a "start of sweep" packet, the requesting client receives a list of radar echoes together with the transponder data. The radio communication is done by multicast UDP—in our opinion, the most natural representaion of radio communication in an IP network. Each radio channel is a multicast group and the clients can join and leave the multicast groups by selecting the radio frequency. Because of the multicast nature of the radio protocol and the implementation design of the Linux IP stack, the control application with its radio chat interface can be started only once per virtual machine. Therefore, the students need to maintain all three virtual machines for efficient game play.

4.2 Disturbances

As described before, only attacks on the integrity and availability of the underlying infrastructure—network and power—are performed. We used the following threat model: the attacker (Mr. Murphy) has access to the virtual network between the systems of the students and the game environment which implements the aircrafts and flight plan simulation as well to the virtual power grid supplying the students' system.

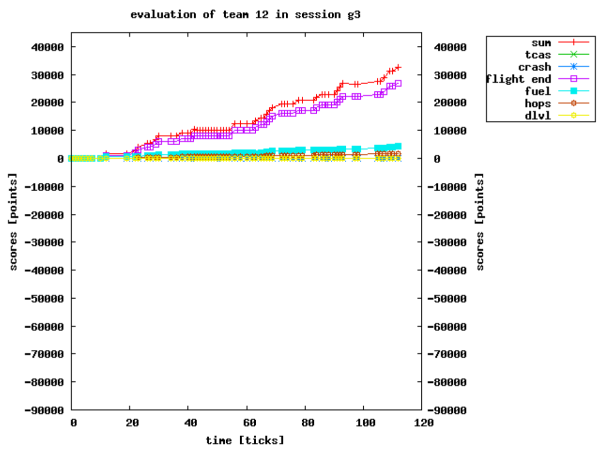
All disturbances are implemented in the virtualization host. All four virtual machines are not directly attacked. Even the game environment is friendly and not involved in the disturbances. During an attack, the game environment will always answer correctly on the requests of the students. However, this correct answer is then modified by Mr. Murphy in the virtual network before delivering it to the students' machines.

All scenarios are designed such that there is at least one solution to keep the game play alive. We tell the students that there will be no unsolvable catastrophical scenarios like shutting down the whole network or switching off all their systems. Furthermore, the SSH control connection to each machine is never affected.

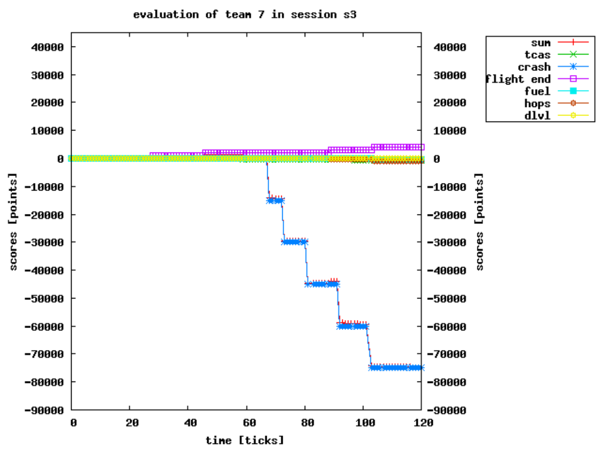
The disturbances are scripted before each session. During the session the problems arise in all virtual networks at the same time. Therefore, all student teams should be in a similar game state when a disturbance is activated.

To ensure that the disturbances are fair and reasonable the trainers test the complete session in a (final) rehearsal. The amount and strength of the disturbances are reduced until the test-session is passed without an aircraft crash.

The network disturbances are realized using iptables rules within the virtual network. Some attacks can be directly implemented with existing iptables commands, like shutting down the radar service by dropping all radar packets. More sophisticated attacks are implemented using userland programs called by iptables via nfqueue. For example, using these techniques we are able to jitter the position of the aircrafts in the radar echoes to simulate a failure in the transponder system. Another example is partially overwriting radio responses from aircrafts with static noise, thus making it harder to



(a) Session of a well prepared team.



(b) Same session of an unprepared team.

Figure 5: The impact of the disturbances heavily depends on the preparation and the selected policy of the students. One tick represents 30s and each phase is 20min or 40 ticks long.

understand position and heading reports from aircrafts over the radio. Furthermore, Mr. Murphy is able to inject packets to submit forged flight plans or to distract the radio operators by random chitchat.

The unreliable availability of power is simulated by forcing a shutdown of the virtual machine. We use the unrealistic regular shutdown procedure instead of a real unexpected power loss to avoid random total losses of virtual machines caused by repeated file system checks due to inconsistencies of the virtual hard disk filesystem. The machines are turned off for a defined time and start normally again after the simulated outage. We shut down a maximum of two machines at the same time.

4.3 Experiences and Evaluation

The teams have developed different strategies to manage the risks more or less successfully. These are some of the team strategies we could observe in the sessions. The teams

1. tried to build a robot to solve the air traffic control, fully or semi-automated.

2. used ad-hoc methods during the session (usually caused by the fact that the team didn't spend preparation time)

3. fixed bugs in the software of their VMs (e.g. vulnerabilities in the flight plan protocol)

4. improved the software of their VMs (e.g. additional redundancy features, logging)

5. developed information base redundancy using non-computer technology (e.g. paper work) to log the game's situation

6. trained (organizational) procedures for emergency cases.

The performance of teams applying strategy 1 was typically below the score point average of all teams. In most cases the automation was so complex and full of bugs that these teams were doubly handicapped. The automata often made the wrong decision so the teams had to compensate its behaviour manually. Additionally valuable session time was spent by the teams trying to fix the automata. The most successful teams have applied a combination of strategies 3 to 6.

For some participants it was needed to explain how a risk analysis is performed. After the teams had made their first experiences with the disturbances, we inserted an extra meeting after the second or third session with a talk from the trainers about the basics of risk analysis and management.

The necessary condition for managing the ATC challenge was the avoidance of an aircraft crash which had the most negative impact on the scoring. See the Figures 5(a) and 5(b) for the comparison of a good and a bad performed session. For the teams which managed the session without crashes the decision for the winner was calculated based on the other scoring categories, e.g. secure flight control (avoiding TCAS alerts) and economic aspects (saving fuel and time). This procedure was widely accepted by the teams.

The Figure 6(a) shows the team's score points at the end of the undisturbed game phase in each session. Some teams were suprised by the growing amount of flights from session to session. This caused some stress situations in this phase, especially in session 2. But most teams reoriented there strategies in the following sessions and expected more flights.

The Figures 6(b) and 6(c) are showing the positive progress of the most teams from session 1 to 4, although the amount of flights and the amount of disturbances have been rised from session to session. You can also see that the majority of the teams was highly motivated, especially in the cases of getting penalties in the first 20 minutes. They were working hard to compensate these negative scores by a good service in the rest of the session.

As the scores show, most teams improved their strategies during the course not only in handling known problems but also to cope with not-yet known disturbances. This trend matches with our observations during class. Additionally, most participants stated to have learned a lot about policy design.

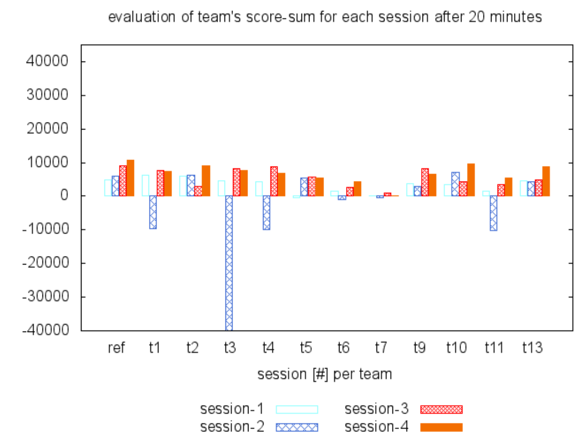
5 Lessons Learned

The *disturbed playing* concept decouples playing the game from the security tasks. In the following, we want to summarize the main aspects to be taken into account when implementing such a game setup.

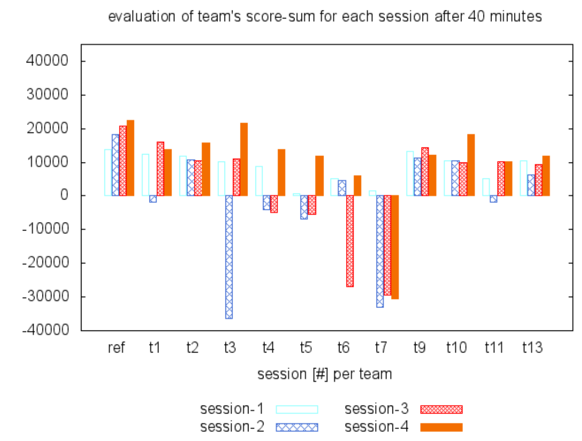
Plan the rules. Restrict the threat model well in advance The game design has to be developed with care taking into account the fun factor for the students, the non-automating, the room for attack and defense, and the implementation aspects. Additionally, the threat model has to be fixed rather early in the development cycle.

Students will try to automate the game play Although you implement a lot of uncertainties and explicitly ask the students to play by hand, there will be always at least one team trying to build an autopilot. As such automation destroys the real-time aspect and therefore the base for the security aspects of the game concept, it is important to add enough obstacles and game complexity.

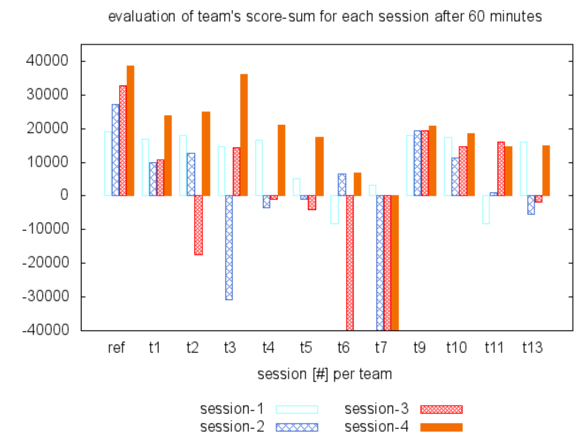
It is a lot of work. You need a good team. Preparing the game, the game environment, the disturbances and scripting the game sessions requires a lot of programming and testing efforts. Additionally, during the game sessions a support team is needed



(a) Sessions 1-4 at the end of the undisturbed game phase.



(b) Sessions 1-4 after the phase with the known disturbances.



(c) Sessions 1-4 at the end of the session after the unknown disturbances.

Figure 6: The majority of the teams t1-13 is making progress over the four sessions. The graphs show the sum of score points after each phase of the sessions 1-4. The first set of scores were realized by the trainer as reference.

to fix problems on the spot. Otherwise, the controlled progress cannot be guaranteed and there is a lot of frustration on the students' side. We worked in a team of at least three persons to develop, maintain, and run the game in addition to the trainer supervising the game sessions.

Fun factor is hard to calculate. Relation to security teaching not obvious. Some of our students complained that the game is boring and has too complex rules. Other students had problems to see the security aspects of the game. A good introduction and at least one intermediate discussion of the students' security policies is required to open the eyes.

The students' motivation is at least constant or even rising during the semester After understanding the game concept, most teams start to develop elaborated strategies, complex technical security measures, and new organizational methods. Some teams even start to perform emergency trainings before the game sessions.

It works! Students develop very interesting strategies during the game. Due to the repeated game sessions and by comparing their strategies with the other teams, the students learn what is important for a security policy. Additionally, even very programming oriented students understand the need for organizational security measures.

6 Conclusion and Outlook

We introduced our *disturbed playing* concept to teach IT security. The students play a game unrelated to security while the trainer disturb the underlying IT systems. The students can prepare a security policy as well as technical and organizational security measure before each game session. Unlike other concepts, we are able to address security aspects regarding system availability, too. Using this game concept, the students learn to develop and evaluate security policies. Additionally, the trainer can guide the students to try out multiple defense techniques.

We implemented our game concept in a practical class using an air traffic controller scenario. We discussed our rule set and the technical setup before sharing our experience of multiple runs of the game.

In future, we plan to further refine our game concept and to add more scenarios. Furthermore, we want to gain more real-time information from the game play and from other sources to further study our approach. We plan a formal study on the benefits of disturbed playing for the learning outcome.

References

[1] Mink, M., and Greifeneder, R. Evaluation of the offensive approach in information security education. In *Security and Privacy - Silver Linings in the Cloud*, K. Rannenberg, V. Varadharajan, and C. Weber, Eds., vol. 330 of *IFIP Advances in Information and Communication Technology*. Springer Boston, 2010, pp. 203–214. 10.1007/978-3-642-15257-3_18.

[2] Näf, M., and Basin, D. Two approaches to an information security laboratory. *Commun. ACM 51* (December 2008), 138–142.

[3] Romney, G. W., and Stevenson, B. R. An isolated, multi-platform network sandbox for teaching it security system engineers. In *Proceedings of the 5th conference on Information technology education* (New York, NY, USA, 2004), CITC5 '04, ACM, pp. 19–23.

[4] Schneier, B. Does risk management make sense? `http://www.schneier.com/blog/archives/2008/10/does_risk_manag.html`, 2008.

[5] Thompson, M., and Irvine, C. Active learning with the cyberciege video game. In *Proceedings of the 4th conference on Cyber security experimentation and test* (Berkeley, CA, USA, 2011), CSET'11, USENIX Association, pp. 10–10.

[6] Vigna, G. Teaching Network Security Through Live Exercises. In *Proceedings of the Third Annual World Conference on Information Security Education (WISE 3)* (Monterey, CA, June 2003), C. Irvine and H. Armstrong, Eds., Kluwer Academic Publishers, pp. 3–18.