Understanding Hackers' Work: An Empirical Study of Offensive Security Practitioners

Andreas Happe
andreas.happe@tuwien.ac.at
TU Wien
Vienna, Austria

Jürgen Cito
juergen.cito@tuwien.ac.at
TU Wien
Vienna, Austria

ABSTRACT

Offensive security-tests are commonly employed to pro-actively discover potential vulnerabilities. They are performed by specialists, also known as penetration-testers or white-hat hackers. The chronic lack of available white-hat hackers prevents sufficient security test coverage of software. Research into automation tries to alleviate this problem by improving the efficiency of security testing. To achieve this, researchers and tool builders need a solid understanding of how hackers work, their assumptions, and pain points.

In this paper, we present a first data-driven exploratory qualitative study of twelve security professionals, their work and problems occurring therein. We perform a thematic analysis to gain insights into the execution of security assignments, hackers' thought processes and encountered challenges. This analysis allows us to conclude with recommendations for researchers and tool builders, to increase the efficiency of their automation and identify novel areas for research.

CCS CONCEPTS

• **Security and privacy → Usability in security and privacy**.

KEYWORDS

software testing, offensive security testing, ethical hacking

ACM Reference Format:
Andreas Happe and Jürgen Cito. 2023. Understanding Hackers' Work: An Empirical Study of Offensive Security Practitioners. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '23), December 3–9, 2023, San Francisco, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3611643.3613900

1 INTRODUCTION

For convenience and efficiency reasons, more and more devices are being connected and thus exposed to public networks. While beneficial, this has a dark undercurrent: the respective system's attack surface is increased and could be exploited by malicious actors. In a perfect world, all created software would be free from faults. As recent [19, 20], and not so recent [29], news implies, we are sadly not there yet. While secure software development, enabled by defensive security testing [49, 57, 63], is the long-term goal, short-term interventions are needed. In addition, there is an ever-increasing abundance of legacy software whose security needs to be verified too. A pragmatic approach is to perform security assessments, also known as penetration tests (pen-tests), to identify vulnerabilities and remediate them before they are discovered and exploited by malicious actors.

This approach is limited by the availability of skilled offensive security professionals [41, 46]. While this situation should be remediated through increased enrollment in IT security educational programs, improving the efficiency of the penetration testers through tooling is an equally important measure. To accomplish this, research and tooling should be well-aligned with security professionals' activities and needs.

However, to the best of our knowledge, there has been no empirical research into what type of security assessments are performed, what actions are regularly performed within those, or how professionals select attacks to be run against their targets. Without this, developments might be swift but misguided, and thus eventually irrelevant.

Research Questions & Structure of this Work. We used three research questions to drive the development of this work; the applied research method is described in the Methodology section.

Our first research question was **"What do common security tests look like?"** We present the gathered information in section Performing Security Tests, detailing different types of assignments, their particularities, common actions performed during assignments, and the role of automation.

The second research question **"How do Hackers perform their tasks?"** focused on the inner world of our participants. Education is an important part of socialization, therefore, results about this aspect is included in section Becoming A Hacker. In Section How do Hackers think? we present recurring themes detected during our analysis. We focus on thought processes during assignment execution, target and attack selection, dealing with uncertainty, and internal quality assurance.

The Discussions and Implications Section is the response to the final **"What tedious or time-consuming areas could be improved?"** question. We grouped the identified research and development opportunities according to our target audience of researchers and tool builders.

2 RELATED WORK

While there has been ample research on secure software development and defensive security testing [49, 57, 63], the focus of our study is offensive security testing. To the best of our knowledge,

ESEC/FSE '23, December 3–9, 2023, San Francisco, CA, USA

ACM ISBN 979-8-4007-0327-0/23/12.
https://doi.org/10.1145/3611643.3613900

this is the first work that focuses on how hackers work, i.e., the context within which a security professional moves and the processes that influence their decisions during security assignments.

Huaman et al. [40] performed a large-scale interview study of German small-to-medium enterprises (SMEs). While SMEs are making up a third of Germany's GDP, they often lack resources for establishing an effective cyber-security posture. It analyzes their preconception with regards to cybercrime, their adoption of security measures and their experiences with attacks. In contrast to our study, this focuses upon the potential "victims", not upon security operators. One interesting finding was that 45.1% of interviewed companies had a cybersecurity incident warranting manual response in the preceding 12 months — further highlighting the need for trained personnel.

Smith, Theisen and Barik [54] describe Red Teams working at Microsoft. They cover a wide range of topics including how corporate culture and red teaming interact. They also lightly touched on how people became security professionals and the interactions in their daily work. Its interviewees were recruited from within Microsoft, a single large-scale company, and thus might not reflect wider industry practices which, as referenced by the previously mentioned paper, consists to a large degree of SMEs. In contrast, this publication focuses on the execution of security assignments, highlights hacker's thought processes and details challenges in academic and automation research. Furthermore, this paper is not limited to the discipline of red-teaming.

Van den Hout [60] investigated the impact of different penetration test methodologies on the quality of the tests performed but concluded that only one reviewed methodology had widespread adoption, but its recommendations for a structured approach were not taken into account. This could indicate a gap between "real" penetration testing and codified methodologies.

Multiple papers describe aspects of penetration-testing without focusing on the operator's mindset or their decision processes. Munaiah et al. [48] analyze event datasets and manually map attack patterns to *MITRE ATT&CK Enterprise*. This is used to show a-posteriori attack patterns but does not analyze how hackers select the attacks to execute. *MITRE ATT&CK* itself is a taxonomy of TTPs (Tactics, Techniques, and Procedures) and not a full attack methodology. Bhuiyan et al. [23] uses GitHub security bug reports to identify the origins of bug reports. Examples of these origins are software source code, software log files, binary files, etc. This details what data are used during reporting, but does not explain how a security professional identifies potential vulnerabilities for research in the first place, e.g., why a security professional analyzes a mentioned log file for relevant security information.

Other papers focus on narrow sub-disciplines of hacking which cannot be projected upon the hacking industry at large. Ceccato et al. [28] describes how hackers attack protected software, i.e., how software protection mechanisms in provided binary files are analyzed through reverse engineering. Based upon the responses of our interview series, reverse-engineering is not representative for activities performed by offensive operators at large.[1]

[1]During the interview series, a single participant mentioned using fuzzing to hunt for vulnerabilities. They were switching to other disciplines due to the high resource and time requirements of fuzzing.

Table 1: Participants

Participant	Primary	Secondary
Participant 1	web	infrastructure, iso27001
Participant 2	web	infrastructure, mobile
Participant 3	red-team	AD, OT, web
Participant 4	web	social engineering
Participant 5	red-team, IoT/OT	web, social engineering
Participant 6	web	AD, social engineering
Participant 7	infrastructure	web, tool development
Participant 8	web	infrastructure
Participant 9	infrastructure	AD
Participant 10	red-team, AD	
Participant 11	OT, IoT	web
Participant 12	web	

AD denotes Internal network tests; *web*, *infrastructure* and *IoT* denote pen-tests.

The PhD thesis "How Hackers Think" [56] is a high-level treatise on hacker history, culture and their thought processes. It identifies multiple characteristics of hackers, e.g., being highly self-motivated and curious, being able to tolerate ambiguity, and their use of mental models and patterning. Its focus lies on a high conceptual level and does not analyze how hackers actually identify and chose vulnerabilities. Neither does the study identify how different areas of penetration-testing, e.g., OT or red-teaming, might impact a hacker's mindset.

3 METHODOLOGY

This paper follows a *pragmatist* approach [47, 51] combining methods from the *empiricist* and *summarist interpretist* traditions [33].

We used semi-structured interviews to gather insights into hackers' work and thought processes.

Ethical Considerations. Our institution does not have a formal IRB process but offers voluntary submission to a Pilot Research Ethics Committee. As human interviews were conducted, the committee was consulted, and topics were discussed, including ethically relevant methodological clarifications, more specifically questions related to the involvement of voluntary participants in the research, as well as mitigating the risk of contextual identification. Participants gave their informed consent before the interviews took place; all data collected were anonymized by researchers prior to analysis. All data storage and processing complied with national privacy regulations and the EU's General Data Protection Regulation (GDPR).

Recruitment. We define the target population as offensive-security practitioners that work directly with customer systems. Previous research has found that security professionals are reluctant to communicate with outsiders [45], especially when it comes to their methodology and techniques. To counteract this, researchers reached out to public figures: the initial seed was populated by contacting security companies, finalists of public security challenges, and security conference participants. We use snowball sampling to improve the interview pool: At the end of each interview, we asked the current interviewee to connect us with other offensive security professionals. In addition, we cold-called both a hacking education

YouTuber and a public hacking collective that is well known for publishing vulnerability disclosures. Both were mentioned by the participants during the interviews, both did not react to the contact attempt further enforcing the idea of a close-knit community [45].

We sampled new interview participants until theoretical saturation was reached, that is, no new information was obtained during the interviews. When considering theoretical saturation we differentiated between common themes and themes specific to the interviewee's specialty area. We continued interviews until neither two subsequent interviews contributed new specialty area information, nor three subsequent interviews contributed new common themes. Theoretical saturation was reached after the 12th interview which fit recommendations [31, 34].

Participants. We considered participants that worked primarily in an offensive security field and excluded participants that primarily worked within social engineering or physical security. If participants were working in a hybrid field, such as reverse-engineering or source-code analysis, their primary focus had to be offensive. To gain seasoned results, we only reached out to professionals with at least four years of experience in the IT security field.

To our dismay, we were not able to recruit any offensive security professionals that identified as non-male. While we come from a culture that naively prides itself on blind meritocracy [24], we found this contradiction disturbing. As we did not deem it relevant, we did not ask about our participants' religious or cultural affectivities, but in hindsight, we can assume diversity in that area.

To protect the anonymity of the participants, we cannot detail their employment status, ethnicity, work experience before security work, and time of employment within the security field, etc. When excluding education and CTF-participation, participants had an average work experience of 9 years ($\mu = 9.0, \sigma = 6.5, median = 8$).

Interview Protocol. We conducted semi-structured interviews utilizing video conferencing software. All but two interviewees enabled both video and audio transmission. The average duration of the interview was 55 minutes. Before the interview started, the participants were informed about data processing, and their rights, and asked for their informed consent.

We opened the interviews with questions about the interviewee's job description and how they acquired the needed skill set. Those were followed up by talking about the types of security assignments the participants are involved with. For 1–3 of these areas, detailed questions about particularities, procedures, automation, and problems were asked; since the questions were open-ended, the interviews branched out to subtopics organically. The interviews were closed with questions about grievances and additional thoughts related to the field of IT security.

We recorded and manually transcribed all interviews. During the transcription, sensitive data was scrubbed from the interview; the transcribed interview was then submitted for confirmation to the interviewee. Scrubbed interviews were loaded into *delve* [5] for thematic analysis.

Analysis. *Reflexive Thematic Analysis* [26] was chosen to perform a data-driven exploratory analysis of interview transcriptions. In summary, when performing thematic analysis, the researchers initially familiarize themselves with the data, and extracts of the data are tagged with codes. These codes are then used to create clusters that identify or construct underlying themes. Then, those themes are reviewed, defined, and named. The results of the findings are presented in Section 4–6.

Data Availability. The data used in this study was collected through interviews with a close-knit community of ethical hackers. Deanonymization would likely not be preventable. In accordance with ethical guidelines and agreement with the interview participants, the decision was made not to release the interview data. All meta-information related to the interviews, including the interview guide and consent forms are part of our replication package.

Threats to Validity. Any interview-based study faces the threat of *selection bias* (*internal threat*). To counteract this, we performed snowball sampling, recruited random security professionals during security conferences, and explicitly invited security professionals from different disciplines. For ethical reasons, interview participation was limited to white-hat hackers (*internal threat*). According to prior analysis, the activities of black-hat hackers, e.g., Ransomware groups, can be seen as a subset of the activities performed by ethical red-teams [3, 4] which are covered in this work.

Another potential bias would be *experimenter bias* (*internal threat*). To reduce the risk, all the data collected was analyzed separately by the different authors, and their respective labeling results were compared for differences, ambiguities were discussed and resolved.

Hacking contains multiple disciplines. Our results might only capture common themes of a subset of those (*external validity*). We try to counteract this by inviting interviewees from various hacking fields, as is reflected in Table 1. The geographical distribution covered roughly Central Europe. Other geographic regions might be more advanced when it comes to the utilization of the different types of security assignment.

4 BECOMING A HACKER

The interview responses reveal several interesting themes regarding the path to becoming a hacker.

Academic Education. All but one participant attended at least a single university-level class. Nine completed bachelor's degree studies in IT (or related field, such as CS), and of those, all continued to add a master's level degree. The percentage of interviewees enrolled in IT security specific programs increased from 55% ($n = 5$) for bachelor's studies to 78% ($n = 7$) for master's studies. This fits the perceived lack of IT-Security and Secure Development lectures during non IT-security centric programs, which was partially addressed by attending CTFs or enrolling for non-mandatory security classes. Classes were often taken in an extra occupational capacity. All fitting a common theme of "*fascination with IT security*" combined with high intrinsic motivation.

Experience before IT-Security. Having 2–3 years of non-security IT exposure before entering the IT security field was found to be advantageous. Another related recommendation was to have a broad IT security base combined with one or two specialization areas. Within our group of interviewees, the common base was web security or internal network assessments; examples of specializations were red teaming or cloud-specific knowledge.

Staying relevant. All interviewees perceived a need for ongoing education. The ubiquitous information source was Twitter/X, followed by other online services such as YouTube channels, blog posts, Reddit, Github, or commercial online courses. In the physical

Table 2: Types of Security Assessments

Type	Covert	Team-Size	Effort in Days
Vulnerability Assessment	not typical	1	2-4
Penetration Test	optional	1-2	5-10
Internal Network Test	optional	1-2	7-10
OT Test	never	1-2	7-10
Red-Teaming	always	3-4	30+

world, colleagues and conferences were mentioned. The quality of online material was considered high, although one interviewee had qualms about publishing information due to potential misuse. A single participant regularly used the Darknet as a news source.

To CTF or not. CTF attendance was a common theme. Participants saw a bidirectional information transfer: skills learned in CTFs were applicable at work and vice versa. Tasks in CTFs were considered very targeted in that they narrowly focus on a vulnerability, and solving the challenge or reading a write-up were considered efficient ways of gathering knowledge about the respective vulnerability. Specialized security practitioners, e.g., from the OT or ICS area, found CTFs to be introductory and shallow.

5 HOW DO HACKERS WORK?

While we encountered the common muttering of "*every projects is different*", these sections identify types of penetration tests, each with distinct requirements, strategies, and particular actions. When looking at a pen-tester's work, this is the external view, i.e., how a pen-tester's work is perceived from the outside.

5.1 Types of Security Tests

Although different assignments have a similar project organization, their execution differs due to the respective client and target environment. Table 2 shows the main types of security assignments encountered during interviews.

Vulnerability Assessments focus upon achieving a high coverage of the targeted assets, which are typically external IP-ranges (including web servers) or internal networks (including clients and internal infrastructure). Enumerating targets, e.g., through web crawling or network scans, leads to the creation of important inventory databases. Those are subsequently used to test against known vulnerability databases, known configuration errors or generic vulnerability classes such as SQL injections. As assignments typically include large amounts of potential targets, a high level of automation is necessary.

Penetration Tests (Pen-Tests) share similarities with vulnerability assessments. The demarcation point between those two varied between interviewees. The situation is further complicated as vulnerability scans are often used as an initial step during pentesting. Generally speaking, while vulnerability assessments focus on breadth, pen-testing focuses on depth, i.e., thoroughly breaking a single target. Pen-Tests are within the realm of application security: in addition to well-known vulnerabilities or configuration errors, new vulnerabilities are hunted within the software under test. Penetration tests are often performed against custom-written software where no prior vulnerabilities are published in vulnerability databases. As the scope is tight, customers commonly provide dedicated test environments against which destructive tests can be performed. Another benefit of the limited scope is that the execution of a penetration test can be highly structured, some ($n = 2$) interviewees went as far as calling them "*catalog-based*". Pen-tests are primarily performed manually.

Internal Network Tests verify the security and resilience of internal networks. Their basic assumption is "*assumed breach*", i.e., the adversary is already within the local network and now attempts to gain sensitive data or achieve higher privileges — emulating Ransomware scenarios that have recently scourged companies. Microsoft Active Directory (AD) is ubiquitous in corporate networks; thus, if present, it is the main target. In these cases, the security assignment's intent is to obtain domain administrator privileges. The focus lies on exploiting known vulnerabilities, product features, mis-configurations, and insufficient access-control or hardening measures. Another big aspect is Lateral Movement, i.e., using compromised systems to pivot to new targets. Assignments are made against productive environments.

OT Tests target Operational Technology (OT) such as SCADA or ICS (Industrial Control System) networks. They can be differentiated into product tests and in-situ network tests of already configured systems. As solutions consist of off-the-shelf software that is highly customized for usage within the corresponding client network, the latter are often preferred by the customer. Tested subjects often use proprietary protocols; therefore, reverse engineering is a common practice in OT tests.

OT facilities, e.g., power plants, are expensive and often hard to come by, thus a dedicated testing environment is rarely available. Testing commonly occurs during scheduled down-times; this severely impacts the available test window. Another related particularity: availability often trumps the breadth or depth of performed security tests. As test subjects are "*connected to the real world*", negative side effects are potentially catastrophic. Security tests are therefore highly coordinated with customers to prevent any negative fallout. This often prohibits any covert action. Regulatory requirements [21] lead to a convergence between IoT and OT devices. In addition, Microsoft Active Directory starts to encreep OT networks, thus creating an overlap with Internal Network Tests.

Compared to other approaches, in **Red-Teaming** the attackers have a concrete mission, e.g., gain access to a defined subset of computers or a source code repository. While during *Internal Network Penetration Tests* gaining Domain Admin is often the final goal, this is only a means for achieving the mission during Red-Teaming. Attackers holistically target a company and employ additional techniques such as Open Source Intelligence (OSINT) and Social Engineering; Post-Exploitation is more prominent compared to other disciplines. Red teaming is not concerned with broad coverage, but with achieving the team's defined objective. Red-Teaming does not only attack the target's technical security posture but also the response of the blue team, i.e., defenders. Thus covert operations, hidden persistence, command&control systems (C2) and evasion of defensive techniques enter the picture.

Assignments are often performed in larger teams and over extensive time frames, making information transfer between participants more important. Adding additional team members to speed up an

ongoing operation is problematic as the new team members do not share the existing member's target system knowledge.

5.2 Black- vs. Gray-box Security Testing

When it comes to test execution, an important distinction is the amount of information and support provided by the customer. During black-box tests, practitioners go in "*blind*"; no information except the scope is given. During white-box tests, full system access or even the source-code of the tested application is given. Gray-box tests lie in-between: often access credentials or system architecture descriptions are provided before testing commences.

Pure white-box tests, as in "source-code reviews", are rarely performed due to their prohibitive costs. The type of assignment is also of importance: red-teaming is almost always performed as a black-box test as the target's personnel is not involved beneficially. OT tests are often performed in tight lock-step with customers (to reduce the potential fallout) and thus are gray-boxed. Interviewees overwhelmingly **recommended moving from black-box towards white-box testing**. The reasons given were time and thus cost efficiency, as well as potential for improved test coverage.

In other areas, customers are helping pen testers to improve efficiency too. "Assumed breach" scenarios in Internal Network Penetration Testing conceptually assume that a client computer will be breached eventually and thus use a breached computer as a starting point for investigations. During web pen tests or during external scans, rate limits or firewalls are commonly disabled to allow swift pen test execution. During web application pen-tests, internal details, such as used technologies, are commonly provided to reduce the search space.

5.3 Typical Testing Workflows

Participants were asked to detail the execution of the different types of assignments. This section describes the peculiarities of the different areas.

Activities performed during **Web Penetration Tests** can be separated into exploratory intuitive testing and exhaustive testing against checklists or standards. All interviewees utilized both, no specific ordering between those two was detected, although if the checklist-verification was automated, it often was run in parallel to exploratory testing. If a high-level of automation is achieved, the manual exploratory testing can be integrated into the automation: one interviewee detailed a multi-stage automated test-setup containing multiple enumeration steps, where the result of each step was manually verified, rectified and used to instrument subsequent automated steps. Manual testing, e.g., manual crawling, was integrated as an additional input into the tested steps.

According to interviewees, most time and effort are spent upon authorization tests. An application typically has multiple user groups with different access rights. During testing, penetration testers request one or more users per existing group and try to perform unauthorized data access with one user using data of another user. To verify responses, testers need documentation about the implemented access groups. If none was given, interviewees approximate a model of the access rules through probing/testing and experience.

With the exception of testing for authentication or authorization, automated testing was deemed well-established and automated tooling was commonly employed. Common injection attack vectors were well covered by tooling, for example, *sqlmap* [18] for testing for SQL injections. Multiple Web-Application-Testers "complained" that typical injection-based attacks which were common 10 years ago are now seldom seen and are rather used for illustrative purposes during education. Their suspected "culprit" is the rise of web application frameworks with sane defaults that automatically prevent many attack classes. Multiple interviewees considered switching their area of interest due to this development.

Multiple interviewees described API-based tests as tedious. Typically an API test is performed by calling a sequence of operations. Each operation is detailed through an API specification provided by the customer, e.g., through OpenAPI/Swagger or WSDL files. In theory, directly testing the back-end API reduces the pen-testing overhead as the tester can focus upon the core functionality; in practice, API tests become time-consuming due to a lack of documentation with sufficient quality. API documentation only describes single operations, often lacking detailed descriptions of valid input formats and their semantics. In addition, to achieve good test coverage, test cases need to perform a sequence of causally dependent API calls, potentially reusing and refining data between operations. While performing a traditional web application pen test, this causality and examples of input data can be derived from the captured web traffic. When performing API tests, these have to be derived from the API specifications or, more realistically, by pestering the customer's liaison contact.

Internal Network Tests often occur in phases which are ordered from "*quiet*" to "*loud*" when it comes to visibility. A typical assignment targeting a Microsoft Active Directory might include the following phases: initially, only network access is granted. The attacker either sniffs the network for exposed access credentials or utilizes MitM- and spoofing attacks to gain user credentials or tokens. In addition, anonymously accessible network shares are investigated for "*juicy*" information such as user or admin credentials. Exploits are used against vulnerable network services if the risks of detection and stability are deemed acceptable. In the second phase, an attacker has either already gathered user credentials or has been provided with those by the customer. These credentials are typically for non-privileged domain users, and attackers utilize them to further enumerate shares, gain access to additional domain accounts or computers, or gain local administrative privileges. Lateral Movement often incurs during this phase. In the next phase, the attacker has either gained or is provided local administrative privileges and tries to perform further Lateral Movement until a domain administrative account is compromised. With that, the whole network is owned.

Please note, that phases do not follow a traditional waterfall model. According to interviewees ($n = 2$), often the domain admin credentials can be gathered during the initial phase. This is then noted, and additional attacks are performed until the agreed upon timebox is reached.

Many automated attacks, e.g., EternalBlue [25] or certify [7], were described as "*too loud*" or "*unstable*" for use during the initial phases. Another automation topic was the identification of "*juicy*" files within network shares: this activity is performed primarily manually as the identified data are context specific. In addition, creating a full-copy of a network share is time- and network-sensitive

Table 3: Commonly Named Tools.

Tool	Area	Availability	#
PortSwigger BURP Suite [12]	Web-Testing	free, commercial	7
BloodHound [2]	AD Enumeration	OSS	5
SQLMap.py [18]	Web/SQLi	OSS	3
nmap [14]	Network	OSS	7
nessus [13]	Network	commercial	8
gobuster [8], dirbuster [6]	Network	OSS	2
certify [7]	AD Exploitation	OSS	4
metasploit [11]	Exploitation	OSS	3
nuclei [15]	Exploitation	OSS	3

# denotes the interviewee count mentioning the corresponding tool.

as well as easily detectable, and countermeasure systems using honey-tokens are beginning to be deployed at customers' sites.

All interviewees in the IoT area mentioned applying industrial standards as well as the usage of checklists that included the OWASP IoT [36] and OWASP Firmware Testing guides [35].

Red-Teaming is special due to its evasion- and deception-based methods as well as through its objective-based approach. A red team initially has knowledge of its objective, e.g., gain access to a special server in department X, as well as a broad allowed scope, e.g., the targeted company. Teams initially model how to breach the company, e.g., by identifying potential social engineering victims. After the breach, low-key enumeration is used to covertly model "*how a company works*" and then abuse that knowledge to derive attacks that mirror expected traffic and behavior patterns. Throughout a red-teaming campaign, a map of known or breached elements is built and compared to the imagined map of the company that includes the final objective: if both converge, the objective should be achieved.

Automation employed for network lateral movement or breaching web applications originate from the other pen-testing disciplines but have to be re-evaluated against their chance of being detected. As red-team assignments are performed against real and live systems, the scope of destructive operations might be limited.

OT-Tests have their own challenges. Due to the prevalence of proprietary protocols, time-consuming reverse engineering of those protocols often occurs. Mentioned experiences of our interviewees indicate that Security-by-Obscurity is still common; this would match the perceived resistance of some ICS suppliers when faced with responsible disclosure requests. Due to the time burden of reverse-engineering, it frequently has to be aborted due to the timeboxed nature of testing.

Due to the potentially catastrophic side-effects of testing, a risk-based approach is often applied: together with the customer, a threat model workshop can be performed, and potential scenarios that warrant testing identified. Those scenarios, and only those, are subsequently manually executed against the OT system. As the available amount of time is fixed, threat modeling and performing the derived tests compete for the same temporal resources.

5.4 Automation

All interviewees used pre-made tooling, while few ($n = 3$) wrote additional tooling on their own. Overall, the tooling situation for specific testing areas was seen in a positive light. In contrast, "*all-in-one*" tools were seen in a negative light. Multiple interviewees remarked that a "*fully automated tool cannot replace a pen-tester*" or, as one interviewee cynically replied, "*yeah, I want a tool where I can click a button and magically I get a finished pen-test report*". Practitioners relied on multiple small tools for different areas, e.g., *gobuster* [8] for content discovery or *sqlmap* [18] for testing SQL injections. PortSwigger's *BURP Proxy Suite* [12] was used by every web application pen-tester interviewed. See Table 3 for a list of commonly named automated tools.

Problems with tooling. Interviewees remarked that the setup overhead of automation tools can be problematic. Especially for short-term projects, such as vulnerability assessments or tightly-timed web application pen-tests, the initial setup overhead and processing time can be prohibitive for deploying tooling. Another problem was coverage: even within the same problem area, the coverage of different tools widely diverges, and the situation is made worse as commonly no tool provides full coverage of a testing area. To counteract this, practitioners commonly use multiple tools redundantly, yielding more processing time overhead and needing manual merging of the different tools' results.

Some areas were described as not suitable for automation. As OT systems are finicky and the potential fallout catastrophic, automated tests are often not feasible. Additionally, when performing social engineering during red-team assignments, fully automated tools are avoided for both fear of detection and ethical qualms because they would be used on human targets.

Extendability and Community was identified as an important discriminator by practitioners. Both are related to fast-paced developments within the exploit community: if a tool can be proactively extended or be scripted by the community, it and its implemented methods can evolve faster compared to reactive development within walled gardens. An example of an OSS tool utilizing community-provided detection rules is *nuclei* [15]; an example of a commercial tool with good OSS extendability is the PortSwigger *BURP Proxy Suite* [12] with its integrated *BApp Store*.

Manual fine-tuning to reduce search space. Multiple interviewees mentioned that they are adjusting the tooling according to their ongoing findings. Examples of this feedback loop would be limiting tested vulnerability classes to feasible ones, e.g., not testing a static website for SQL injections, or limiting tested database queries to concrete database dialects.

6 HOW DO HACKERS THINK?

While Section 5 describes the external view on pen-tests, their type and activities activities performed during them, this section focuses on the inner workings and thoughts of security professionals during testing, detailing their decision processes and potential sources of their intrinsic motivation.

6.1 Exploiting Configuration vs. Applications

A reoccurring theme was the distinction between searching for known vulnerabilities and hunting for new vulnerabilities.

Examples of the former would be executing a known vulnerability scan against off-the-shelf software or investigating a Microsoft Active Directory for misconfigurations; an example of the latter

Table 4: Excerpt of sub-themes of "Identifying Vulnerable Areas or Operations"

Subtheme	#	Representative Quotes
High-Level Targeting	7	*"We select the attack that would be the most cost-effective for the attacker"* *"…before we attack proprietary protocols we'll attack a windows domain server missing updates."*
Experience	12	*"How we actually work? We look for obvious vulnerabilities, those that jump out immediately"* *"I know that from my time programming C/C++…I find the errors that I made back then"* *"I search for vulnerabilities that I have seen and exploited before."* *"…often I see systems that I have already seen when doing CTFs…then I already know how to attack it"*
Familiarity with Target	4	*"If it is a repeat customer then you already know how they tick and what their problems are"*
Observed Features	10	*"One runs through the web applications and sees a feature and thinks "this looks interesting, could it be implemented weirdly"?"* *"If there's an upload function, I am interested."*
Observed Technology	11	*"Some things cannot be done securely, for example PHP."* *"Well, you always feel happy when the application is somehow a PHP application."*
Modeling Behavior	9	*"Testing is manual, as you need to get a feel how the application is supposed to work and answer"* *"You search for unexpected behavior…for example a database that throws an error when you enter a '. "*
Intuition	8	*"This will be esoteric…but I believe there is some organ that tingles if an operation looks fishy"*

Subthemes mentioned by interviewees, # denotes the interviewee count.

would be searching for SQL injections within a custom written application or discovering a new vulnerability class.

Synonyms given for "*searching for known vulnerabilities vs. hunting for new vulnerabilities*" were "*vulnerability assessments vs. application security*" or "*hacking configuration vs. hacking programs*".

These two categories are fluid. For example, findings from "*hunting for bugs*", i.e., a new 0-day exploit against a software, can end up within "*searching for known vulnerabilities*", i.e., when a rule for detecting 0-day is added to a web vulnerability scanner.

While not stated explicitly during the interviews, we assume that our interviewee's mental model is primed through their understanding of this divide, and highly impacts tool and technique selection. As an interviewee mentioned, "*you don't hunt for 0-days during an Active Directory assignment*". This implies that pen-testers will not consider spending days fuzzing a domain controller for new vulnerabilities during internal network scans.

6.2 Identifying Vulnerable Areas or Operations

Participants often described exploratory testing during which they were guided by intuition. Through follow-up questions, further information about this intuition was gathered.

All interviewees were analyzing requests and responses; the former for conspicuous parameters and the latter for occurrences of error messages or other suspicious behavior, that is, behavior that does not fulfill the testers' expectations.

During the interviews, multiple areas were identified where security testers possessed a mental model of the expected behavior of the software-under-test; during testing security testers were trying to find operations that could trigger unexpected behavior which, in turn, might turn into a security vulnerability. Those mental models were built from experience, e.g., prior assignments or experience within the specific business area, as well as adapted during the security test itself, e.g., *"learning how the application works"*. A summary of multiple observed mental models is shown in Table 5.

Pen-testers attributed their intuition to experience which could be built from previous penetration tests, participation in CTF events, prior engagements with the same client or industry area, or by implementing similar software solutions during their former life as software developers. Participants remarked that during testing, they are triggered by vulnerabilities or exploits they had recently read about and, in response, would start additional research. One penetration tester mentioned creating a topic map during everyday research which they then refer back to during assignments.

Related to experience, practitioners had preconceptions about the technologies used or features implemented. Some functionality, e.g., file uploads or XML processing, were thought to be hard to implement in a secure manner — to quote a participant, "*there are some things that just cannot be implemented correctly*". Similar resentments were discovered about used technologies. Some programming languages were deemed to increase the probability of an application containing defects; an interviewee mentioned thinking "*let's see how developers have been fooled again*" when going into assignments. As cynical as it may be, PHP was often mentioned as such a technology.

Two distinct positions were experienced regarding the learnability of this intuition. On one side, "*nobody is born a super hacker*", on the other hand, one interviewee mentioned that the best penetration testers in their peer group exhibited hacking-style behavior already during kindergarten. Debating nature-vs-nurture or art-vs-craft would go beyond the scope of this publication. Regardless of this, common consensus was found that hacking skills are improved through practice.

It is important to note that participants may be subject to *selection* and *survivorship* bias. They might find vulnerabilities in areas they focus on, ignoring plentiful vulnerabilities in other areas they are historically ignoring. After a vulnerability has been found in an area, the increased attention upon that area often yields multiple subsequent vulnerabilities [9].

Table 5: Excerpt of observed models

Area	Input	Identified Elements	Describes	Used for
Web Testing	Web Traffic	Access Rules	ACL model	Authentication Checks
Red-Teaming	Network Traffic	Communication Patterns	Expected Communication	Covert Channels
Network Tests	local data and network shares	File data and metadata	Company Data	find juicy information
OT tests	data flows	data flow model	system architecture	identify test scenarios
OT tests	network traffic	network commands	network protocol	protocol reversing
Web Testing	web traffic, context	used technologies	technology stack	potential vulnerabilities
Web Testing	web traffic	HTTP requests and responses	input model	generate tests

6.3 Dealing with Uncertainty

Pen-testers routinely have to deal with uncertainty as they lack transparency of the tested system: pen-testers must make assumptions about requirements, the tested system's architecture, as well as about accepted input values and the corresponding expected output parameters [59]. They evaluate those against their expectations, and if a system deviates, examine the deviation for exploitability. When in doubt, testers can escalate and query their clients, but this is deemed to be time-inefficient and thus minimized.

Examples of uncertainty would be a pen-tester issuing a HTTP request where they expect an "access denied" response but instead, receive a successful response containing data that cannot be clearly classified as belonging to the current user or not. Another example would be testing for time-based blind SQL injection vulnerabilities where the measured latency is not sufficiently deterministic for verifying the vulnerability. Similarly, second-order attacks cannot easily be attributed to the initial request but only to the operation that eventually contained the vulnerability.

Penetration testers modify existing valid requests to include malicious payloads. When these requests produce errors, the reason can be uncertain: was it a potential vulnerability? A successful input filtering algorithm? Or an application error that cannot be exploited? This classification impacts the selection of subsequent requests and attacks.

Another instance of uncertainty occurs during tool optimization: tool output is continuously used to further optimize subsequent tool invocations. Interviewees performed a sanity check if reported system fingerprints were feasible and forfeited them otherwise. In addition, some high-impact decisions, such as limiting the expectations to a single DBMS type, were verified with the client before incorporating them into tooling selection or configuration.

6.4 Don't Waste my Time

One theme discovered was that interviewees feel the need to be time-efficient. This might be related to tight time-budgets or very constrained test-bed availability being anathema to good test coverage. Shortcuts were taken to reduce menial tasks. For example, during internal network tests, a breach is already assumed. The interviewees defended this decision through "*this will eventually happen through social engineering anyways*". A similar argument was given for being provided accounts with local administrative privileges: "*a real attacker can just wait for the next 0-day*", or for disabling Anti-Virus solutions as evading them "*takes time not skill*". Tests with foregone conclusions were considered tedious, one example given was testing an Anti-Virus solution embedded within a web-application with different payloads. The repetitiveness of this task might contribute to this too. This aversion to responsible disclosure procedures might be correlated to bad experiences during prior disclosures: the vendor's responses were mostly "wasting" the interviewee's time.

6.5 Quality Control

Pen-Testers were concerned about the quality of their work, especially when working with high-stakes data such as health records — "*nobody wants to be that pen-tester that overlooked a vulnerability that was later exploited*". A tester's attention is also a limited resource: at least one pen-tester remarked that web application tests can be monotonous and that after 3–4 days their motivation degrades. Usage of checklists, automated baseline scans, and working in teams were encountered as quality improvement measures.

The applicability of checklists depends upon the testing domain. Some domains, e.g., Web-Applications or Mobile Applications, were seen as narrow and thus supporting the creation of security checklists. Other domains such as IoT were described as diverse and impeding the creation of a unified security checklist.

Checklists were often derived from open industry standards; they were maintained and extended by companies, but the resulting in-house checklists were seldom given back to the community and published. Common base for checklists was the OWASP trifecta of Vulnerability Top 10, Software Verification Standard and Testing Guide; instances of those are provided by OWASP for multiple domains such as Web-Applications [50, 61, 62], Mobile Applications [39, 52], IoT [36] or Firmware [35]. Surprisingly, neither MITRE ATT&CK© [55] nor PTES [17] were mentioned by our interviewees. Working in teams or asking colleagues can be seen as a broadening of the available experience pool or as employing a "human checklist". The use of automated tools as baseline scans that upheld minimal quality standards can also be interpreted as quality control. Interviewees mentioned usage of fully-automated commercial web vulnerability scanners such as NetSparker [10] or Acunetix [1] for this purpose. Some HTTP interception proxies, for example, PortSwigger BURP [12] or OWASP ZAP [16], have gained similar scanning capabilities. Those were used by some of the interviewees and encroached on terrain traditionally taken by web vulnerability scanners. In defense of testers, full coverage of the software-under-test is not feasible due to the black- to gray-box nature of security assignments.

Table 6: Excerpts of *Dealing with Change*: How is Security-Testing changing?

Sub-Theme	#	Representative Quotes
Impact of Frameworks	5	*"Security improves because frameworks help developers write secure code"* *"Pen-Testing has become boring as critical vulnerabilities are found less often"* *"Usage of secure frameworks pushed vulnerability hunting towards business logic."*
Defensive Mindset	3	*"Developer awareness about security has become better."*
Changing targets	7	*"In the future we might use social engineering not only for the initial foothold, but also for lateral movement"* *"Rich-client applications are still fun… they feel like web applications twenty years ago."* *"Active-Directory: I moved into this area because it is fun to break into a system within days."* *"The situation in OT will stay the same. It's hard to modernize all the legacy hard- and software."* *"Some OT networks are ransomware-ready."*

6.6 Dealing with Change

Security is in a constant state of flux. Compared to other disciplines, the existence of active adversaries — the struggle between red and blue teams — lead to a Red Queen's Race: participants must run to stand still [27, 38]. If not evolving, the respective adversary will overcome.

Interviewees lamented that some areas — breaking into web applications, breaching external infrastructure/perimeters, and reverse-engineering — have become harder due to boosted defenses such as usage of frameworks, improved default configurations, and heightened awareness of security posture (cf. Table 6). They are partially switching work areas, i.e., turning towards OT or internal network testing.

7 DISCUSSION AND IMPLICATIONS

We review our findings following the structure of our initial research questions to formulate points of discussion and implications for security researchers and practitioners.

7.1 Alignment between Research and Industry

We started this study with two questions, **"What do common security tests look like?"** and **"How do Hackers perform their tasks?"**. Those questions were broadly formulated to gain insight into how common assignments for practitioners look like, and how practitioners navigate their tasks within those assignments. These questions were particularly motivated by the fact that existing work is not grounded in the realities of offensive security practices.

7.1.1 Research Must Match a Project's Scope. During interviews, we identified typical security assignments with their respective typical resource allocations. Research should heed those resources allocated. For example, when targeting web vulnerability assessments, a typical project was given with 2–4 days of manual effort. Setting up a fuzzing pipeline, running the fuzzer, and analyzing its results is not feasible in this short time frame, thus rendering generic fuzzing rather infeasible for web security practitioners. Still, searching Google Scholar for "*fuzzing web applications*" yields 23000 results.

Given that interviewees mentioned the prevalence of web application frameworks and their preference for grey-box testing, SBOM-based solutions should be a better fit and would warrant additional research.

Most assignment types were done in solitary or as a paired team, indicating that research into collaborative solutions might be of limited use. The one exception using larger teams was Red-Teaming although here collaborative solutions integrated into C2-frameworks are already commonly used.

Automation with direct target-interactions were deemed problematic in the Red-Teaming and OT areas due to the sensitivity of their targets. In OT, security by obscurity still seems to be common, limiting the opportunities for source-code analysis based approaches. On the other hand, improvements to reverse-engineering binaries or protocols would be appreciated by practitioners.

Recently, the usage of Large Language Models (LLMs) for automated security testing has been explored [37]. While preliminary results look promising, to maximize their long-term impact the resulting automations should be aligned to the mentioned industry issues.

7.1.2 Security Researchers and Security Practitioners. Separating security into academic research and industry creates a false dichotomy. Industry itself is, at least, separated into security practitioners and security researchers. The former are practitioners that perform customer-specific assignments: those are the people that typically perform short-term penetration tests and directly communicate with clients to improve their security. In contrast, security researchers do not exclusively work on short-term client projects but spent time researching new attack techniques and vectors. An example of the former would be an anonymous pen-tester working on a different web-application every week; an example of the latter would be James Kettle investigating and documenting a new attack class, HTTP Request Smuggling, over many years [43, 44]. Security researchers search for new attack vectors or analyze a software product for a prolonged period of time to release exploits or be awarded CVEs. Security Practitioners are more focused on hunting configuration errors, exploiting well-known vulnerabilities, or identifying new instances of known attack classes. They utilize information and tools from security researchers for that.

Tools such as fuzzers are thus more applicable to security researchers than to security practitioners. The large amount of research into fuzzing indicates that academic research is targeting security researchers rather than practitioners and thus are only indirectly improving the security landscape when information from security researchers trickles down to practitioners.

7.2 Opportunities for Research

We now want to answer the important final question, **"What tedious or time-consuming areas could be improved?"** throughout the rest of this section and frame them as opportunities for future research that directly benefits security practitioners.

7.2.1 Automating Authorization Testing. For security tests with a relatively restricted scope such as *web application tests*, we suggest research into covering additional vulnerability classes. **Authorization Testing** is currently performed manually and was named one of the most time-consuming parts of testing and thus would be a fruitful target for automation research. Current gaps are manifold: detection of potential operations, accepted parameters, and potentially malicious parameters; generation of payloads as well as the assessment of an attack's success. A subtle problem is the classification of returned web pages and downloads into authorized and unauthorized content as this is highly context specific.

7.2.2 Gray-box Testing. The preference for gray-box testing by software security professionals was surprising and can have a significant impact on software testing design: if the target's configuration or source code can be accessed (or if the target is willing to instrumentalize the target software through sensors as is done in IAST), **automated software testing approaches using source-code or configuration** become increasingly feasible for security testing. Further research into automated source code and configuration file analysis from a security perspective, is currently underexplored and ripe for investigation. Research in this area yields dual-use tools, aiding both offensive security professionals searching for vulnerabilities as well as defensive software developers trying to prevent vulnerabilities from entering their code in the first place.

7.2.3 API Workflow Discovery for Security Test Generation. Interviewees lamented that the manual creation of API security test-cases is a tedious and time-consuming process. While the automation of API test generation would be advantageous, the following gaps currently prevent this: discovery of API endpoints and operations, generation of benign requests as a baseline, combining single requests into test flows using social and semantic information, deriving malicious test cases, and finally evaluating test outcomes. The **automatic generation of security test suites based upon API definitions and traffic patterns** would reduce testers' odium for utilizing this important class of testing. While there have been several works that propose approaches for API discovery [58, 65], the kind of discovery we envision would focus on maximizing coverage for security tests.

7.2.4 Information Discovery for Security Testing. Internal Network Tests and *Red-Teaming* are highly dependent on discovering and utilizing client-specific information. **Stealthy information gathering from compromised systems or network shares** is performed manually, and thus its efficiency could be improved. The goal is the automated identification of "juicy" information while reducing the number of read requests to minimize network impact or the chance of triggering intrusion detection systems. Research in this area would also benefit defenders as it would make forensic work, e.g., analyzing data breaches, more efficient.

7.2.5 Scaling Personalized Phishing with ML. Phishing is an important part of the red-teaming workflow and is commonly done manually, due to the nature of customization proper phishing requires. We see an opportunity to investigate the **increase of scalability of social engineering through machine learning** techniques. To create highly effective phishing mails, currently, mails are manually customized to fit the respective recipient. Machine learning techniques could automate this and thus provide **Spear Phishing at Scale**, as they have already been shown to personalize natural language communication in other domains [30, 42, 64]. An additional avenue for research is the identification of potential targets for social engineering, both from an external perspective (identifying initial recipients within a company) as well as **detecting informal networks within companies to enrich subsequent social-engineering campaigns** — this is an example of the red-teaming theme of "*understanding how companies function*".

7.2.6 Human-in-the-loop for OT Testing. OT professionals were weary of fully automated security tests due to the potential negative impact on stability and thus availability. We suggest research into supplemental areas while letting humans decide which attacks to execute. One example would be to **reduce the pain and effort of reverse engineering protocols**: OT tests are very time-bound thus there is little time for fuzzing or reverse-engineering OT protocols while the potential benefit might be immense due to security being provided by the obscurity of those protocols. Combining fuzzing with automatic reverse-engineering should yield large benefits [32]. The fear of potential fall-out has other consequences too: OT-tests are often performed by executing scenarios in lockstep with the customer. The scenarios are identified through threat modeling components and their data flows. To reduce the time spent on this effort, ways of **automatically deriving scenarios including attack paths** from system and data flow diagrams should be investigated.

Both OT professionals and red-teams were weary of fully automated testing solutions due to the potential negative impact upon stealth (red-teaming) or stability (OT). To facilitate the deployment of automated systems, **research into Human-Computer Interactions to bolster the acceptance of ML and automated systems** is needed. It is assumed that important topics will include humans-in-the-loop as well as the explainability of automated reasoning.

7.2.7 Studying Knowledge Communities for Security Testers. Our interview participants unsurprisingly felt the need for ongoing education w.r.t. new vulnerabilities and security trends. They synthesized information from multiple sources, the pivotal one being Twitter/X. Research on how developers stay current [53] and how development communities shape around news outlets [22] should be extended to the security arena, especially now that recent stewardship changes at Twitter might impact its reach. **Automated recommender systems utilizing diverse hacking news sources** such as news outlets, social media, and, the "darknet" should enable security professionals to stay up to date easier.

ACKNOWLEDGMENT

We thank the anonymous interview participants for their time, and Loren Kohnfelder and Geraldine Fitzpatrick for providing feedback.

REFERENCES

[1] [n. d.]. Acunetix: Web Vulnerability Scanner. https://www.acunetix.com/. Accessed: 2022-09-30.

[2] [n. d.]. BloodHoundAD: Six Degrees of Domain Admin. https://github.com/BloodHoundAD/BloodHound. Accessed: 2022-09-30.

[3] [n. d.]. Conti cyber attack on the HSE, Independent Post Incident Review. https://www.hse.ie/eng/services/publications/conti-cyber-attack-on-the-hse-full-report.pdf. Accessed: 2022-09-30.

[4] [n. d.]. Conti's Hacker Manuals — Read, Reviewed & Analyzed. https://www.akamai.com/blog/security/conti-hacker-manual-reviewed. Accessed: 2022-09-30.

[5] [n. d.]. Delve: Software Tool to Analyze Qualitative Data. https://delvetool.com/. Accessed: 2022-10-01.

[6] [n. d.]. DirBuster. https://www.kali.org/tools/dirbuster/. Accessed: 2022-09-30.

[7] [n. d.]. GhostPack/Certify: Active Directory certificate abuse. https://github.com/GhostPack/Certify. Accessed: 2022-09-30.

[8] [n. d.]. gobuster: Directory/File, DNS and VHost busting tool written in Go. https://github.com/OJ/gobuster. Accessed: 2022-09-30.

[9] [n. d.]. https://nakedsecurity.sophos.com/2021/07/16/more-printnightmare-we-told-you-not-to-turn-the-print-spooler-back-on/. https://nakedsecurity.sophos.com/2021/07/16/more-printnightmare-we-told-you-not-to-turn-the-print-spooler-back-on/. Accessed: 2022-10-03.

[10] [n. d.]. Invicti: Web Application Security for Enterprise. https://www.invicti.com/. Accessed: 2022-09-30.

[11] [n. d.]. Metasploit: Penetration Testing Software. https://github.com/rapid7/metasploit-framework. Accessed: 2022-09-30.

[12] [n. d.]. Methodology for Top 10. https://groups.google.com/a/owasp.org/g/leaders/c/pFLxDLE28ZA. Accessed: 2022-09-30.

[13] [n. d.]. Nessus Vulnerability Assessment Solution. https://www.tenable.com/products/nessus/nessus-professional. Accessed: 2022-09-30.

[14] [n. d.]. Nmap: the Network Mapper — Free Security Scanner. https://nmap.org. Accessed: 2022-09-30.

[15] [n. d.]. Nuclei: Fast and customizable vulnerability scanner based on simple YAML based DSL. https://github.com/projectdiscovery/nuclei. Accessed: 2022-09-30.

[16] [n. d.]. OWASP Zed Attack Proxy (ZAP). https://www.zaproxy.org/. Accessed: 2022-09-30.

[17] [n. d.]. PTES Technical Guidelines. http://www.pentest-standard.org/index.php/PTES_Technical_Guidelines. Accessed: 2022-09-30.

[18] [n. d.]. sqlmap: automatic SQL injection and database takeover tool. https://sqlmap.org/. Accessed: 2022-09-30.

[19] [n. d.]. Windows Print Spooler Remote Code Execution Vulnerability (CVE-2021-34527). https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-34527. Accessed: 2022-09-30.

[20] [n. d.]. Zero Day Initiative. https://www.zerodayinitiative.com/blog. Accessed: 2022-09-30.

[21] 2016-07-06. DIRECTIVE (EU) 2016/1148 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 6 July 2016 concerning measures for a high common level of security of network and information systems across the Union. https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016L1148. *Official Journal of the European Union* L 194 (2016-07-06), 1–30.

[22] Maurício Aniche, Christoph Treude, Igor Steinmacher, Igor Wiese, Gustavo Pinto, Margaret-Anne Storey, and Marco Aurélio Gerosa. 2018. How modern news aggregators help development communities shape and share knowledge. In *Proceedings of the 40th International conference on software engineering*. 499–510.

[23] Farzana Ahamed Bhuiyan, Akond Rahman, and Patrick Morrison. 2020. Vulnerability discovery strategies used in software projects. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering Workshops*. 13–18.

[24] Loyd Blankenship. 1986. The Conscience of a Hacker. *Phrack* 7 (Jan. 1986). http://www.phrack.org/archives/issues/7/3.txt

[25] Petar Boyanov. 2018. Educational exploiting the information resources and invading the security mechanisms of the operating system Windows 7 with the exploit Eternalblue and Backdoor Doublepulsar. *Association Scientific and Applied Research* 14 (2018), 34.

[26] Virginia Braun and Victoria Clarke. 2019. Reflecting on reflexive thematic analysis. *Qualitative research in sport, exercise and health* 11, 4 (2019), 589–597.

[27] Vit Bukac, Vaclav Lorenc, and Vashek Matyáš. 2014. Red queen's race: APT win-win game. In *Cambridge International Workshop on Security Protocols*. Springer, 55–61.

[28] Mariano Ceccato, Paolo Tonella, Cataldo Basile, Paolo Falcarin, Marco Torchiano, Bart Coppens, and Bjorn De Sutter. 2019. Understanding the behaviour of hackers while performing attack tasks in a professional setting and in a public challenge. *Empirical Software Engineering* 24 (2019), 240–286.

[29] Zakir Durumeric, Frank Li, James Kasten, Johanna Amann, Jethro Beekman, Mathias Payer, Nicolas Weaver, David Adrian, Vern Paxson, Michael Bailey, et al. 2014. The matter of heartbleed. In *Proceedings of the 2014 conference on internet measurement conference*. 475–488.

[30] Stefano Ferretti, Silvia Mirri, Catia Prandi, and Paola Salomoni. 2016. Automatic web content personalization through reinforcement learning. *Journal of Systems and Software* 121 (2016), 157–169.

[31] Jill J Francis, Marie Johnston, Clare Robertson, Liz Glidewell, Vikki Entwistle, Martin P Eccles, and Jeremy M Grimshaw. 2010. What is an adequate sample size? Operationalising data saturation for theory-based interview studies. *Psychology and health* 25, 10 (2010), 1229–1245.

[32] Hugo Gascon, Christian Wressnegger, Fabian Yamaguchi, Daniel Arp, and Konrad Rieck. 2015. Pulsar: Stateful black-box fuzzing of proprietary network protocols. In *Security and Privacy in Communication Networks: 11th EAI International Conference, SecureComm 2015, Dallas, TX, USA, October 26-29, 2015, Proceedings 11*. Springer, 330–347.

[33] Egon G Guba, Yvonna S Lincoln, et al. 1994. Competing paradigms in qualitative research. *Handbook of qualitative research* 2, 163-194 (1994), 105.

[34] Greg Guest, Arwen Bunce, and Laura Johnson. 2006. How many interviews are enough? An experiment with data saturation and variability. *Field methods* 18, 1 (2006), 59–82.

[35] Aaron Guzman. [n. d.]. OWASP Firmware Security Testing Methodology. https://scriptingxss.gitbook.io/firmware-security-testing-methodology/. Accessed: 2022-09-30.

[36] Aaron Guzman and Cedric Bassem. 2020. OWASP IoT Security Verification Standard. https://github.com/OWASP/IoT-Security-Verification-Standard-ISVS/releases/download/1.0RC/OWASP_ISVS-1.0RC-en_WIP_.pdf.

[37] Andreas Happe and Cito Jürgen. 2023. Getting pwn'd by AI: Penetration Testing with Large Language Models. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (San Francisco, USA) *(ESEC/FSE 2023)*. Association for Computing Machinery, New York, NY, USA, 5 pages. https://doi.org/10.1145/3611643.3613083

[38] Richard Harang and Felipe N Ducau. 2018. Measuring the speed of the Red Queen's Race. *BlackHat: Las Vegas, NV, USA* (2018).

[39] Carlos Holguera, Bernhard Müller, Sven Schleier, and Jeroen Willemsen. 2022. OWASP Mobile Application Security Verification Standard. https://github.com/OWASP/owasp-masvs/releases/latest/download/OWASP_MASVS-v1.4.2-en.pdf.

[40] Nicolas Huaman, Bennet von Skarczinski, Dominik Wermke, Christian Stransky, Yasemin Acar, Arne Dreißigacker, and Sascha Fahl. 2021. A large-scale interview study on information security in and attacks against small and medium-sized enterprises. In *In 30th USENIX Security Symposium*.

[41] (ISC)2. 2022. (ISC)2 CYBERSECURITY WORKFORCE STUDY 2022. https://www.isc2.org//-/media/ISC2/Research/2022-WorkForce-Study/ISC2-Cybersecurity-Workforce-Study.ashx. Accessed: 2023-04-28.

[42] Ioannis Katakis, Grigorios Tsoumakas, Evangelos Banos, Nick Bassiliades, and Ioannis Vlahavas. 2009. An adaptive personalized news dissemination system. *Journal of intelligent information systems* 32 (2009), 191–212.

[43] James Kettle. 2019. HTTP Desync Attacks: Request Smuggling Reborn. https://portswigger.net/research/http-desync-attacks-request-smuggling-reborn. Accessed: 2023-08-18.

[44] James Kettle. 2022. Browser-Powered Desync Attacks: A New Frontier in HTTP Request Smuggling. https://portswigger.net/research/browser-powered-desync-attacks. Accessed: 2023-08-18.

[45] Andrew G Kotulic and Jan Guynes Clark. 2004. Why there aren't more information security research studies. *Information & Management* 41, 5 (2004), 597–607.

[46] Sydney Lake. 2022. The cybersecurity industry is short 3.4 million workers—that's good news for cyber wages. https://fortune.com/education/articles/the-cybersecurity-industry-is-short-3-4-million-workers-thats-good-news-for-cyber-wages/. Accessed: 2023-04-28.

[47] Noella Mackenzie and Sally Knipe. 2006. Research dilemmas: Paradigms, methods and methodology. *Issues in educational research* 16, 2 (2006), 193–205.

[48] Nuthan Munaiah, Akond Rahman, Justin Pelletier, Laurie Williams, and Andrew Meneely. 2019. Characterizing attacker behavior in a cybersecurity penetration testing competition. In *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 1–6.

[49] Bruce Potter and Gary McGraw. 2004. Software security testing. *IEEE Security & Privacy* 2, 5 (2004), 81–85.

[50] Elie Saad and Rick Mitchell. 2020. OWASP Web Security Testing Guide. https://github.com/OWASP/wstg/releases/download/v4.2/wstg-v4.2.pdf.

[51] MNK Saunders and PC Tosey. 2013. *The layers of research design*. Technical Report. University of Surrey.

[52] Sven Schleier, Bernhard Mueller, Carlos Holguera, and Jeroen Willemsen. 2022. OWASP Mobile Application Security Testing Guide. https://github.com/OWASP/owasp-mastg/releases/latest/download/OWASP_MASTG-v1.5.0.pdf.

[53] Leif Singer, Fernando Figueira Filho, and Margaret-Anne Storey. 2014. Software engineering at the speed of light: how developers stay current using twitter. In *Proceedings of the 36th International Conference on Software Engineering*. 211–221.

[54] Justin Smith, Christopher Theisen, and Titus Barik. 2020. A Case Study of Software Security Red Teams at Microsoft. In *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 1–10.

[55] Blake E Strom, Andy Applebaum, Doug P Miller, Kathryn C Nickels, Adam G Pennington, and Cody B Thomas. 2018. Mitre att&ck: Design and philosophy. In

Technical report. The MITRE Corporation.
[56] Timothy C Summers. 2015. *How hackers think: A mixed method study of mental models and cognitive patterns of high-tech wizards*. Case Western Reserve University.
[57] Ari Takanen, Jared D Demott, Charles Miller, and Atte Kettunen. 2018. *Fuzzing for software security testing and quality assurance*. Artech House.
[58] Romina Torres, Boris Tapia, et al. 2011. Improving web api discovery by leveraging social information. In *2011 IEEE International Conference on Web Services*. IEEE, 744–745.
[59] Catia Trubiani, Pooyan Jamshidi, Jurgen Cito, Weiyi Shang, Zhen Ming Jiang, and Markus Borg. 2019. Performance Issues? Hey DevOps, Mind the Uncertainty. *IEEE Software* 36, 02 (2019), 110–117.
[60] Niek Jan van den Hout. 2019. *Standardised Penetration Testing? Examining the Usefulness of Current Penetration Testing Methodologies*. Ph. D. Dissertation.
[61] Andrew van der Stork, Brian Glas, Neil Smithline, and Torsten Gigler. 2021. OWASP Top 10:2021. https://owasp.org/Top10/0x00-notice/.
[62] Andrew van der Stork, Josh Grossman, Daniel Cuthbert, Elar Lang, and Jim Manico. 2021. OWASP Application Security Verification Standard. https://github.com/OWASP/ASVS/raw/v4.0.3/4.0/OWASP+Application+Security+Verification+Standard+4.0.3-en.pdf.
[63] Chris Wysopal, Lucas Nelson, Elfriede Dustin, and Dino Dai Zovi. 2006. *The art of software security testing: identifying software security flaws*. Pearson Education.
[64] Mengwei Xu, Feng Qian, Qiaozhu Mei, Kang Huang, and Xuanzhe Liu. 2018. Deeptype: On-device deep learning for input personalization service with minimal privacy concern. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 4 (2018), 1–26.
[65] Kuat Yessenov, Ivan Kuraj, and Armando Solar-Lezama. 2017. DemoMatch: API discovery from demonstrations. *ACM SIGPLAN Notices* 52, 6 (2017), 64–78.

Received 2023-05-18; accepted 2023-07-31