# Skipping the Security Side Quests: A Qualitative Study on Security Practices and Challenges in Game Development

Philip Klostermeyer
CISPA Helmholtz Center for Information Security
Hanover, Germany
philip.klostermeyer@cispa.de

Sabrina Amft
CISPA Helmholtz Center for Information Security
Hanover, Germany
sabrina.amft@cispa.de

Sandra Höltervennhoff
Leibniz University Hannover
Hanover, Germany
hoeltervennhoff@sec.uni-hannover.de

Alexander Krause
CISPA Helmholtz Center for Information Security
Hanover, Germany
alexander.krause@cispa.de

Niklas Busch
CISPA Helmholtz Center for Information Security
Hanover, Germany
niklas.busch@cispa.de

Sascha Fahl
CISPA Helmholtz Center for Information Security
Hanover, Germany
sascha.fahl@cispa.de

## Abstract

The video game market is one of the biggest for software products. Video game development has progressed in the last decades to complex and multifaceted endeavors. Games-as-a-Service significantly impacted distribution and gameplay, requiring providers and developers to consider factors beyond game functionality, including security and privacy. New security challenges emerged, including authentication, payment security, and user data or asset protection. However, the security community lacks in-depth insights into the security experiences, challenges, and practices of modern video game development. This paper aims to address this gap in research and highlights the criticality of considering security in the process.

Therefore, we conducted 20 qualitative, semi-structured interviews with various roles of professional and skilled video game development experts, investigating awareness, priorities, knowledge, and practices regarding security in the industry through their first-hand experiences. We find that stakeholders are aware of the urgency of security and related issues. However, they often face obstacles, including a lack of money, time, and knowledge, which force them to put security issues lower in priority. We conclude our work by recommending how the game industry can incorporate security into its development processes while balancing other resources and priorities and illustrating ideas for future research.

## CCS Concepts

• **Security and privacy** → **Usability in security and privacy**; **Software security engineering**; • **Applied computing** → **Computer games**; • **Software and its engineering** → Software creation and management; • **General and reference** → **Empirical studies**.


*CCS '24, October 14–18, 2024, Salt Lake City, UT, USA*

ACM ISBN 979-8-4007-0636-3/24/10
https://doi.org/10.1145/3658644.3690190

## Keywords

Usable Security; Video Games; Software Development

**ACM Reference Format:**
Philip Klostermeyer, Sabrina Amft, Sandra Höltervennhoff, Alexander Krause, Niklas Busch, and Sascha Fahl. 2024. Skipping the Security Side Quests: A Qualitative Study on Security Practices and Challenges in Game Development. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3658644.3690190

## 1 Introduction

With billions of generated revenue annually [1], the gaming industry has solidified its position as one of the largest markets in the entertainment sector [2], aiming for easy online availability on every platform, including mobile devices using Android or iOS. Given their immense popularity, video game products have naturally become a prime target for cyberattacks [3], as they are susceptible to the malicious intentions of cybercriminals, threatening billions of players and industry revenues.

Attacks have exhibited an alarming frequency [4], and exploitable attack vectors in games become more prevalent. Modern games, especially so-called Games-as-a-Service [5], often allow users to play online with others [6], and include in-game or associated stores for game related items [7–9]. However, such online features require network components, authentication, and usage of payment application programming interfaces (APIs) of, e.g., credit card providers. Hence, games open up new vectors for criminals to exploit weaknesses to gain unfair advantages in competitive play or to perpetrate fraud, e.g., by illegitimately acquiring in-game currency with tangible monetary value [10, 11]. Consequently, modern games need user account security, fraud protection, anti-cheating measures, and children and minor protection [12, 13]. In addition, game clients can be vulnerable targets, as the game software or related components may contain weaknesses allowing, e.g., injection attacks or remote code execution (RCE) [14–17], putting players at serious risk.

Beyond game clients, attackers may also target the services and infrastructure used during development and maintenance after release. These include denial-of-service attacks on servers, leading to downtimes for billions of players and significant financial

damage [18–20]. Moreover, attacks against data storage to access sensitive user data and payment information [20–22], or the theft of valuable assets such as source code and secret information [23–26] further put companies and players at risk.

Video game development is a complex process under high time, cost, and social pressure, especially for developers [27, 28]. At their core, games are a particular form of computer software, with the specialty of being *fun* and played by a vastly varying user group of all ages, genders, or socioeconomic backgrounds [29]. The development commonly involves many roles in a *game studio*, e.g., *managers*, *producers* serving as intermediaries between roles [30], external *publishers* often being the primary funders contracting studios, and *developers* creating a game. Furthermore, many additional roles can influence security decisions, e.g., *backend*, *infrastructure* or *security specialists*, or *quality assurance (QA)*. Depending on the size of a production, roles of individual stakeholders may further overlap, e.g., small *indie* studios may even concentrate the duties of all these roles onto only a few shoulders. Particular development challenges are, e.g., a broad skill set required from developers, *feature creep* [31], rapidly and fundamentally changing requirements, projects and codebases, and complicated workflows and tooling. These obstacles lead to halted development, staff or feature cuts, a lack of thorough testing, and the release of unfinished products, completed only through patches after release [32, 33]. Additionally, a sheer countless amount of security incidents from the industry [34–38] and a lack of research on the subject suggest significant and unexplored underlying problems within the industry's security practices.

To conclude, as the industry has traditionally focused on entertaining users, ensuring their games are secure is an emerging issue becoming increasingly challenging with the addition of new and intertwined features. As previous work on security in video games is scarce, it commonly does not investigate the human factors that influence security decisions. Furthermore, there is no compiled list of critical security areas relevant to the game industry, and no discussion of the industries' security priorities. While there are scientific works that acknowledge unique challenges in game software engineering (SE) [32, 33, 39], to the best of our knowledge, these challenges previously did not focus on security. It remains unclear to what extent security is a guiding principle in development and to what degree studios and publishers are aware and concerned with the secure implementation of their software. To address this gap, we examine game development experiences, challenges, and practices by interviewing seasoned stakeholders from studios and publishers about game security. Further, we contextualize their experiences with related work for secure SE and provide directions for stakeholders and future research. Our work addresses the following research questions:

- **(RQ1):** *How aware are and what security areas do studios and publishers deem most important?* – Our findings show that developers are aware, but prioritize user-facing features and issues over security. Nourishing a fun concept has utmost priority. Security measures are motivated by threats to revenue and to adhere to data protection laws.
- **(RQ2):** *What are studios' and publishers' challenges regarding respective security areas?* – We identified several challenges, including a severe lack of time, budget, team size and knowledge. Security measures are influenced by unique key characteristics and areas, such as game genre, release platform and cheating. The fast-paced nature of the industry leaves it unable to address security thoroughly.
- **(RQ3):** *What methods, guidelines, concepts, and best practices do studios and publishers rely on to ensure the security of their products?* – As security is rarely proactively deployed, processes are either ad-hoc, non-existent or lacking in terms of available knowledge, training, documentation, and quality. Ready-made, but often black box solutions are utilized wherever possible.

Additionally, our replication package supports our work's comprehensibility, transparency, and reproducibility. We include our study invitations and related postings, the pre-survey, the complete interview guide, and our anonymized codebook[1].

## 2 Related Work

We discuss related work in two key areas: Comparing video game development with traditional SE, and studies around security and privacy areas in video games.

### 2.1 Video Game Development & Traditional SE

We present previous work on unique characteristics of game development in contrast to traditional SE. We highlight selected differences to emphasize the circumstances under which games are developed, impacting game security.

Chueca *et al.* conducted a SLR on SE practices in game development and found an increase in publications. Industry priorities appear to be software validation and design. Publications on security issues were much less prominent. They concluded game SE to be an independent scientific domain, differing from traditional SE [40]. Kanode *et al.* contrasted game development with SE, highlighting differences to support the unique needs of game development. They found an enormous complexity of tools and emphasized the importance of management structure [32]. Murphy-Hill *et al.* conducted interviews and surveys with stakeholders to identify differences between SE and game development. They highlighted tensions between creative desires and technical constraints faced by developers. Further challenges include limited testing and deadlines negatively impacting projects [33]. Pascarella *et al.* performed a mixed-method study on 60 open-source game vs. non-game projects. They found significant differences in developer specialization, testing, clarity of requirements, error handling, and code security issues in games compared to non-games. They attribute this to a potential lack of awareness and testing [39]. Difficult development processes were also confirmed by publications that analyzed software postmortems published after releases. These found issues in project management from planning to aftercare, testing and QA, tooling, lack of documentation and knowledge, and stressors such as tight budgets, timelines, crunch work and feature creep [41–45]. Thus, these reports highlight many aspects that influence security practices.

In contrast to previous work that distinguished between SE and game development, our interview study focuses on the development of games and their unique challenges regarding security.

[1] https://doi.org/10.17605/OSF.IO/U5NQK

## 2.2 Security & Privacy in Video Games

While previous work often discussed aspects of game development in general, security and privacy have been addressed more situationally in specific areas. In the following, we provide an overview of existing research. [46, 47]

Agrahari *et al.* analyzed communications on Stack Exchange and GitHub about game development and identified coding anti-patterns. They also found security-relevant patterns such as memory leaks or ensuring overall game security [48]. Ki *et al.* presented a taxonomy of attacks in online games with four layers: client, server, network, and environment. They propose solutions to security problems at each of these layers [49]. In a security risk assessment study, Mohr *et al.* assessed the corporate security of publisher and developer ZeniMax Media Inc. They highlight security issues associated with games, services and servers, system security policies, incident response plans, and employee security training [50].

A widespread security area in gaming concerns cheating [46, 47]. According to Consalvo, *cheating* refers to gain any "unfair advantage" [51]. Cheating and countermeasures (anti-cheating) are widely studied. We provide only a brief overview. Yan investigated the trend towards online games, in which changing security mechanisms are undermined by cheating [52]. Yan *et al.* conducted a systematic classification of cheating and added a taxonomy, showing what vectors can be exploited and what can be gained in the process [53]. Webb *et al.* also provided an overview of known cheats and discussed designs for prevention [54]. Parizi *et al.* analyzed literature for security implementations and challenges in online games. They emphasize the need for security policies and recommend to increase testing efforts, and white box cryptography [55]. Bryant. *et al.* investigated an attack pattern based on network lags with potential cheating effects [56]. Woo *et al.* examined related work and industry practices about the unauthorized acquisition of digital assets, thus cheating game systems, and presented countermeasures [57]. Concerning the technical realization of networking in online games and associated dangers of manipulation, Yahyavi *et al.* investigated the use of P2P architectures in online games, highlighting security aspects related to cheating and discussed techniques for detection [58]. Bryant *et al.* examined the network architectures of online games and how they affect the balance of security and performance in competitive play. In a case study, they evaluated networking concepts, such as client-side exploits or latency and state synchronization [59].

Regarding anti-cheating, Zhang demonstrated detection improvements through deep learning methods [60]. Similarly, Lukas *et al.* leveraged reinforcement learning agents and propose an anti-cheating system [61]. Spijkerman *et al.* compared various AI-driven tools encompassing a selection of classifiers to separate cheating from benign player behavior [62]. Dedicated anti-cheating software has been controversially discussed due to privacy-invasive patterns, such as requiring kernel-level permissions on client operating systems. Bugs in such systems have already led to RCEs [15]. Maario *et al.* addressed this issue by discussing anti-cheating approaches at kernel-level and the impact on security and privacy, and discussed possible non-intrusive alternatives [63]. In a comparable setup, Dorner *et al.* compared several well-known anti-cheat solutions in terms of their behavior compared to malicious rootkits [64].

Regarding mobile games, Yahyavi *et al.* discussed architectural problems of cheating on mobile devices and proposed solutions [65]. Several studies analyzed free and paid Android games and found that the cause for security and privacy issues are, e.g., too many permissions, data collection through third-party tracking, and the security of server connections [66–68]. While previous work often examined specific areas regarding the security of games, ours is the first to directly interview stakeholders with varying roles in the game industry, thereby gathering a novel broad overview of developers' and publishers' awareness and priorities.

# 3 Methodology

In the following, we outline our methodology. We describe our interview development, piloting, and data collection, including details on the interview guide and qualitative data analysis. Finally, we discuss the limitations and ethical considerations of our work.

Our study elaborates on experiences, challenges, and practices regarding security and privacy in video game development. We started with qualitative semi-structured pilot interviews to fathom essential security areas. Later, we included these areas in further interviews for which we invited various roles involved in video game development, including developers, producers, publishers, managers, and freelancers, to gain cross-cutting insights. Overall, we conducted 20 semi-structured interviews.

## 3.1 Interview Guide Development and Structure

To develop topic areas for our pilots, we brainstormed questions with regard to our RQs (cf. S. 1). In addition, we reviewed previous work to identify critical security areas [49, 69].

**Piloting the Interview Guide.** We piloted the initial interview guide with two industry participants from our professional network. We incorporated their valuable feedback and, where necessary, refined our questions, added clarifications, and follow-ups. Therefore, we followed established practices in our field, constructing semi-structured open questions that transition into more closed questions to explore specific subjects in-depth [70, 71].

During our pilot interviews, it became apparent that, given the multitude of roles involved in the game development process, it was not sensible to ask each question in every interview. Publishers, e.g., have a deciding position, but are generally less familiar with the technicalities of game coding. Therefore, we created different categories for our interview questions. We used these categories to decide whether questions were applicable, depending on the respective participants' backgrounds. Firstly, we annotated highly business-related questions as *Managing, Producing, and Publishing (MPP)*. These questions were tailored towards participants in management or publishing, who made executive decisions. Secondly, we marked questions tailored towards developers and programmers with the category *Developers (D)*. In these cases, interviewees were more likely to have in-depth insights into coding processes. We further used *Freelancers (F)* to label questions for participants who were at the intersection of managing their own work and game development, and the base category *All (A)* to label questions fitting for all participants. This setup was guided by other qualitative interview studies that used a similar approach in asking slightly varying questions to different stakeholder groups [72–74]. In using

this approach, we were able to gain deeper insights into certain aspects of game development and draw connections that would be otherwise difficult to explore, such as between the different perspectives of stakeholders on security issues (cf. S. 4.2), e.g., publishers deciding a contractual background and developers working within these constraints.

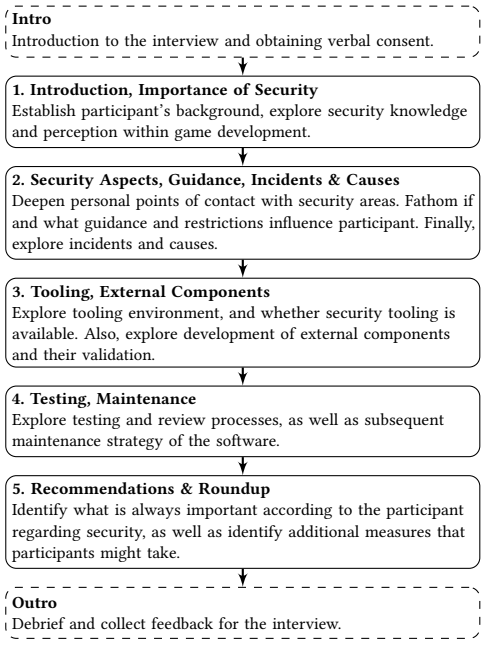


**Figure 1: Structure of the interview guide, consisting of five blocks in which the areas, importance and application of security in game development are questioned.**

**Structure of the Interview Guide.** In the following, we describe the structure of our final interview guide (see Figure 1).

**1. Introduction.** First, we asked our participants about their game projects and work environment to create a basis for further discussion. We were also interested in discussion and decision processes during development. Moreover, we were interested in how security is perceived in the industry from their perspective and whether and in which ways this perception has changed over time.

**2. Security Aspects, Guidance, Incidents & Causes.** In the next section, we focused on the importance of security in their daily work. We asked about security-related hurdles that our participants face and for a self-assessment of the security of their work. Then, we asked about the security-critical areas we had identified during the piloting. The following key areas have been addressed:

*Asset Security & DRM.* We asked about security mechanisms and procedures to protect against, e.g., theft of game code, graphic assets, or other intellectual property.

*Networking.* Video games can be categorized as singleplayer, multiplayer, or a combination. In addition, various online features can be integrated, such as social logins or store functions that require a network connection. For this reason, we were interested in how related functions are realized in the area of networking. We were also interested in whether and how network communication was encrypted and considered secure.

*Cheating & Anti-Cheat.* A relevant industry area we noticed in our pilots and related work concerned cheating. Thus, we were interested in what measures were taken to prevent game manipulation.

*User Privacy & Endpoint Security.* Lastly, we asked about procedures related to players' personally identifiable information (PII), financial and telemetry data that the game could collect. This also included how the game handled player-generated content, e.g., if chat messages were sanitized.

For each area, we also asked what obstacles restricted the implementation and what guidance was available to the participants. We then asked about participants' security education or training and details on their workplace, such as the code documentation or who they consulted for security advice. Further, we were interested in incidents that the interviewees might have experienced, and if so, we inquired for more details on what happened, why, how the issues were handled and the implications.

**3. Tooling & External Components.** In the next section, we addressed the development process. We asked for tooling and the participants' productive environment, including their integrated development environment (IDE) or potential use of game engines. These bundle necessary libraries and packages and often provide a graphical interface for real-time game preview. Developers typically choose between creating their own technologies or using a standard game engine, such as the closed-source Unity Engine [75] or the source-available Unreal Engine [76], based on project scope, code reuse, or needed features. Because game engines bundle such essential components and are rather common [77], we deemed them important and included them in our questions. We were further interested in other integrated external libraries and software, outsourced components, and, in both cases, the associated issues of validating their security and the necessary trust.

**4. Testing & Maintenance.** The next section of our interview guide discussed the testing and maintenance of video games. Frequent reports from the video game industry argue that video games often do not reach full functionality after public releases [5, 78, 79]. We were primarily interested in typical testing strategies and processes for video games, whether they are tested explicitly for security, and how aftercare in the form of updates is realized.

**5. Recommendations, Roundup & Debriefing.** In the final part, we asked the interviewee if he wanted to discuss anything else. If not, we turned off the recording, thanked the participant, and clarified compensation details. Finally, we asked for general feedback.

We continuously revised the interview guide while conducting our interviews, e.g., we added examples or made minor changes to the wording of the questions if there were any ambiguities. The final interview guide is part of our replication package (see S. 1).

## 3.2 Participant Recruitment

We aimed to recruit various experts involved in the game development process. As we expected these experts to be hard to reach as, e.g., non-disclosure agreements (NDAs) are not uncommon in the industry, we recruited over multiple channels and used a mixed-methods approach of snowballing, random, and systematic sampling.

We started by advertising on public Subreddits and Discord servers dedicated to game development. Before we published our call for participation, we asked the moderators if posting about our study was allowed. Next, we contacted national associations connecting development studios and publishers, namely Entertainment Software Association (ESA), Interactive Software Federation of Europe (ISFE), International Game Developers Association (IGDA), The Independent Game Developers' Association (TIGA) and Verband der deutschen Games-Branche e.V. (GAME) to spread our call for participation. Thirdly, we used Twitter, LinkedIn, and email to contact studios and candidates. We further searched for publicly accessible contact addresses of open-source game projects and sent them study invitations. Finally, we recruited online through Upwork from the list of freelance software developers specializing in video game development. We provide the cover letters and postings used in our recruitment as part of our replication package (see S. 1). Candidates had to complete a pre-survey to collect demographic data and general information about their profession and security experience. Overall, our recruitment was complicated and only yielded a low response rate. We reviewed the pre-surveys of all interested candidates to see if they fit the study, meaning if their work could be considered game development and if we deemed them able to answer our questions. We invited candidates by email, and used Calendly for scheduling. The interviews were then conducted through our institutional instance of Zoom. Figure 1 shows participants' demographics and the length of each interview. The average length was 01:06:41. We offered participants a compensation of $100 via PayPal, Upwork, or an Amazon voucher from a country of their choice. This amount was chosen due to the difficult recruitment and the generally high average salaries of developers [80]. Overall, we conducted interviews until we reached thematic saturation and no new topics emerged [81].

### 3.3 Data Analysis

We leveraged iterative, open coding to analyze the transcribed interviews [82–84]. Each interview was coded by two researchers who merged and discussed their codings after each interview, resolving conflicts and iteratively created a codebook through selective and axial coding [85, 86]. We revisited completed interviews whenever we introduced a new code. Four researchers were involved in the coding, during which the codebook and structural ambiguities were regularly discussed and resolved. With this approach, we followed an established methodology to analyze interviews qualitatively [87–91]. Similarly, we omitted the calculation of inter-rater reliability, as our analysis method was based on regular discussions and merge meetings to resolve conflicts [87]. For the subsequent analysis of the coded data, we used discussions and affinity diagramming [92] to group the codebook thematically and extract dominant themes to answer our research questions.

### 3.4 Ethics & Limitations

**Ethics.** The design of our study, including the pre-survey, interviews, processing, handling, and analysis of participant data, was approved by our institutions' Ethical Review Board (ERB) and crafted in consideration of the ethical principles of information and communication technology research as discussed in the Menlo Report [93]. Participants accepted a consent form before partaking in our study and were informed about the interview procedure and the General Data Protection Regulation (GDPR)-compliant data collection and processing. At the beginning of each interview, we emphasized that participants could skip any questions they did not want to answer and tried to make them as comfortable as possible. We stored participant data in our self-hosted cloud and utilized the GDPR-compliant transcription service Amberscript. We anonymized all participant quotes or identifying data for our publication.

**Limitations.** While we aimed for a diverse and even recruitment, our study has cultural limitations and biases, as we found Europeans more likely to answer our call for participation. We tried to mitigate this by distributing our call for participation via online communities such as Discord or Reddit, and were able to overall recruit 40% from different regions than Europe. Overall, the participants in our study tended to origin from western industrialized nations nevertheless.

Furthermore, several other biases are typical for interview studies, including self-selection, self-reporting, the probability of social desirability, and skewed memories of our participants. This might lead to over- or underreporting in our results, preventing them from generalizing to the greater SE field. However, a lack of generalization is typical for a qualitative and exploratory study like ours. However, as we recruited a broad participant sample with different backgrounds and professions, we are convinced of the high quality of our findings.

## 4 Results

In this section, we describe the results of our interview study. First, we provide basic information about our participants, including their professional background, work environment, and projects to illustrate their roles within the industry. Following this, we discuss more specific results regarding the influence of the game industry on development and security, and our participants' stance on various security measures for video games. Apart from the participant demographics, we refrain from reporting numbers and instead provide qualifiers to not suggest the generalizability of our results and further protect our participants' anonymity. We provide a mapping to percentages in the Appendix in Figure 2. Note that some interviews were held in German. Consequently, we manually translated quotes from the respective interviews without altering the original statements.

### 4.1 Participant Demographics

Overall, we conducted 20 interviews with participants from 15 different nations (60% EU, 25% NA, 15% APAC). We provide details on their background in Table 1. Participants sometimes discussed older or multiple projects, so numbers may not add up to 20 participants. In accordance with our consent form, we provide only aggregated information for demographics such as location or age.

Concerning their professions, we were able to interview a broad set of study participants. Most commonly, twelve participants mentioned varying experience levels in developing or software engineering, nine of which included leading roles. Further, two participants worked primarily in the specialized area of backends and platforms or had overlaps with work in this area. Others have mentioned activities in this area. Five participants reported working in producing

**Table 1: Overview of interviews and participants' self-described demographics. As we used several channels to advertise our study (cf. Section 3) and did not ask participants how they found us, we use the term "Open Call for Participation (Open CfP)" for cases where we cannot accurately state the precise recruitment.**

| ID | Duration | Location | Age Group | Recruitment Channel | Industry Exp. (yrs) | Occupation | Work Env. Size (ppl) | Sec. Exp.[1] | Interview Category[2] |
|---|---|---|---|---|---|---|---|---|---|
| P1 | 01:04:13 | EU | N/A | Prof. Contact | 10+ | Producer$^{\triangle}$ | 1–4 | 2 | Pilot |
| P2 | 00:49:58 | EU | N/A | Prof. Contact | 10+ | Security Specialist | 20–49 | 3 | Pilot |
| P3 | 01:00:49 | NA | 30–39 | Open CfP | 6–10 | Manager | 50–999 | 1 | MPP |
| P4 | 00:39:12 | NA | 20–29 | Open CfP | 1–2 | Compliance Specialist | 50–999 | 3 | MPP |
| P5 | 00:58:55 | APAC | 30–39 | Upwork | 10+ | Developer/SE$^{\triangledown\triangle}$ | 10–19 | 3 | D,F |
| P6 | 01:01:47 | APAC | 30–39 | Open CfP | 6–10 | Developer/SE$^{\triangledown}$ | 1000+ | 2 | D |
| P7 | 01:10:13 | EU | 30–39 | Open CfP | 6–10 | Developer/SE | 5–9 | 3 | D |
| P8 | 01:11:17 | EU | 30–39 | Upwork | 10+ | Producer$^{\triangle}$, Developer/SE | 20–49 | 2 | A |
| P9 | 00:59:58 | EU | 30–39 | Open CfP | 10+ | Manager, Game Designer$^{\triangledown}$, Publisher | 50–999 | 2 | MPP |
| P10 | 00:40:00 | EU | 20–29 | Open CfP | 1–2 | Developer/SE$^{\square}$ | 1000+ | 2 | D |
| P11 | 01:00:06 | EU | 30–39 | Upwork | 6–10 | Developer/SE$^{\triangledown}$, Game Designer$^{\triangle}$ | 1–4 | 3 | D,F |
| P12 | 01:17:32 | EU | 20–29 | Upwork | 3–5 | Developer/SE, Game Designer$^{\triangle}$ | 5–9 | 2 | D,F |
| P13 | 01:10:59 | EU | 50–59 | Prof. Contact | 10+ | Developer/SE, Game Designer$^{\triangledown}$ | 20–49 | 2 | MPP |
| P14 | 01:01:58 | NA | 30–39 | Upwork | 3–5 | Developer/SE$^{\triangledown\triangle}$, CTO | 1–4 | 3 | D,F |
| P15 | 01:34:34 | NA | 40–49 | Upwork | 10+ | Quality Assurance$^{\triangledown\triangle}$ | 50–999 | 2 | D,F |
| P16 | 01:03:53 | EU | 20–29 | Open CfP | 3–5 | Developer/SE | 1000+ | 2 | D |
| P17 | 02:01:10 | NA | 20–29 | Open CfP | 3–5 | Platform Engineer$^{\diamond}$, Developer/SE$^{\square}$ | 50–999 | 3 | MPP |
| P18 | 00:54:51 | APAC | 20–29 | Open CfP | 3–5 | Developer/SE$^{\triangledown}$ | 50–999 | 2 | D |
| P19 | 01:29:07 | EU | N/A | Snowball | 6–10 | Security Specialist | 1000+ | 3 | A |
| P20 | 01:03:05 | EU | 30–39 | Upwork | 6–10 | Quality Assurance | 50–999 | 2 | D |

[1] Order (from low to high): *1: Little – 2: Some – 3: Considerable* [2] *MPP: Managing, Producing & Publishing – D: Developers – F: Freelancers – A: All*, as described in 3.1 $^{\triangle}$ Also freelancing / consultancy activities $^{\triangledown}$ Team lead position $^{\square}$ Backend activities / DevOps $^{\diamond}$ Senior level experience

and managing teams, and seven also did consulting or freelancing in this area. Moreover, two participants worked in QA, and three hold specialist positions, e.g., in security or compliance. A total of eleven participants were part of a company, and 13 stated they work in a team, while the others reported mainly working alone.

Concerning their working environment, eleven participants stated that they worked on games that contained multiplayer components, and six had worked on singleplayer games. The specific game types included action games, building games, RPG games, casual and hyper-casual games, and games containing different gambling or casino aspects. Regarding targeted release platforms, 18 worked on PC and mobile games, and ten projects were web or browser-based games. In addition, six projects were released for game consoles such as Microsoft Xbox, Sony PlayStation, or Nintendo Switch. Eight further reported using dedicated IDEs, such as Visual Studio. Regarding game engine usage, twelve participants reported having contact with Unity Engine, five reported contact with Unreal Engine, six described internal engines and five reported using various other standard engines. While these can include features to increase a game project's security level, we also heard complaints that, in contrast to non-game development tools, engines abstract security too much and do not communicate code security issues adequately.

## 4.2 Industry Dynamics Shaping Game Security

Below, we present our findings on the special circumstances within the game industry that influence how games are developed and affect security.

**Faced-paced Industry.** Many participants explained the urgency to land hits, as the studio would otherwise succumb, describing a fast-paced mentality within the industry. One participant stated that "*you need to make a hit because [the] game industry is a hit-driven industry, and to make a hit you have to make many attempts so rapidly*" (P6). In relation, a few participants mentioned producing rapid prototypes. The speed was also reflected in a short software lifecycle described by a few. One participant contrasted the game industry to other software: "*For something like SAP, it's always about the long term. You're building systems, if not for eternity, then for the next 20 years […] In video game development, it doesn't matter, it's mostly fire and forget*" (P1). Some participants mentioned that this problem particularly affects so-called legacy games when the production of a game is concluded and the projects' teams are downsized or disbanded, but online features and servers are still available. In these cases, support becomes limited or stops completely, potentially leading to unsolved security issues.

**Disconnects With Security.** We asked where security plays a role in discussion processes and interrelationships between developers, managers, and publishers. Some participants mentioned that their studios are bound by their publishers to integrate security-relevant features such as logins and authentication, advertising and analytics, or payment processes, through, e.g., APIs. One participant described that "*whenever we publish a game […] we do have a list of things that we give to the developers, e.g., these are all the ad APIs that you have to implement which are GDPR compatible. […] You cannot bring one of your own*" (P9). However, this sometimes led to problems as developer teams may use unorthodox tooling, preventing them from implementing the demanded features.

Further, some participants reported that publishers provide their own software development kits (SDKs) that interface with plenty of APIs, but there is barely any possibility for the developers to examine their source code or assess their security. Also, the integration of the SDKs is not trivial, as one participant stated that "*most of the time, you try to integrate it and then nothing works at all. […] When we got […] API providers that said it was plug and play, it was never plug and play*" (P1).

A few participants stated a discrepancy between developers and higher-ups in their knowledge and communication about the

technical implementations, also regarding security. One participant described this disconnect between developers and publishers: "*if you work with a developer, it's a lot easier because they have full intelligence regarding […] the entire technical stack […]. If you go to a publisher, it does happen that they have no clue about the source code and technical stack. They have no insights into how mature is the code from a security point of view*" (P2).

**Security is Done on Demand.** Taking care of security as early as possible and incorporating security considerations into a development plan from the outset was not a common practice and only formulated by a few participants. Some participants described that they seldomly talk about security, it "*usually only comes up in conversation when the client specifically asks for it. […] Otherwise, we do a very bare minimum of checks*" (P18). This was driven by an economic view of security and the necessary precautions, questioning if security is worth the extra-cost. In particular, we also found for about half of the participants that there was only a reactive interest in many security or privacy-related problems, practicing aftercare rather than prevention. About half of our participants cited the fear of public outcry, and some participants explicitly mentioned monetary losses or a possible lawsuit as reasons to implement and care for security.

During development, many developers suggested implementing security features and mechanisms themselves, and some also made the final decisions regarding implementation. A few reported that they would decide in consensus and discuss feature implementation during team meetings. However, some other participants reported that they have certain roles or departments within their hierarchy, e.g., a technical director, who must approve final implementations. About half of our participants reported to have specialized departments such as QA or security departments. But only some mentioned dedicated security testing.

**Time, Money & Size are the Biggest Enemies.** In line with the fast-paced industry, almost all participants cited time or budget pressures as significant challenges influencing all areas around security. One security specialist added that "*security analysis assessments require time. We're talking about even a month-long, only dedicated to that task. […] after you've done with it, you have to wait for the team that is responsible to fix that […]. Then you look at the calendar, and it's been four months just for that*" (P19). As security is expensive and takes up a lot of time, participants reported that it was neglected. About half of our participants admitted that they did not have time to research or fix security issues, or went with security-wise bad decisions. This problem was also reflected in testing, as some emphasized that testing is very complex due to many test cases and not feasible to do in-time. Further, some participants stated to not audit chosen tooling or integrated assets, because of time pressure. Instead, many participants integrated tools they perceived trustworthy, primarily when published by a well-known company, such as Microsoft's PlayFab [94]. Only one participant mentioned taking time and deleting unknown code in integrated assets, also to get their team used to this procedure.

A majority of our participants also mentioned the size of development studios is influencing security. Especially the indie games sector has difficulties to implement security, as they have less resources and have to consolidate necessary development roles into only a few people.

**Guidance on Security is Scarce.** We also asked participants about helpful resources, guidance, and industry security knowledge. Many participants used Google to answer security questions. Moreover, many participants utilized special forums or communities, such as the official forums of standard game engines, Discord, Stack Overflow or even underground forums, in this case to research on cheating. Further, ChatGPT has been used by a few participants. For external components, the respective official documentation was named as the main source by about half of our participants. Only a few mentioned internal company guidelines. Some participants explicitly stated to have no guidelines. The majority of our participants also had not attended any security courses or training.

Only some participants had access to security specialists or specialized security departments, while a few participants mentioned external security specialists in the field of penetration testing with whom they had contact. Further, only a few spoke of lawyers or legal teams they could contact for advice.

Regarding project documentation, some participants noted that their projects were documented. Overall, a few explicitly stated that they had no documentation. About half of our participants stated that they were dissatisfied with their documentation situation. One developer stated that they "*have some documentation that is very early on in the project on how we would [have liked] to do stuff, but that doesn't mean that we did stuff accordingly. I don't think the documentation has been updated in forever*" (P16).

Some participants were unhappy that "*there isn't a de facto go-to place*" (P16) for discussions about video game security. We assume this is also caused by the industry's secrecy regarding game internals, as stated by many of our participants. One participant added, that: "*We need to have lifted NDAs on this subject [of security] between peers because otherwise, we can't talk about solutions. […] We cannot find the commonalities in what we need and do the requirements analysis for [solutions]*" (P19).

> Key Insights. The game industry is fast-paced and business-oriented. As security does not directly influence revenue, it is considered a secondary goal and often only happens when absolutely necessary. Resources such as time, money, guidance, and specialized personnel are rare, especially for smaller studios, leading to a further de-prioritization of security and issues with testing the software besides gameplay.

## 4.3 Relevant Security Areas

During our interviews, we identified five security-related areas in the game development process. We address each area and outline attack vectors as well as associated constraints, challenges, and solutions. We found that the priority of many areas depends on the type of game, as stated by about half of our participants. For example, network security was less relevant for games without online features.

*4.3.1 Anti-Cheat.* Most of our participants talked about encountering (anti-)cheating "*on a daily basis*" (P9), making it one of the most common security areas. Also, most reported incidents were related to cheating, and about half stated that they perceived their game's security the weakest regarding anti-cheat. We grouped cheating

into two categories: First, the majority of our participants mentioned cheating as gaining unfair advantages over other players. Second, many participants referred to users cheating the system or game, e.g., by unlocking exclusive paid content without paying for it. In some cases, however, problems may overlap.

When asked if they had previously encountered software-based anti-cheating detection, some of our participants mentioned tools such as BattlEye, Easy Anti-Cheat, Valve Anti-Cheat, and others. Some mentioned using in-house solutions that are also aimed to counteract cheating, while some participants stated not using any type of cheating detection. Among them were a few who explained that they were working on singleplayer games and therefore did not require anti-cheat, as in their type of game, cheaters would harm no other users.

A few participants stated that they consider standardized anti-cheat solutions to be only partially effective in solving the cheating. One security specialist attributed this to the fact that "*anti-cheat is not going to do anything about the vulnerabilities that are in your game. […] It's just a matter of time of getting around that anti-cheat and then exploiting it*" (P2), implying that cheating issues are rooted within code security. In contrast, many participants stated that cheating does not necessarily stem from insecure or broken code, but, e.g., some reported logic flaws. This was partly due to misconceptions, as coding issues were not interpreted as such.

For example, we learned of an incident with the verification logic of the player's interactions with the service endpoint. Another participant spoke of a "hack" where players could obtain paid content without paying. This was caused by an unsecured web service that accepted requests without validation. An additional report referred to an early closed beta test where players were able to receive a much higher number of special characters than intended, which our participant attributed to a bug in the server structure.

Almost all mobile games developers reported that using existing anti-cheat software is difficult, since it is usually only offered for other platforms, requiring mobile developers to create their own, in-house solutions. However, this was described as problematic as well: "*[the] main problem is to create from scratch. […] It's hard to create in short time really good technology*" (P12).

In general, cheating was perceived as a tedious and difficult problem by many, to the point of about half stating that cheating cannot be solved and that it is "*a constant arms race that we are never really winning*" (P9). To combat this, some participants mentioned researching the cheats' origins in an attempt to reverse-engineer them to know which parts of the game are abused. However, this approach requires programming or even reverse-engineering skills, which can be too much work, especially if the cheating issues are not directly threatening revenues.

As root causes for cheats are manifold, some participants suggested a "Swiss cheese model" approach as the most sensible solution. This model from risk analysis requires not one, but several solutions to address all weaknesses within an organization's security [95]. Additionally, some participants mentioned consistency checks to, e.g., verify in-app purchases. Many also mentioned a server-authoritative design, in which a game or backend server must verify every "move" of a player and rejects suspicious requests as cheating, i.e., the client is never trusted. However, this model is also associated with high requirements, because it makes development and debugging more complicated, as a few participants noted. Nevertheless, one security specialist argued in favor of a more elegant and situation-dependent separation of concerns-variant "*You make that hybrid […] You find a compromise*" (P19). They described a model in which the game is only server-authoritative where necessary, e.g., when directly interacting with other players. In this way, cheating issues could be remediated, while at the same time, server load is limited.

> Key Insights. Cheating was one of the most common security issues for participants and thus, several participants applied anti-cheat measures. However, countering cheating was also perceived as tedious and almost unsolvable especially with standard tooling, as it does not help when cheaters exploited vulnerabilities.

*4.3.2 Asset Security.* Another important area was the protection of various assets and game code. The term *asset* describes any resource that is necessary to create the full game, including but not limited to art and graphics, sound files, or resources such as animations and textures, but also source code. These assets are the intellectual property of games and can hold brand memorability. Therefore, security measures are necessary to protect their brand from abuse.

About half of our participants explained that they do not secure their assets. The majority of these felt that such measures are too difficult or that it is even impossible to prevent assets from being stolen or leaked. This was partially because, unlike source code that can be obfuscated, it was harder to refuse access to something players could see or hear. In some cases, games can be decompiled, or assets are, by default, available in a game's or website's source code. The other half of our participants stated they would secure assets. Some participants mentioned using code obfuscation, as one described: "*scramble the code as much as possible. Even if it gets decompiled, it will be too unreadable*" (P8). Half of these participants also mentioned using custom code obfuscation solutions. A further half indicated using other code obfuscation methods, such as compiler-level flags, external services, or built-in tools in game engines. Only a few stated measures for securing non-code assets, e.g., encrypting their graphics.

Many of our participants reported incidents concerning stolen assets. Attackers then publicly reused assets or leaked code, e.g., by trying to run illegitimate game servers. More commonly, participants mentioned reuse in the mobile games sector, when code, assets, or entire games were stolen, slightly modified, and re-released, to generate revenue with the stolen product.

To counteract unauthorized reuse, the participants stated solutions, such as silently checking for the existence of specific variables and cryptographic signatures of the original build.

> Key Insights. The second most common security issue participants reported was the theft or re-use of assets, which were, in the most extreme cases, abused to rebuild the game to earn money. About half of our participants reported protecting their game assets using measures such as obfuscation or asset encryption. However, participants felt it was impossible to truly secure assets, especially regarding art and sound.

*4.3.3 Network Security.* In the following, we present our findings about the games' backends, networking components, connective server code, and the relevance and extent of security measures.

**Servers, Services & Data Storage.** Almost all of our participants reported having contact with network communication within their game projects. Starting with the backend, most of our participants mentioned authentication services related to their games. This included many that are adjacent to a game, i.e., authentication services either from their organization, e.g., through a company storefront, or other single sign-on services, through which data is passed to the game, but it does not hold it. In addition, some also mentioned other web services, such as connections to payment providers for, e.g., in-app payments. A few also mentioned services such as analytics used to collect telemetry data or crash reports. Moreover, a majority reported database systems in the backend, of which a few also contained financial user data. Some participants told us about game servers, that served different purposes, e.g., some handled multiplayer features, while others served as APIs for game-related requests.

Regarding networking code, many mentioned implementing and handling it by integrating external SDKs. About half of our participants used a wide array of ready-made tools and platform SDKs, including advertising, authentication, single sign-on and social logins, and payment or crash analytics. They sometimes also expressed being almost overwhelmed by the available software: "*I've worked with so many analytics SDKs that I can't really name them all*" (P16). However, only some participants mentioned experience with networking tools designed explicitly for game traffic.

While various ready-made tools were mentioned, these were often insufficient for the respective projects' purposes. Hence, about half of our participants mentioned situations where networking code had to be adapted or written from scratch. This ranged from sophisticated high-level custom web service endpoints, e.g., for handling game items, currencies, game sessions, and states to custom implementations of standard network protocols, handshaking mechanisms, data streaming, sockets, and fully custom protocols.

**Network Encryption.** To protect networking traffic, about half of our participants indicated using encryption in game projects, of which about half referred to standard SSL encryption through, e.g., HTTPS. One participant suggested using ready-made software to relieve developers from writing the respective functions for e.g., authentication, payments, or data synchronization themselves and the need to take care of encryption. We found some participants to not use encryption, of which a few restricted this to game traffic that they considered harmless.

When participants mentioned custom networking implementations or abstaining from encryption, a common argument was to avoid performance decreases due to security measures that lead to increased latency times or bandwidth costs. This was described as unacceptable, especially in the mobile gaming sector. Instead, customized networks were implemented with only simple encryption or without security in mind. However, it was also acknowledged that custom networking code requires highly trained and "*extremely good people to know what they're doing*" (P8). Keeping a balance between security and performance formed a primary obstacle.

Participants reported several network security incidents , mainly about servers, services, and data storage. Typical problems included, e.g., unknown web requests that could indicate possible ping sweep or other scouting activities, denial of service attempts, database injection attacks, data breaches, and authorization and authentication flaws. The latter included incorrect request validations or sockets not appropriately closed after sessions.

> Key Insights. Almost all participants had contact with connective code. While ready-made tools were used, participants found them insufficient and required custom solutions. Only about half of our participants used security measures such as encryption, as game traffic was perceived as harmless, and encryption decreased the game's performance, which was a priority.

*4.3.4 Game Client Security.* Below, we present our results regarding the security of game clients and endpoints, especially code security and practices.

To explore this, we queried our participants about software stability. Overall, we consider stability as avoiding programming deficiencies, such as memory corruption, that could lead to vulnerabilities. About half of the participants reported stability as an essential goal, as one stated that "*it's very important to have the game as bug-free as possible because that obviously hurts the game experience*" (P11). However, we discovered a disconnect between our understanding of code security as essential to stability and that of our participants. One participant noted: "*[S]tability doesn't necessarily have to do with security, but with whether the game is executable*" (P13). Stability problems were described as impacting the user experience and, therefore, being important, but were never linked to typical security flaws potentially causing these issues.

Further, one security specialist recalled an incident where they noticed that the interconnection between game clients could be abused and that one "*could execute arbitrary code on other people's games just because they were online*" (P19). Interestingly, there was an unrelated incident with a similar attack pattern regarding the well-known title Dark Souls [14].

Some participants reported incidents related to their code base. In one case, an endpoint containing logic flaws was described, which accepted scores multiple times and was accordingly exploited. Another participant outlined that during code reviews, their security department frequently rejected them, but fortunately, they were able to fix weaknesses before incidents occurred.

In addition, the care with which good quality code is written depended on further factors, aligning with those described in S. 4.2. This includes, e.g., time and money constraints, as one participant stated: "*You miss out on writing good security measures and writing good code to handle because you have your boss on your head to get it delivered*" (P18). One security specialist located vulnerabilities especially in "*the custom nature of development on top of whatever framework, programming language, or library you're using*" (P2).

Some participants mentioned that "more diligence" is needed to counter code security issues. To achieve this, it was suggested to improve the processes from the beginning of video game development. One participant proposed that developers utilize ready-made solutions, which would be "*usually the most optimal way to do stuff*" (P16), thus, not reinventing the wheel in coding.

> Key Insights. Game stability was perceived as important but unrelated to security. While participants had strong feelings for a bug-free game experience, the debugging processes were seldom connected to security or vulnerabilities. Similarly, previously established resource constraints limited the developers' abilities to deliver qualitative and secure code.

*4.3.5 User Privacy.* Another important security area is the processing and protection of user data.

**User data.** Many of our participants had to deal with user-related data and PII in particular. A few spoke specifically about account data such as passwords, hashes, or alternative login methods using asymmetric cryptographic methods. In addition, a few also mentioned data concerning users socializing with others, e.g., chats and leaderboards. A few also clarified that user data is seldomly stored within the game. Instead, "*[…] the platform […] holds onto the data, and the game or applications are given access to it*" (P15) through interfaces to, e.g. centralized storage or external providers. About half of our participants showed an exceptionally high level of awareness in this area, emphasizing the importance of carefully handling user data. A few were also in favor of data minimization, explicitly stating to be against a flat-out data collection without a specific use for it.

Another central area has been financial user data, which a few of our participants stated to self-process or store. However, the majority of participants reported that financial data is mainly processed through attached systems, e.g., payment APIs.

**Telemetry data.** A majority of participants described collecting and processing telemetry data. First, a few participants mentioned behavioral user data, e.g., time spent in in-game areas such as shops. Second, some developers aimed to improve anti-cheat systems, e.g., by detecting suspicious client behavior that might unveil cheating attempts and be able to punish clients. Finally, many participants used information about clients or software for troubleshooting and debugging. For the latter, games collected in-depth user hardware data, down to installed driver versions. This was also referred to as "*fairly standard practice*" (P3). Security concerns were only formulated by one participant stating that "*if game handles any identifiable data, PII, you would need to take care to upload them securely as well*" (P7). However, concerns did not appear widespread, as P3 described: "*I've never encountered any security concerns throughout that process. I've never heard anyone bring any up*" (P3).

Regarding user data and privacy, a few participants reported incidents. One was an injection vulnerability, where user data including password hashes had been exfiltrated. A user input interface was not sufficiently sanitized in an external service connected to a productive database. We found that the most significant impact on increased priority and awareness of data handling were data privacy laws such as the GDPR, COPPA, HIPAA or SOC2, which were mentioned by about half of our participants. One security specialist pinpointed that "*if you get hacked through the game and people access players' information, […] You're also exposed to legal issues later on*" (P19), as any data breaches and related leaks would lead to particular compliance issues if not properly addressed. In this sense, privacy laws force the industry to deal with security and user data, including the protection of minors. Some participants stated that they incorporated support for privacy laws. A few participants further commented that they "*think it's better to have some regulations, […] if your company can create [the game] how they want, it's not okay*" (P12), indicating a certain appreciation of the data protection laws. However, one participant also noted that privacy issues are still seen as a rather minor problem, stating that "*if somehow 2 million user records are hacked, we'll just have to issue a letter of apology. And then that's it again*" (P1). Unfortunately, a few participants also stated that the laws-to-code process is quite cumbersome. The main solution was to enlist the help of legal or compliance specialists or to rely on ready-made software already covering the necessary data handling.

> Key Insights. Games commonly collected both user data and telemetry. Due to the emergence of privacy laws and the consequences of neglecting data protection, developers were aware of the required security measures. They applied them, often with the help of ready-made software or specialized experts.

## 5 Discussion

We gained in-depth insights into the game industry's structure, work processes, and views on security. Security evoked mixed feelings, priorities, and problems in a fast-paced, hit-driven industry. Below, we discuss our findings and relate them to our RQs on the game industries' security awareness and importance (RQ1), and their concerns and specific challenges for security-related areas (RQ2). Further, we discuss methods, guidance, and best practices the industry relies on to ensure the security of their products (RQ3). We conclude with recommendations for stakeholders and the industry and put our work into context with previous research.

### 5.1 Despite Security Awareness, Security Has Low Priority

First and foremost, time, budget, and team size appear to be the most relevant factors. These factors negatively impact security considerations by prioritizing possible vulnerabilities much lower than a game's playable content. External influences, such as publishers, provide security-related input but mainly prioritize security to protect companies' revenue or the public image (RQ1, cf. S. 4.2). Similarly, we find that areas that might otherwise lead to angered customers, e.g., when incorporating anti-cheat software or adhering to data-protection laws are a higher priority. However, beyond that, security is seen as dispensable (RQ1, RQ2).

While larger companies have the resources to handle security issues by hiring designated security or QA specialists, smaller or indie studios often lack the necessary resources or structures (RQ3). Hence, they are more likely to ignore security and less able to deal with incidents after they happen, as all these highly demanding responsibilities are put on a few developers who are often not equipped to handle them (RQ1, RQ2).

Our data shows that even if developers are aware of the inadequate security of their projects, higher authorities may lack knowledge and decide against acting, overruling developers and forcing them to ignore security issues (RQ1, cf. S. 4.2). We also discovered that introducing security into games is no easy task. Security specialists reported that higher-ups are often disconnected from development, leading to difficulties in making security tangible and understandable for them (RQ2). Our results also show that within a video game's software lifecycle, problems can arise regarding responsibility for resolving security issues that may subsequently occur (RQ1, RQ2).

Another noticeable reason for the lack of action is a defeatist attitude towards attacks and vulnerabilities. Due to a lack of documentation, guidelines, tooling, education, and training concerning

many game security aspects, developers describe vulnerabilities as inevitable and incidents as unavoidable (RQ2, RQ3). Implementing security is perceived as highly difficult and time-consuming, so the development focus shifts away from it, regardless of a company's size (RQ1).

Overall, we observed that the game industry is fickle, forcing development to move too quickly to deploy in-depth security measures, such as creating video game threat models from the start and acting accordingly (RQ2). While measures such as early security considerations could mitigate issues that are costly in the long run, the industry lacks security knowledge, leading them to neglect security overall (RQ2).

## 5.2 The Game Industries' Security Perspective

During the interviews, we noticed that our security definitions differed from those of our interviewees. While we assumed security as the system's integrity, in which attackers cannot easily abuse vulnerabilities or coding deficits maliciously, e.g., run custom code, or access company and user data, interviewees typically mentioned the protection of their product and business model.

**Protection of Revenue.** While cheating includes hacking and software abuse, participants mentioned deploying anti-cheat methods not to improve players' security but to protect the studios' revenues or to avoid customer backlash if cheaters made the player experience unenjoyable. Therefore, strategies to encounter cheating or fraud, such as in-house solutions or anti-cheat software (cf. S. 4.3) were mentioned most often. Consequently, we criticize that dedicated anti-cheat software may be too aggressive by requiring extensive permissions on a client device, i.e., kernel-level access, raising severe security and privacy concerns [64]. Indeed, in 2022, a vulnerability in the anti-cheat of *Genshin Impact* proved that it is possible to abuse its kernel-level access for a RCE [15].

Similarly, asset security was discussed as an important area, as studios aim to secure some of their assets utilized in building the game, such as graphics or audio, e.g., to hinder unauthorized re-use or re-publication. However, this measure again protects the studio's intellectual property and does not strengthen gamers' security.

**User-Facing Issues Have Priority.** We find that the parts of the game that users face, e.g., the game design and performance, have a higher priority than security measures. If security is applied, it is typically to improve the user experience, what about half participants confirmed (cf. S. 4.2). This leads to security being neglected and purposely rejected if it conflicts with delivering an enjoyable experience. Encryption is a typical example, which can require high computational resources, increase computation times, and enlarge network packages, negatively impacting the game experience. The stability is only ever interpreted as user-facing. Stability has to be achieved, since a non-functioning game can ultimately incur the wrath of users. The interpretation of stability in terms of code security, i.e., in particular the avoidance of typical programming flaws such as memory corruption, buffer overflows, or other security-critical weaknesses, is not associated with stability. However, incidents such as a code deficiency leading to a RCE vulnerability in the critically acclaimed Dark Souls [14] series show that such flaws exist and pester the industry, putting clients at serious risk.

**Data Handling & Services.** An illustration of the game industry's understanding of security and privacy is the extensive collection and use of user and telemetry data, for, e.g., debugging purposes. While game studios are aware of and following surrounding data protection laws, collecting user data, e.g., for advertising or analytics, was common among participants (cf. S. 4.3.5), as also discussed in related work [67, 68]. However, collecting sensitive data such as PII or telemetry must be handled and stored securely and appropriately, as leaks can have consequences for users if, e.g., attackers know which user client is susceptible to attacks. This becomes more apparent when the data is stored in a backend, requiring additional care regarding infrastructure security. Our findings suggest that while perceived as obstructing and hard to implement, data protection laws like the GDPR force developers to take measures.

**Differences Across Platforms & Game Types.** During our interviews, we noticed that the challenges our participants faced were highly dependent on the characteristics of their development eco-system. This included the target platform of games, as both the client's hardware and the respective storefronts, e.g., on PC or the various consoles, impose differences in terms of capabilities and requirements. In contrast, other application domains typically require development for fewer platforms, e.g., desktop or mobile.

We saw ready-made anti-cheat measures being less relevant on mobile platforms overall. Cheating in these cases more commonly aimed to cheat the system (cf. S. 4.3.1); anti-cheat solutions were typically self-made and based on server-side validations. In contrast, anti-cheat on PC was more commonly a ready-made tool, as reported by our participants.

Similarly, asset security appeared more difficult for mobile or web game developers. Overall, it was described to be much easier to decompile a mobile game or utilize browser developer tools to extract assets. In contrast, games on other platforms were more commonly using custom or proprietary structures, and assets were, hence, harder for attackers to access. Therefore, this type of abuse, which involves stealing and re-releasing games or assets, appears less profitable and uncommon beyond the mobile or web context. Participants who used a ready-made game engine, such as Unity, using a publicly known structure of published games, however, reported difficulties in securing assets regardless of platform.

Beyond security measures themselves, our interviews have demonstrated that the overall development process also differs, especially with regard to the early stages. We found that mobile and web developers tended more towards rapid prototyping before settling for an idea, while participants reported a more structured and mature approach for other platforms. While we did not uncover any direct reasons for this difference, we suspect that it is based on how mobile and web games are commonly simpler or more casual games that aim to target a wider audience of customers. In contrast, other platforms are aimed towards more dedicated players.

We further found that the type of game can have a high impact on security considerations. Online or multiplayer game designs posed additional challenges due to their networking functionalities, and development was further complicated by higher performance requirements when multiple clients needed to interact smoothly. In these cases, security became paramount due to the online connectivity and, simultaneously, more complex due to the surplus of

attack vectors. Additionally, network encryption demanded higher technical costs.

Finally, we also learned that privacy and data protection measures varied between the types of games, as some games required higher permissions or collected more sensitive information from potentially vulnerable user groups, especially in the context of serious or therapy-focused games.

These characteristics lead to different trade-offs and angles from which security is viewed, with shifting awareness, priorities, and measures. Research should consider these key characteristics when exploring specialized areas such as game development.

**Creative Aspects of Game Development.** Our study further revealed several differences between the game industry and other SE. Game developers require a diverse skill set and collaborate with various departments, including music, art, and narrative design. These creative aspects heavily influence the feel of a product and, therefore, its development, while they can also become security-sensitive intellectual assets. The objective to evoke emotional responses and provide entertainment sets game software further apart from, e.g., utility software and their development, which must serve a purpose. This is reflected in the game development process as it is changing the way the software is developed, marketed and sold, i.e., it needs to appeal to a wider range of different individual users instead of selling larger license volumes to, e.g., companies. This was further reflected by the self-image participants displayed about their work, which sometimes lead to a non-economic perception of game development by claiming that game development "is a passion" or that games are "only for entertainment anyway". At the same time, the emotional aspect increases public interest and, thus, pressure on the studio and developers. This is a unique pressure point beyond the lack of resources game development shares with other development fields [96–98]. These multidisciplinary aspects of game development further contribute to a complex security landscape, necessitating tailored approaches to game security that balance creative freedom, player experience, and security measures.

## 5.3 Lack of Security Tooling & Knowledge

Interviewees recalled a wide range of development-related tools, with only a slight overlap, e.g., in using standard engines such as Unity, Unreal, or standard IDEs, however, participants criticized engines' security features not being apparent (cf. S. 4.1).

Security-related tools were rarely mentioned, only by specialists who referred to, e.g., particularly extensive reverse engineering kits for security analysis, which we deem out of scope for a typical game developer. No participant reported using tooling-based automated security support, e.g., vulnerability detectors. Participants whose processes included QA or security departments mentioned that they outsourced relevant tests to the respective departments. However, participants mainly reported that QA primarily tested a game's functionality rather than security.

Overall, our results emphasize that security plays only a minor role, if any when selecting development tools. Knowledge and awareness of secure programming need to be improved, and future work is necessary to aid the game development process.

**Sharing Knowledge.** Our participants reported that finding game security advice is challenging. Moreover, security specialists stated that consolidation of game development security knowledge is overdue, even though first attempts are being made [99, 100]. Better education and improved awareness around the topic were the greatest wishes of our participants. Participants desired a shared community to establish places where diverse industry stakeholders could exchange ideas, problems, and solutions and discuss security guidelines for developers.

## 5.4 Recommendations & Future Work

Based on our findings, we provide recommendations to improve game development security and for future research areas. Our interviews suggest that security is not prioritized in game development due to severe obstacles such as lack of time, money, or security experts. Given this context, we provide recommendations for game development studios concerning their varying resources and unique circumstances.

**Introduce Security Early.** Our results suggest that, in many cases, game development does not factor in security during game design (cf. S. 4.2). As a result, we identified a lack of overview into which components can cause what security failures, and contingency or incident response plans are rarely developed, leading to higher impact and repercussions from security incidents [18–20]. Overall, previous work has shown that, especially in agile or rapid development environments, security is lacking, which in turn harms the security of the finished software [101–103]. Therefore, Attwood *et al.* have proposed a rapid prototyping approach merging vertical prototypes with early security considerations to help developers estimate security requirements and plan their development accordingly [104]. As game studios lack the resources to support games long-term after release, incidents may become even more costly in the long run [105], we recommend them to introduce security early.

**Game Design Documents.** Creating standardized game design documents at the beginning of development, e.g., in pre-production, can improve the overall development process by capturing drafts of planned game content and features [32, 106]. Further, Schmalz *et al.* suggest thorough pre-production can also help reduce overall budget and schedule risk by minimizing over-runs due to revisions [30]. This can be a starting point to introduce threat modeling to identify critical security-relevant components and consider measures to protect them [107, 108]. However, further research is needed to develop best practices for properly integrating security assessments into game design documents and the software life cycle overall, as factors such as production size, release platform, or game genre can influence security requirements.

**Dedicated People.** The most direct way to tackle security risks early and thoroughly is to either employ specialized security experts, foster security *champions* [109, 110], or equip key roles, such as producers of game productions, with the required security knowledge [30]. Having dedicated people for such tasks also has been one of the most prominent wishes of our participants. However, we recognize that while straightforward, this measure is costly compared to others and likely not feasible for many, especially smaller studios, especially as game developers are already required to have a diverse skill set. Additionally, future work needs to investigate the unique needs and practices of game security experts to spread

awareness and open a dialogue about security in game development (cf. S. 4.2).

**Testing & Reviews.** Our participants typically reported only functional testing by playing a game instead of testing and code reviewing to detect security issues (cf. S. 4.2). Yet, testing is the most critical and low-cost way to search for and mitigate security issues and automation through, e.g., continuous integration (CI) aligns with recommendations for environments with fast software development life cycles having tight deadlines [111]. Recent work on automated bug detection in games [112, 113] shows promise for identifying security issues while saving developer time, which could be valuable given the lack of code security understanding and resources we observed. Further, the use of large language models (LLMs) during code review processes may help with annotating potential security flaws for security-inexperienced developers [114, 115], although it must be emphasized that LLMs cannot reliably identify and analyze security vulnerabilities [116, 117].

**Game Engines, Extensions & Plugins.** We recommend equipping game engines and IDEs with automated support regarding security, such as linters for game projects [118]. Additionally, De Cremer *et al.* discussed the use of plugins or extensions for IDEs as beneficial for enforcing secure coding guidelines [119]. In preproduction, security policies could be established and automatically enforced in IDEs during production. However, future work needs to examine to which extent game engines or security plugins in IDEs can improve game security, as the former were described to currently abstract security too much.

**Ready-Made Tools.** Due to the limited security knowledge and the wish for quick and straightforward solutions, participants stated that ready-made solutions are favored, avoiding to "reinvent the wheel" (cf. S. 4.3.3). We generally agree with this assessment and recommend using pre-built tools wherever possible. Nevertheless, we must point out that non-open source solutions are a black box for developers and require trust and reliance on a third party regarding their security practices. In any case, developers should weigh whether such external solutions are necessary or if the desired results can be achieved with, e.g., inbuilt methods. This is especially true in the case of anti-cheat solutions that can require aggressive access rights.

**Dependency Checks.** Participants reported difficulties tracing code dependencies, especially since a lack of time hinders reviews and forces them to use the first working tool. However, supply chain incidents within popular games already occurred [120]. We recommend using software composition and dependency analysis tools as an automated and supporting source of information about known vulnerabilities in existing dependencies, saving valuable time [121, 122]. Adoption of such dependency management tooling can improve supply chain security but may require at least some security knowledge [123]. Beyond code dependencies, the handling of the multifaceted asset dependencies of games needs further exploration regarding security.

## 5.5 Putting Our Contributions Into Context

While previous work rarely discussed security or only examined specific aspects in game development, our work provides a novel human-centered overview of security in game development. We provide novel insights into the industry's awareness around security, and their key influencing characteristics, e.g., production size, game genre or release platform. Further, we describe the state of knowledge and neglect regarding security and how pressure points negatively affect security decisions.

Beyond, we confirm previous work, especially on complex testing and verification processes for security components [33, 69] and the fast and rather unstructured SE approaches in general discussed by Kasurinen *et al.* [124]. Pascarella *et al.* hinted at a problematic handling of code security [39], which we discussed in-depth in S. 4.3.4. We extend their findings with the widespread and critical misconception that stability-affecting bugs are not perceived as security issues. While Yan *et al.* found that client vulnerabilities are abused for cheating, we add to their findings by identifying a lack of awareness and care for code security, which leads to vulnerabilities within games [53].

Our work extends taxonomies from Ki *et al.* by adding first-hand experience, user privacy issues, and anti-cheat efforts in an overarching way. Additionally, we confirm the cheating taxonomy considerations of Yan *et al.* and the importance of a balanced approach between server authority and performance, similar to how one participant recommended a distinction of which areas of a game require server authority [53]. Moreover, we build upon the work of Bryant *et al.* on the general difficulty of implementing custom networking by adding insights of the lack of networking knowledge, leading to the industry's reliance on ready-made software instead [59].

While games further play an important role on AR/VR platforms [125], they were rarely mentioned in our interviews. However, AR/VR development faces similar privacy and user experience challenges based on the sensitive data collected by various device sensors. Moreover, authentication is usually handled differently and is often associated with large platform accounts [126], while our interviewees reported a diverse login landscape including social logins and custom or proprietary solutions (cf. S. 4.3.3).

Overall, while previous related work shed some light on the precarious working conditions, we explore them more in-depth, and report novel insights into how security is seen as a dispensable feature. Our study identifies a need for more collaboration and dissemination of security knowledge, which could be achieved by further research and a better result transfer to the industry.

# 6 Conclusion

We conducted semi-structured interviews with 20 game development experts to investigate awareness and priority areas regarding security, uncover processes and challenges, and finally draw up recommendations. Game security proved to be a complex landscape under intense pressure and heavy resource constraints. Therefore, security measures are often neglected and security flaws are knowingly accepted. However, especially in the area of user privacy, the industry shows particular awareness and high priority, even if primarily through laws such as the GDPR, which force the industry to act. Despite a rather low amount of knowledge about the topic, we found a relatively high awareness and great interest in game security, and possible solutions for the industry.

## References


[1] https://newzoo.com/resources/trend-reports/newzoo-global-games-market-report-2023-free-version, 2024.
[2] https://pwc.com/gx/en/industries/tmt/media/outlook/insights-and-perspectives.html, 2023.
[3] https://akamai.com/newsroom/press-release/state-of-the-internet-security-web-attacks-and-gaming-abuse, 2019.
[4] https://akamai.com/newsroom/press-release/akamai-research-shows-attacks-on-gaming-companies-more-than-doubled-over-past-year, 2022.
[5] L.-E. Dubois and J. Weststar, "Games-as-a-service: Conflicted identities on the new front-line of video game development," *New Media & Society*,
[6] L. Gu and A. L. Jia, "Player Activity and Popularity in Online Social Games and their Implications for Player Retention," in *16th Annual Workshop on Network and Systems Support for Games (NetGames)*, 2018.
[7] A. V. M. Moreira, V. V. Filho, and G. Ramalho, "Understanding mobile game success: A study of features related to acquisition, retention and monetization," 2014.
[8] https://forbes.com/sites/jasonwosborne/2023/05/25/how-loot-boxes-in-childrens-video-games-encourage-gambling/, 2023.
[9] https://bbc.co.uk/news/entertainment-arts-65855157, 2023.
[10] https://resources.irdeto.com/media/irdeto-global-gaming-survey-report, 2018.
[11] https://blog.irdeto.com/video-gaming/cheating-in-games-everything-you-always-wanted-to-know-about-it/, 2022.
[12] https://washingtonpost.com/video-games/2022/09/09/roblox-ads-metaverse/, 2022.
[13] https://theesa.com/2023-essential-facts/, 2023.
[14] https://flashpoint.io/blog/rce-vulnerability-dark-souls/, 2022.
[15] https://trendmicro.com/en_us/research/22/h/ransomware-actor-abuses-genshin-impact-anti-cheat-driver-to-kill-antivirus.html, 2022.
[16] https://bleepingcomputer.com/news/security/counter-strike-2-html-injection-bug-exposes-players-ip-addresses/, 2023.
[17] https://pcgamer.com/games/battle-royale/respawn-finally-comments-on-apex-cheating-scandal-that-saw-pro-players-games-hijacked-basically-says-nothing-at-all/, 2024.
[18] https://justice.gov/usao-sdca/pr/utah-man-sentenced-computer-hacking-crime, 2019.
[19] https://washingtonpost.com/video-games/2022/10/05/overwatch-2-servers-ddos/, 2022.
[20] https://en.wikipedia.org/wiki/2011_PlayStation_Network_outage, 2011.
[21] https://telegraph.co.uk/technology/sony/8475728/Millions-of-internet-users-hit-by-massive-Sony-PlayStation-data-theft.html, 2011.
[22] https://zdnet.com/article/ubisoft-confirms-just-dance-data-breach-amid-developer-exodus/, 2021.
[23] https://engadget.com/cd-projekt-red-says-it-was-hacked-but-wont-pay-any-ransom-090055291.html, 2021.
[24] https://vice.com/en/article/qjky8d/hackers-demand-dollar10m-from-riot-games-to-stop-leak-of-league-of-legends-source-code, 2023.
[25] https://theguardian.com/games/2022/sep/19/grand-theft-auto-6-leak-who-hacked-rockstar-and-what-was-stolen, 2022.
[26] https://polygon.com/24009631/insomniac-games-leak-hack-rhysida-files-breach, 2023.
[27] https://kotaku.com/crunch-time-why-game-developers-work-such-insane-hours-1704744577, 2016.
[28] https://time.com/5603329/e3-video-game-creators-union/, 2019.
[29] https://pewresearch.org/short-reads/2017/09/11/younger-men-play-video-games-but-so-do-a-diverse-group-of-other-americans/, 2017.
[30] M. Schmalz, A. Finn, and H. Taylor, "Risk Management in Video Game Development Projects," in *Proceedings of the 2014 47th Hawaii International Conference on System Sciences*, 2014.
[31] https://gamesindustry.biz/mastering-feature-creep-while-changing-the-world, 2017.
[32] C. M. Kanode and H. M. Haddad, "Software Engineering Challenges in Game Development," in *Sixth International Conference on Information Technology: New Generations*, 2009.
[33] E. Murphy-Hill, T. Zimmermann, and N. Nagappan, "Cowboys, Ankle Sprains, and Keepers of Quality: How is Video Game Development Different from Software Development?" In *Proceedings of the 36th International Conference on Software Engineering*, 2014.
[34] https://techspot.com/news/98046-diablo-4-beta-has-bricking-graphics-cards.html, 2023.
[35] https://secret.club/2021/04/20/source-engine-rce-invite.html, 2021.
[36] https://medium.com/@prizmant/hacking-punkbuster-e22e6cf2f36e, 2020.
[37] https://cyberint.com/blog/research/critical-account-takeover-vulnerability-discovered-and-patched-in-ea-games/, 2019.
[38] https://github.com/zeropwn/vulnerability-reports-and-pocs/blob/master/EA-Origin-RCE-CVE-2019-12828.md, 2019.
[39] L. Pascarella, F. Palomba, M. Di Penta, *et al.*, "How is Video Game Development Different from Software Development in Open Source?" In *Proceedings of the 15th International Conference on Mining Software Repositories*, 2018.
[40] J. Chueca, J. Verón, J. Font, *et al.*, "The consolidation of game software engineering: A systematic literature review of software engineering for industry-scale computer games," *Information and Software Technology*, 2024.
[41] M. Washburn, P. Sathiyanarayanan, M. Nagappan, *et al.*, "What went right and what went wrong: an analysis of 155 postmortems from game development," in *Proceedings of the 38th International Conference on Software Engineering Companion*, 2016.
[42] G. C. Ullmann, C. Politowski, Y.-G. Guéhéneuc, *et al.*, "Video Game Project Management Anti-Patterns," in *Proceedings of the 6th International ICSE Workshop on Games and Software Engineering: Engineering Fun, Inspiration, and Motivation*, 2022.
[43] C. Politowski, F. Petrillo, G. C. Ullmann, *et al.*, "Dataset of Video Game Development Problems," in *Proceedings of the 17th International Conference on Mining Software Repositories*, 2020.
[44] D. Callele, E. Neufeld, and K. Schneider, "Requirements engineering and the creative process in the video game industry," in *13th IEEE International Conference on Requirements Engineering (RE'05)*, 2005.
[45] F. Petrillo, M. Pimenta, F. Trindade, *et al.*, "What went wrong? A survey of problems in game development," *Comput. Entertain.*, 2009.
[46] https://forbes.com/sites/nelsongranados/2018/04/30/report-cheating-is-becoming-a-big-problem-in-online-gaming/, 2018.
[47] https://readwrite.com/how-the-multiplayer-gaming-industry-is-evolving-due-to-hacks-cheats/, 2023.
[48] V. Agrahari, S. Shanbhag, S. Chimalakonda, *et al.*, "A catalogue of game-specific anti-patterns based on GitHub and Game Development Stack Exchange," *J. Syst. Softw.*, 2023.
[49] J. Ki, J. Hee Cheon, J.-U. Kang, *et al.*, "Taxonomy of online game security," *Electron. Library*, 2004.
[50] S. Mohr and S. Rahman, "IT Security Issues Within the Video Game Industry," *International Journal of Computer Science & Information Technology*, 2011.
[51] M. Consalvo, *Cheating: Gaining Advantage in Videogames*. 2007.
[52] J. Yan, "Security design in online games," in *Proceedings of the 19th Annual Computer Security Applications Conference*, 2003.
[53] J. Yan and B. Randell, "A systematic classification of cheating in online games," in *Proceedings of 4th ACM SIGCOMM Workshop on Network and System Support for Games*, 2005.
[54] S. D. Webb and S. Soh, "Cheating in networked computer games: a review," in *Proceedings of the 2nd International Conference on Digital Interactive Media in Entertainment and Arts*, 2007.
[55] R. Parizi, A. Dehghantanha, K.-K. R. Choo, *et al.*, *Security in online games: Current implementations and challenges*. 2019.
[56] B. Bryant. and H. Saiedian., "A State Saturation Attack against Massively Multiplayer Online Videogames," in *Proceedings of the 7th International Conference on Information Systems Security and Privacy - ICISSP*, 2021.
[57] J. Woo and H. K. Kim, "Survey and research direction on online game security," in *Proceedings of the Workshop at SIGGRAPH Asia*, 2012.
[58] A. Yahyavi and B. Kemme, "Peer-to-peer architectures for massively multiplayer online games: A Survey," *ACM Comput. Surv.*, 2013.
[59] B. Bryant and H. Saiedian, "An evaluation of videogame network architecture performance and security," *Computer Networks*, 2021.
[60] Q. Zhang, "Improvement of Online Game Anti-Cheat System based on Deep Learning," in *2nd International Conference on Information Science and Education (ICISE-IE)*, 2021.
[61] M. Lukas, I. Tomicic, and A. Bernik, "Anticheat System Based on Reinforcement Learning Agents in Unity," *Information*, 2022.
[62] R. Spijkerman and E. Marie Ehlers, "Cheat Detection in a Multiplayer First-Person Shooter Using Artificial Intelligence Tools," in *Proceedings of the 3rd International Conference on Computational Intelligence and Intelligent Systems*, 2021.
[63] A. Maario, V. K. Shukla, A. Ambikapathy, *et al.*, "Redefining the Risks of Kernel-Level Anti-Cheat in Online Gaming," in *8th International Conference on Signal Processing and Integrated Networks (SPIN)*, 2021.
[64] C. Dorner and L. D. Klausner, "If It Looks Like a Rootkit and Deceives Like a Rootkit: A Critical Examination of Kernel-Level Anti-Cheat Systems," in *Proceedings of the 19th International Conference on Availability, Reliability and Security*, 2024.
[65] A. Yahyavi, J. Pang, and B. Kemme, "Towards providing security for mobile games," in *Proceedings of the Eighth ACM International Workshop on Mobility in the Evolving Internet Architecture*, 2013.
[66] E. Alothali, F. Belqasmi, O. Alfandi, *et al.*, "Identification and analysis of free games' permissions in Google Play," in *6th International Conference on Information and Communication Systems (ICICS)*, 2015.
[67] P. Laperdrix, N. Mehanna, A. Durey, *et al.*, "The Price to Play: A Privacy Analysis of Free and Paid Games in the Android Ecosystem," in *Proceedings of the ACM Web Conference*, 2022.

[68] R. Phaenthong and S. Ngamsuriyaroj, "Analyzing Security and Privacy Risks in Android Video Game Applications," in *Advanced Information Networking and Applications*, 2024.

[69] A. Ampatzoglou and I. Stamelos, "Software engineering research for computer games: A systematic review," *Information and Software Technology*, 2010.

[70] D. Turner, "Qualitative Interview Design: A Practical Guide for Novice Investigators," *Qualitative Report*, 2010.

[71] C. Wilson, "Interview Techniques for UX Practitioners," in 2014.

[72] D. Votipka, R. Stevens, E. Redmiles, *et al.*, "Hackers vs. Testers: A Comparison of Software Vulnerability Discovery Processes," in *2018 IEEE Symposium on Security and Privacy (SP)*, 2018.

[73] J. Mink, H. Kaur, J. Schmüser, *et al.*, ""Security is not my field, I'm a stats guy": a qualitative root cause analysis of barriers to adversarial machine learning defenses in industry," in *Proceedings of the 32nd USENIX Conference on Security Symposium*, 2023.

[74] S. Höltervennhoff, P. Klostermeyer, N. Wöhler, *et al.*, ""I wouldn't want my unsafe code to run my pacemaker": an interview study on the use, comprehension, and perceived risks of unsafe rust," in *Proceedings of the 32nd USENIX Conference on Security Symposium*, 2023.

[75] https://unity.com/, 2024.

[76] https://unrealengine.com/, 2024.

[77] https://slashdata.co/post/did-you-know-that-60-of-game-developers-use-game-engines, 2023.

[78] https : / / pocketgamer . biz / news / 82129 / 79 - of - devs - are - increasingly - pressured-to-release-unfinished-games/, 2023.

[79] https://pcmag.com/opinions/stop-giving-your-money-to-game-publishers-that-sell-now-patch-later, 2022.

[80] https://gamesindustry.biz/what-can-we-learn-from-newly-enacted-salary-transparency-laws, 2023.

[81] P. I. Fusch and L. R. Ness, "Are We There Yet? Data Saturation in Qualitative Research.," *Qualitative Report*, 2015.

[82] K. Charmaz, *Constructing Grounded Theory*. 2014.

[83] A. Strauss and J. M. Corbin, *Grounded theory in practice*. 1997.

[84] J. Corbin and A. Strauss, "Grounded theory research: Procedures, canons and evaluative criteria," *Qualitative Sociology*, 1990.

[85] C. Urquhart, *Grounded theory for qualitative research: A practical guide*. 2012.

[86] M. Birks and J. Mills, *Grounded theory: A practical guide*. 2015.

[87] N. McDonald, S. Schoenebeck, and A. Forte, "Reliability and Inter-Rater Reliability in Qualitative Research: Norms and Guidelines for CSCW and HCI Practice," *Proceedings of the ACM on Human-Computer Interaction*, 2019.

[88] A. Krause, J. H. Klemmer, N. Huaman, *et al.*, "Pushed by Accident: A Mixed-Methods Study on Strategies of Handling Secret Information in Source Code Repositories," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023.

[89] C. W. Munyendo, Y. Acar, and A. J. Aviv, ""In Eighty Percent of the Cases, I Select the Password for Them": Security and Privacy Challenges, Advice, and Opportunities at Cybercafes in Kenya," in *IEEE Symposium on Security and Privacy (SP)*, 2023.

[90] S. A. Horstmann, S. Domiks, M. Gutfleisch, *et al.*, ""Those things are written by lawyers, and programmers are reading that." Mapping the Communication Gap Between Software Developers and Privacy Experts," in *Proceedings on Privacy Enhancing Technologies*, 2024.

[91] S. Amft, S. Höltervennhoff, R. Panskus, *et al.*, "Everyone for Themselves? A Qualitative Study about Individual Security Setups of Open Source Software Contributors," in *2024 IEEE Symposium on Security and Privacy (SP)*, 2024.

[92] H. Beyer and K. Holtzblatt, *Contextual Design: Defining Customer-Centered Systems*. 1997.

[93] E. Kenneally and D. Dittrich, "The Menlo Report: Ethical Principles Guiding Information and Communication Technology Research," *SSRN Electronic Journal*, 2012.

[94] https://playfab.com/, 2024.

[95] T. Shabani, S. Jerie, and T. Shabani, "A comprehensive review of the Swiss cheese model in risk management," *Safety in Extreme Environments*, 2023.

[96] S. A. Horstmann, S. Domiks, M. Gutfleisch, *et al.*, ""Those things are written by lawyers, and programmers are reading that." Mapping the Communication Gap Between Software Developers and Privacy Experts," in *Proc. Priv. Enhancing Technol. 2024 (PoPETS)*, 2024.

[97] A. H. Mhaidli, Y. Zou, and F. Schaub, ""We Can't Live Without Them!" App Developers' Adoption of Ad Networks and Their Considerations of Consumer Risks," in *Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019)*, 2019.

[98] J. Haney, C. Cunningham, and S. M. Furman, "Towards Integrating Human-Centered Cybersecurity Research Into Practice: A Practitioner Survey," in *Symposium on Usable Security and Privacy (USEC)*, 2024.

[99] https://github.com/gmh5225/awesome-game-security, 2024.

[100] https://github.com/OWASP/www-project-game-security-framework/blob/master/migrated_content.md, 2024.

[101] B. Ramesh, L. Cao, and R. Baskerville, "Agile Requirements Engineering Practices and Challenges: An Empirical Study," *Information Systems Journal*, 2010.

[102] K. Rindell, J. Ruohonen, J. Holvitie, *et al.*, "Security in Agile Software Development: A Practitioner Survey," *Information and Software Technology*, 2021.

[103] W. Behutiye, P. Karhapää, L. López, *et al.*, "Management of Quality Requirements in Agile and Rapid Software Development: A Systematic Mapping Study," *Information and Software Technology*, 2020.

[104] S. Attwood, N. Onumah, K. Paxton-Fear, *et al.*, "Security-Focused Prototyping: A Natural Precursor to Secure Development," in *2022 13th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, 2022.

[105] B. Matthews, J. Sylvie, S.-H. Lee, *et al.*, "Addressing Security in Early Stages of Project Life Cycle," *Journal of Management in Engineering*, 2006.

[106] M. G. Salazar, H. A. Mitre, C. L. Olalde, *et al.*, "Proposal of Game Design Document from software engineering requirements perspective," in *17th International Conference on Computer Games (CGAMES)*, 2012.

[107] Z. Shi, K. Graffi, D. Starobinski, *et al.*, "Threat Modeling Tools: A Taxonomy," *IEEE Security & Privacy*, 2022.

[108] A. Shostack, *Threat modeling*. 2014.

[109] I. Ryan, U. Roedig, and K.-J. Stol, "Understanding Developer Security Archetypes," in *2021 IEEE/ACM 2nd International Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS)*, 2021.

[110] H. Aalvik, A. Nguyen-Duc, D. S. Cruzes, *et al.*, "Establishing a Security Champion in Agile Software Teams: A Systematic Literature Review," in *Advances in Information and Communication*, 2023.

[111] S. Chalishhafshejani, B. Pham, and M. Jaatun, "Automating Security in a Continuous Integration Pipeline," in *Proceedings of the 7th International Conference on Internet of Things, Big Data and Security (IoTBDS)*, 2022.

[112] B. Wilkins and K. Stathis, "World of Bugs: A Platform for Automated Bug Detection in 3D Video Games," in *2022 IEEE Conference on Games (CoG)*, 2022.

[113] E. Azizi and L. Zaman, "AstroBug: Automatic Game Bug Detection Using Deep Learning," *IEEE Transactions on Games*, 2024.

[114] Y. Almeida, D. Albuquerque, E. D. Filho, *et al.*, "AICodeReview: Advancing code quality with AI-enhanced reviews," *SoftwareX*, 2024.

[115] Z. Rasheed, M. A. Sami, M. Waseem, *et al.*, "AI-powered Code Review with LLMs: Early Results," *arXiv*, 2024.

[116] S. Ullah, M. Han, S. Pujar, *et al.*, "LLMs Cannot Reliably Identify and Reason About Security Vulnerabilities (Yet?): A Comprehensive Evaluation, Framework, and Benchmarks," in *2024 IEEE Symposium on Security and Privacy (SP)*, 2024.

[117] A. Khare, S. Dutta, Z. Li, *et al.*, "Understanding the Effectiveness of Large Language Models in Detecting Security Vulnerabilities," *arXiv*, 2024.

[118] A. Borrelli, V. Nardone, G. A. Di Lucca, *et al.*, "Detecting Video Game-Specific Bad Smells in Unity Projects," in *Proceedings of the 17th International Conference on Mining Software Repositories*, 2020.

[119] P. De Cremer, N. Desmet, M. Madou, *et al.*, "Sensei: Enforcing secure coding guidelines in the integrated development environment," *Software: Practice and Experience*, 2020.

[120] https://decoded.avast.io/janvojtesek/dota-2-under-attack-how-a-v8-bug-was-exploited-in-the-game/, 2023.

[121] https://owasp.org/www-project-dependency-check/, 2024.

[122] https://docs.github.com/en/code-security/dependabot.

[123] I. Pashchenko, D.-L. Vu, and F. Massacci, "A Qualitative Study of Dependency Management and Its Security Implications," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020.

[124] J. Kasurinen, M. Palacin-Silva, and E. Vanhala, "What Concerns Game Developers? A Study on Game Development Processes, Sustainability and Metrics," in *2017 IEEE/ACM 8th Workshop on Emerging Trends in Software Metrics (WETSoM)*, 2017.

[125] T. Kari and M. Kosa, "Acceptance and use of virtual reality games: an extension of HMSAM," *Virtual Reality*, 2023.

[126] S. Stephenson, B. Pal, S. Fan, *et al.*, "SoK: Authentication in Augmented and Virtual Reality," in *2022 IEEE Symposium on Security and Privacy (SP)*, 2022.


# A Mapping Quantifier & Percent

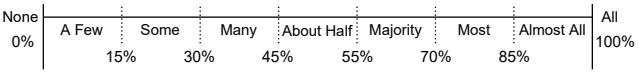


**Figure 2: Mapping between our utilized quantifiers within Section 4 and the respective participant percentage ranges.**