

Reading Sample

In this chapter we'll cover the fundamental questions to consider when it comes to the security of websites on the internet. Professional checks or audits have become a standard measure to minimize risks; however, the services offered vary greatly in quality and scope, making it difficult to make a suitable selection without research. This chapter discusses when to ask for external help, to what extent this makes sense, and what precautions and requirements are involved.



"External Security Checks"



Contents



Index



The Authors

Michael Kofler et al.

Hacking and Security

**The Comprehensive Guide to Penetration Testing
and Cybersecurity**

1141 pages | 07/2023 | \$59.95 | ISBN 978-1-4932-2425-8



www.rheinwerk-computing.com/5696

Chapter 10

External Security Checks

When it comes to the security of websites on the internet, professional checks or audits have become a standard measure to minimize risks. However, the services offered vary greatly in quality and scope, making it difficult to make a suitable selection without research.

Fundamental questions to consider, for example, include when to ask for external help, to what extent this makes sense, and what precautions and requirements are involved. Such questions arise as soon as you make the decision to have a security check performed. As a reader of this book, you'll very likely be confronted with these questions sooner or later, so we want to support you in this process with this chapter.

10.1 Reasons for Professional Checks

The first decision you need to make is whether you want to commission a professional check. With this book, we want to support you in the best possible way to carry out checks yourself. Due to the complexity and depth of the topic, however, we can at best facilitate your entry into the various subject areas; further examination of the individual topics is indispensable in order to be able to constantly identify new and adapted weak points.

Whether you use the services of external penetration testers thus depends, among other things, on your own expertise and experience. If you have the necessary resources to deal with the subject on a daily basis and can thus gain experience yourself, this book is an ideal starting point for carrying out as many checks as possible by yourself in the future.

You may also already have your own internal department of experts who can perform checks for you. If you're only dealing with the topic in passing, then you may have gathered enough knowledge in this book to at least be able to perform automated vulnerability scans yourself (a detailed description of this type of test follows in Section 10.2). Using such scans, you can raise the basic level of your security even with few resources and help your company implement basic security procedures.

For critical systems or for very customized implementations of applications or websites, you should call in professional support. Although many tools support the process

of technical security checks, vulnerabilities in complex environments and specially developed program code can actually only be found through the experience and knowledge of professional testers. In addition to the technical checks, they can then assist with the secure architecture and perform threat analyses with their experience.

Another reason to bring in external support may be that an independent third opinion is needed—for example, if there is a risk of operational blindness because one’s own team has already been entrusted with securing or programming the system under test. External testers are also needed when a third independent party is required to confirm or assess the security of the product, system, or network. This is usually the case during acceptance tests or when you purchase third-party products.

10.2 Types of Security Checks

Each check is a snapshot of the security of a given scope at a given point in time, with no guarantee of having found all vulnerabilities. When a vulnerability is identified, its existence is proven. If a vulnerability is not found, its existence cannot be ruled out.

The more time and knowledge you invest in the investigation, the more detailed the check can be, and the higher the probability of having found a larger proportion of the vulnerabilities that may exist. In order to keep the effort in relation to the value of the object to be checked or the available budget, different types of security checks are basically available.

10.2.1 Open-Source Intelligence

The term *open-source intelligence* (OSINT) originates from the military environment and describes an approach in which information is collected from publicly available data sources and its correlations are interpreted in a targeted manner. Intelligence is thus created from individual, often unrelated data points. For the area of security audits, this means that OSINT analyses use publicly available data on the internet to look for vulnerabilities, points of attack on a company, or key people in the company. Common activities include the following:

- Searching for relevant files via search engines such as Google or Bing
- Searching for information about deployed infrastructure components—for example, via advertised jobs or questions asked by employees in internet forums
- Extracting metadata—such as user names, operating system information, local paths, or software versions used—from files found on the internet
- Searching for personal interests and hobbies of selected employees who could be possible targets—for example, with the help of social media, such as Facebook, Twitter, or Runtastic

- Searching for deployed software with known vulnerabilities via infrastructure search engines, such as Shodan or Censys
- Searching databases published by hackers, so-called database leaks, on the internet

This list isn't exhaustive and will be adapted and extended accordingly depending on the objective of the analysis. An example of this is shown in Figure 10.1 where a search for "bund.de" is carried out using the Maltego OSINT tool. This tool examines published files for metadata, such as the version of Microsoft Word used to create the document.

Performing OSINT analyses can be useful, for example, when:

- internal security guidelines prescribe a certain way of handling information externally and whether these guidelines are being adhered to is to be checked,
- no security guidelines exist and an initial investigation should provide an assessment of whether internal data has already been published on the internet or has been unintentionally made accessible via the internet,
- individuals should be trained in the handling of both company and private data,
- likely targets of spear phishing attacks are to be identified and targeted phishing attacks (also called *whaling*) are to be prepared for, and
- possible points of attack on the infrastructure must be searched for without directly attacking the target company.

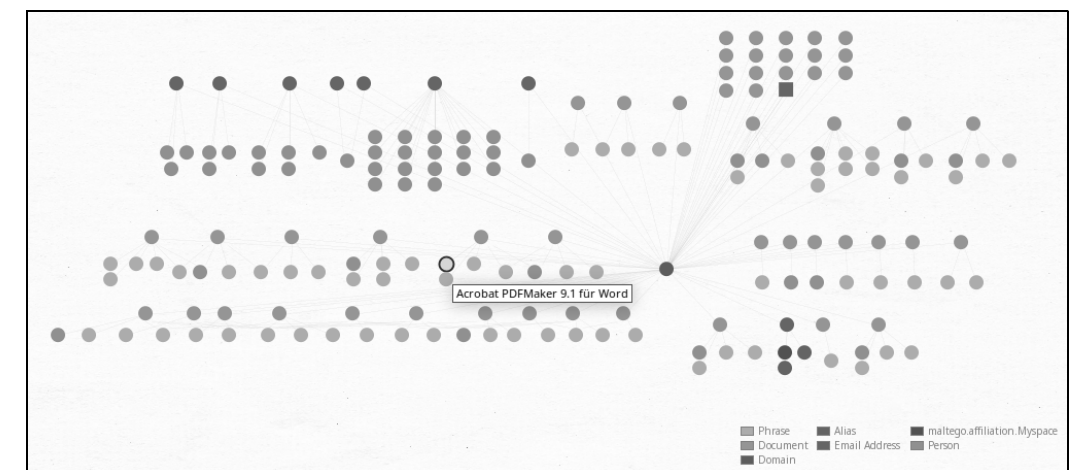


Figure 10.1 Information about Software Versions in Use Can Be Found Semiautomatically via OSINT

OSINT analyses may already be performed as part of other checks, such as red-teaming or spear-phishing assessments. However, these are usually very specific to the assessment in question. If a somewhat broader approach is also desired, this should already be included in the bidding phase.

A regular check of database leaks can be performed by companies themselves. This enables an early response and the ability to warn employees when accounts are hacked on external websites where a company email address has been used. In this way, the risk to the hacked account and the risk of the possible reuse of the password used there can be actively addressed.

10.2.2 Vulnerability Scan

The execution of a vulnerability scan is usually automated. The goal of the test is to perform a security check with as little effort as possible, identifying mainly easy-to-detect vulnerabilities, such as those that would be used by simple attackers (like script kiddies).

The main tools used are nmap, Nessus, Nexpose, OpenVAS, and other well-known vulnerability scanners. Figure 10.2 shows an example using the OpenVAS open-source vulnerability scanner.

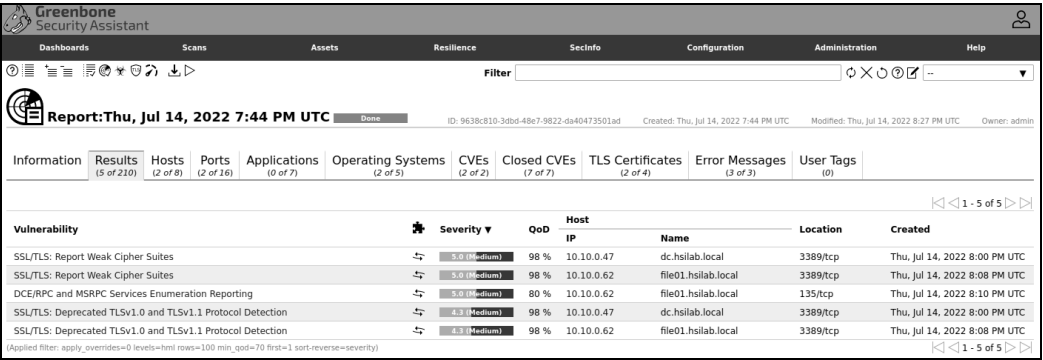


Figure 10.2 Example of Results from a Vulnerability Scan with OpenVAS

The costs of a vulnerability scan are generally lower than for targeted assessments because the proportion of manual checks is reduced to a minimum. Depending on the external provider, the client receives either direct output from a tool or a separate final report in which the tool results have already been cleaned of false results (*false positives*) and prepared in a way that is understandable for the client. Common activities include the following:

- Performing port scans to detect available services
- Performing a vulnerability scan with one or more vulnerability scanners
- Depending on the provider, the removal of incorrect results and preparation in the form of a separate report

Performing vulnerability scans may be useful, for example, when:

- no previous security check has taken place and an initial assessment of the security of standard infrastructure components and applications is to be made;

- regular, comparable scans of the infrastructure are to be performed, such as for early detection of possible security-relevant changes in the network or for performance measurement of the company’s own security management; or
- compliance requirements that can be tested in an automated manner should be verified in a cost-effective way.

With knowledge of vulnerability-scanning procedures, it should be understandable that vulnerability scanners can show their strengths especially in areas where fixed, predefined behaviors or responses can be assigned to specific, known vulnerabilities. This mainly concerns the scanning of standard services, the distribution of which is correspondingly widespread, so that the scan software manufacturers also provide as many test modules as possible for these services. In the area of custom software, the use of scanning software only makes sense in cooperation with professional testers, so penetration tests are advised instead of vulnerability scans for custom solutions and application-specific tests.

If scans are performed at regular intervals and are to be compared over time, or if large areas in particular are scanned, then communicating the results and coordinating them with system managers in particular presents a challenge for many security managers.

In this case, you can be supported by so-called vulnerability-management software, which provides an overview of all vulnerabilities and allows for sorting and filtering of results. It also gives system managers selective access to results, either directly integrated or provided as an interface. Subsequently, processing of vulnerabilities can be centrally monitored via a kind of ticket system. Examples of vulnerability-management software include vendor-specific products such as Tenable.io, Tenable.sc, and Acunetix Premium, or vendor-independent software such as VULCOM (see Figure 10.3).



Figure 10.3 Vulnerability Management Allows the Monitoring and Management of Vulnerabilities Discovered in an Enterprise

10.2.3 Vulnerability Assessment

Vulnerability assessment is the most common form of security check. Due to the high proportion of manual work by professional penetration testers, even hard-to-identify and individual vulnerabilities can be found. In addition, vulnerability scanners are used to ensure that no easily identifiable vulnerabilities are overlooked.

Unlike penetration tests, this test takes care to achieve the highest possible level of coverage. In this way, an attempt is made to find as many different vulnerabilities as possible in as many different functional areas of an application as possible in order to provide the broadest yet deepest possible insight into the security of a system or application. Common activities include the following:

- Performing port scans to detect available services
- Performing an automated vulnerability scan with one or more vulnerability scanners
- Using specially developed test tools that automate simple activities
- Manually checking all automatically identified vulnerabilities for false positives
- Manually searching for new, not yet known vulnerabilities in deployed software
- Preparing the results in a separate report

The step of manually searching for new vulnerabilities is the main component and can include testing network transmissions or checking web applications, as well as reverse engineering binaries. This is fully customized each time for the customer and the system under test. Conducting vulnerability assessments can be useful when:

- software developed in-house or by partner companies is to be subjected to a security check and as complete a picture as possible of the security of the software is desired, or
- the security of an internal network is to be evaluated and the focus is on obtaining as comprehensive a picture of security as possible.

The advantage of vulnerability assessments over vulnerability scans clearly lies in the possible detection of as yet unknown vulnerabilities in software. This also applies in particular to individual software and programs that are not widely used and are not detected by automated scanners. The advantage over penetration testing is that additional testing is performed across the board so that as many different attack vectors as possible can be identified and subsequently remediated.

Well-Known/Popular Software Is Not Automatically Secure!

Experience from past security assessments shows that there is no relation between the prominence of a piece of software or its manufacturer and the security of the software. Even in programs from large, well-known manufacturers, critical vulnerabilities, such

as unauthorized access to administration functionality, are regularly identified in the course of security checks.

Vulnerability Assessment versus Penetration Tests

In common usage, there is often no distinction between the terms *vulnerability assessment* and *penetration testing*. Therefore, you should make sure that you clearly communicate your expectations about the approach during an interview.

10.2.4 Penetration Test

Penetration tests aim to achieve the worst-case scenarios defined before launch. In most cases, this means a combination of automated tools with an additional high proportion of manual work. Compared to vulnerability assessments, however, the tests do not run broadly, but focus on achieving goals such as gaining administrator access in an application or gaining domain administration rights. Common activities include the following:

- Performing port scans to detect available services, which is possibly already limited to relevant services, instead of performing a scan across all possible ports
- Using specially developed test tools that automate simple activities
- Manually searching for new, not yet known vulnerabilities in deployed software
- Preparing the results in a separate report

Performing penetration tests can be useful, for example, when:

- safeguarding measures have already been taken and you need to check whether they provide sufficient security to prevent an attacker from reaching the target; or
- a security check is intended to demonstrate that the application has serious security vulnerabilities in order to have a sufficient argument for further, more detailed tests on the basis of this finding.

Penetration testing is well suited to identify particularly serious gaps as the focus of the application is precisely on these gaps and also the main portion of the time spent testing goes into this goal.

10.2.5 Red Teaming

Red teaming is usually understood to be a form of testing in which the scope of the test is not limited to one application, but tests whether, for example, access to certain data can be obtained. The application through which the tester gains access to the data isn't specified further so that the entire security concept is tested instead of individual

applications. In most cases, the red team assessment is also regarded directly as blue team training.

In this context, the *blue team* is the team for the tested party that is responsible for protecting the systems. This also means that, compared to many penetration tests, the red team tries to hide its activities to avoid detection. Common activities include the following:

- Searching for as many access points as possible to the desired asset or information
- Quickly analyzing which of the identified access points has the least security and would mean the quickest possible success
- Manually searching for mostly new, not yet known vulnerabilities in the respective applications
- Potentially compromising multiple servers and users on the network to get to the target asset

Conducting red team assessments may be appropriate, for example, when:

- security measures have already been taken and a realistic picture of the security of the entire network against targeted attacks is to be determined;
- your company has critical corporate data that needs special protection and you want to check whether existing security measures protect it effectively enough; or
- the internal team is to be trained practically in as realistic a manner as possible and in its own environment.

Red team assessments are a special form of testing in which several specialists from different areas may work together to achieve the previously defined goal. The assessment gives you the most realistic view of an attacker’s most likely attack path but will not evaluate all identified attack paths in detail unless otherwise agreed. Also, there is no detailed assessment of individual applications as the tester looks for the most promising attack opportunities across all applications.

The red team is also intended to train the response and knowledge of your blue team. In the best case scenario, a collaboration between the two teams is achieved, constantly improving your organization’s security and response to attacks.

Clarify the Common Understanding of the Test Type!

Even if the client and the contractor use the same words, they might not mean the same thing by them. Because vulnerability scans are often also sold as penetration tests, this can lead to both misunderstandings during meetings and unusable project results. You should therefore clarify the intent right at the beginning of the first meeting to prevent misunderstandings.

10.2.6 Purple Teaming

The term *purple teaming* is interpreted in different ways. In general, the goal of purple teaming is to promote cooperation between the red team and blue team in order to constantly improve the blue team’s capabilities and be better protected from real attacks in the future. In contrast to pure red teaming, purple teaming focuses primarily on the development of the blue team.

To make the development of the blue team as structured as possible, the first step is to define the type of attacker for which the blue team should be trained and the technical means available to the blue team. Derived from this, information from the MITRE ATT&CK framework can be used to determine the usual steps taken by these groups of offenders and, based on this, a test plan can be derived. Based on this test plan, individual attack steps are specifically recreated by the red team. A subsequent analysis then determines whether existing tools and the blue team were able to detect these activities. If not, the blue team will receive all the necessary information to detect such actions in the future. In this way, the blue team is gradually introduced to the possible attack steps over several iterations and trained to recognize them and initiate appropriate countermeasures.

Common activities include the following:

- Joint derivation of typical attacker types against which the company primarily wants to protect itself
- Researching common attack techniques from actual incidents, in many cases based on the MITRE ATT&CK framework
- Manual or automated execution of targeted test cases to test and train the blue team’s detection capabilities and response in a structured manner
- Disclosure of the red team’s attack techniques used to the blue team
- Examination of which attacks could have been detected and what changes are necessary to be able to detect these attacks in the future
- Regular repetition of training activities

An implementation of purple teaming can be useful, for example, if:

- the blue team is to be trained on new attacks or new offender groups;
- you want to examine in a structured way which attacks your existing security operations center (SOC) can detect, and this should be accompanied by targeted training and an improvement in detection capabilities; or
- you don’t yet have an existing SOC but want to build up the know-how and tools to detect attacks in a step-by-step and targeted manner.

MITRE ATT&CK Framework

The MITRE ATT&CK framework is a knowledge base of so-called attacker tactics and techniques. The database is constantly being expanded and adapted to the findings from actual incidents and the research of IT security companies. For more information, visit <https://attack.mitre.org>.

10.2.7 Bug Bounty Programs

Compared to the techniques presented in the previous chapters, bug bounty programs provide an alternative approach for companies to check their own software or systems for security. In contrast to the forms of security audits mentioned earlier in this chapter, the ordering company doesn't pay for the time invested by the security experts; it only pays if new, previously unknown vulnerabilities are reported by them.

Which and how many security professionals can participate as bug bounty hunters depends on the type of bug bounty program chosen, which range from public programs to private events that can only be entered by invitation. For more detailed information on bug bounty programs, see Chapter 19.

10.2.8 Type of Performance

Once you've decided on an assessment type from the previous sections, you'll be faced with the decision of which *performance method* to choose for that type. What follows is a description of the three basic methods and their advantages and disadvantages:

■ **Black box**

In a *black box check*, the tester receives as little information as possible about the target system before the assessment begins. The tester should be put in the same position as an external attacker with no prior knowledge of the system. This test type thus provides a comparatively realistic view if the tester is also given a correspondingly similar preparation time.

In most cases, however, the budget is limited, so only a limited amount of time is available for testing. This is also where the disadvantage of the black box check becomes apparent: because the attacker is not provided with any information, they must work it out themselves. For example, if several applications are available in different directories on the web server, the tester must find this out by trial and error.

However, due to the limited time, only a limited number of options can be searched for, so the tester may not find all of them and thus not test all of them for vulnerabilities. If the word list used were sorted differently, they might find other directories in the time available. This example shows that in order to get a realistic picture, the tester must either be given more time or accept that the check can only give a first impression regarding security.

■ **White box/glass box**

A *white box check* is the counterpart to the black box assessment. In this type of test, the tester is provided with all necessary information in advance or during the assessment. This ranges from additional administrator accounts on systems, to access to the source code of applications. This assessment type allows a very detailed and, compared to the other two types, more complete security assessment. Due to the abundance of information and the necessary working time for familiarization with the existing source code, the effort can also be higher than for grey box checks (discussed in the next point).

■ **Grey box/gray box**

Grey box checks are a kind of middle ground between black and white box checks. The tester should perform the security check as freely as possible, creatively and without too much influence from existing information, in order to reflect as realistic a view as possible from an unknown attacker. However, to perform the review as efficiently as possible, information is provided that the tester would find out on his own in a reasonable amount of time anyway. This information may include, for example, hidden folders, additional domains, or access to an additional administrator account to view existing admin functions.

10.2.9 Depth of Inspection: Attacker Type

To correctly estimate the scope of a project, in addition to the information already determined so far—such as the goal and technical scope, assessment type, and method of implementation—you need to know the depth of how detailed the assessment should be. This is determined by the definition of the attacker type.

Clearly Define Goal and Scope during Preliminary Discussions

For a successful project process, you should define the scope, goals, nongoals, and worst cases right at the beginning. This ensures that you as the client receive what you expect, but it's also important for the contractor to have this information in order to provide an accurate estimate.

Concerning the scope, you should describe all systems or application parts that will be tested. You should also specify which ones should be excluded from the test. In the goal definition, you have the opportunity to communicate what you expect from the test and what your objective is. The worst cases describe scenarios that shouldn't be possible under any circumstances. Even if these are not technically possible from your point of view, you should include the scenarios. After all, it's the job of the hackers you hire to find opportunities that no one has thought of before.

The following three types of attackers are examples of rough categorizations that may be defined differently in detail from one consulting firm to another:

■ **Script kiddie**

A *script kiddie* is an individual who has learned to attack and compromise systems using existing tools and tool-related instructions on the internet. Script kiddies don't have enough basic knowledge and programming skills to create their own exploits or adapt exploits that don't work. However, due to the prevailing crime-as-a-service model, this type of attacker is becoming less important.

■ **Advanced attacker**

These attackers have a good understanding of protocols, systems, and the technical background behind the available tools. They are able to create simple exploits themselves or adapt existing ones. However, time and resources are also limited for this attacker type.

■ **Expert**

This type of attacker is most easily compared to a state attacker or an attacker that can access a significant amount of time and resources. It's assumed that an expert spares no expense or effort to install even software that is considered to be secure on his own systems, and through it tries to find new, as yet unknown vulnerabilities in order to break into the target system.

Crime as a Service

The so-called crime-as-a-service model describes the development in recent years in which advanced perpetrator groups offer their developed special software for use, often via the darknet, as a service—for example, in a subscription model. This enables even people with little IT know-how to carry out very technically advanced attacks. The best-known form of this model is the so-called ransomware as a service.

10.2.10 Prior to the Order

If you commission a security check, you should make sure the consulting firm you hire asks about all of the items described in this section before providing an estimate. If an estimate is provided earlier, then it must be assumed that the consultant has already made decisions for the customer or the assessment is based on an automated scan that finds results independent of decisions such as the attacker type.

10.3 Legal Protection

When conducting security checks, there's a strong relationship of trust between you and your contractor, as the latter gains a deep insight into your internal processes, structures and data either beforehand or during the test at the latest. You thus give one or more external persons access specifically to the data you want to protect. This makes it all the more important to be cautious when choosing the right partner. Although

trust is important, it's still necessary to make appropriate legal arrangements to ensure that reviews run smoothly. In particular, this includes a nondisclosure agreement, a liability agreement, and written permission to perform the test:

■ **Nondisclosure agreement**

A *nondisclosure agreement* (NDA) obligates both parties to keep the information discussed in the course of the project confidential. For you as the client, this document is important to oblige the contractor to keep their silence about your internal data. Penalties vary, but are usually set between \$50,000 and the actual contract value. It's also customary to state that, at the request of the client, the data must be destroyed on the contractor's side, provided that this is compatible with the legal requirements for the contractor. Confidentiality can be made mandatory for a few years after the end of the project.

■ **Liability agreement**

A *liability agreement* is usually provided by the contractor, either as a separate document or integrated into the bid or permission-to-attack document. In Europe, it's usually the case that the contractor receives a release from liability in this liability agreement during the course of the project for all tests agreed upon in the offer. The client is obliged to provide the necessary information and to perform regular backups of the systems under test in order to minimize the risk. Excluded from the liability agreement are grossly negligent acts of the contractor.

■ **Permission to attack**

In the *permission-to-attack* (PTA) document, the client grants the contractor written permission to carry out attacks on specific targets, usually defined by IP address or DNS name, within the scope of the agreed upon project and the associated tests within a predefined period of time. Especially in the case of tests from the internet, you should make sure that the contractor also tells you the IP addresses from which the tests are performed so that you have the opportunity during and after tests to determine whether they were actual attacks or tests related to the security check.

■ **Order processing contract**

Depending on the objective of the assessment, the contractor may have access to personal data within the scope of the General Data Protection Regulation (GDPR). In these cases, you should be sure to conclude an order processing agreement with your service provider in order to also regulate the correct and secure processing of the data in accordance with the GDPR.

If you have an internal security team that regularly monitors internal network traffic, then you should have the test team provide you with the IP addresses they're currently using on an ongoing basis so that you can share them with your own incident or monitoring team in parallel. This prevents additional costs from being incurred due to incorrectly categorized attacks.

Legal Support

Note that the descriptions in this section can only give you an impression of the content of the documents. It's advisable to consult a lawyer for one-time preparation of the documents and to have them check all legal issues.

10.4 Objectives and Scope

As with any other project planning, to successfully complete a project, it's necessary to determine the scope, goals, and nongoals of the project prior to its start. This is usually agreed upon in the course of a joint meeting and recorded in writing in the form of a quotation or project description. When discussing the scope, you should be clear about in how much detail the respective systems or software should be tested. Your project partner will help you with this, and in some cases the response will already be codetermined based on the defined attacker type.

To provide a concrete estimate, your contractor will most likely need detailed information about the target system, such as a description of existing services, a description of the parameters of available interfaces and endpoints, a network map, or even screenshots or documentation of the application under test. If this involves internal, sensitive data, then you should establish an NDA with the potential contractor before the conversation.

In addition to defining the handling, it's advantageous for a test if you already talk through the so-called worst-case scenarios, the typical use cases, and the hot spots that you would like to have examined in more detail with your contractor in advance. All of this information will then help the penetration testers tailor the test to your needs as best as possible and spend the available time most efficiently on those aspects of the assessment that are most important to you.

Just as important as defining the scope, objectives, and worst-case scenarios is defining nonobjectives. Clearly define together which systems or applications should explicitly *not* be tested or which attack types (e.g., denial-of-service tests) should explicitly *not* be performed. For example, many security assessments explicitly exclude denial-of-service attacks.

10.4.1 Sample Objective

The goal of the black box vulnerability assessment is to determine the risk of attacks on the MySecurePortal application from the perspective of an external attacker from the internet. The assessment is intended to provide the broadest possible overview of different attack capabilities of the existing infrastructure.

The type of attacker chosen was a so-called script kiddie, who has experience in using ready-made tools, but no in-depth expertise in finding as yet unknown vulnerabilities or programming their own tools. Explicitly excluded are social engineering attacks on employees or customers; denial-of-service attacks that aim to restrict the availability of the application also are not permitted.

10.4.2 Sample Worst-Case Scenarios

The following three worst-case scenarios were defined for the project: direct access to the database or operating system, read or write access to customer data, and access to administrative functions (starting with */admin/* in the URL).

10.4.3 Sample Scope

The scope includes the entire MySecurePortal application, which can be accessed at the following address: *https://mysecureportaltest.targetcompany.com*. The infrastructure components that build on it are also part of this. Testers are provided with administrator access to the application so that they can test external access to internal functions as comprehensively as possible. However, all tests are still performed from an attacker's perspective without credentials.

The server is hosted and operated by the customer and exclusively provides the test instance of the MySecurePortal application. If other applications are found on the server under the same IP address during the test, they will also be included in the scope, as they can potentially be used as a point of attack on the application data. However, the main focus of the assessment should be on the MySecurePortal application itself.

10.5 Implementation Methods

The approach used to perform security assessments depends on the test in question, but also, and above all, on the approach chosen by the respective provider. Most providers use a combination of approaches customized to their own needs. This ensures that a minimum standard is maintained during the assessment, but the testers are deliberately given freedom in their approach. After all, security assessments are a creative activity.

The following is a brief explanation of the most popular approaches mentioned by many providers:

- **Open Web Application Security Project**
The Open Web Application Security Project (OWASP; *https://www.owasp.org*) is a nonprofit organization founded in 2004 to make know-how for the development and the operation of secure web applications available publicly and independent of manufacturers.

OWASP operates several subprojects. Among the best known are the OWASP Top 10, a list of top vulnerabilities in web applications; and software such as the Zed Attack Proxy (ZAP) for analyzing web traffic. The *OWASP Web Security Testing Guide* briefly describes a few organizational details about web application testing and puts its main focus on detailed elaboration of tests to be performed, both in their execution and with subsequent recommendations. In comparison, the Application Security Verification Standard is used to describe requirements or tests that can be used by architects, developers, and testers. The testing guide again can be used to perform the tests.

■ NIST SP 800-115

NIST has issued several documents related to system security. One of the best known is the SP 800-115 standard, *Technical Guide to Information Security Testing and Assessment* (<http://s-prs.co/v569673>).

In contrast to OWASP, this standard does not deal with technical details; instead, it describes the organizational process of security assessments.

■ Penetration Testing Execution Standard

Like OWASP, the Penetration Testing Execution Standard (PTES) was born out of the community and thrives on the input of security professionals. The standard is available online in the form of a wiki (<http://s-prs.co/v569674>) and covers both the organizational preparation and the process as well as the technical execution of tests. The PTES hasn't been updated for a long time. However, many forms of attack documented there are still valid. Because of its practical nature, PTES continues to be a good starting point for those interested.

10.6 Reporting

As a result of a security assessment, the vulnerabilities identified in the course of the project, including date appropriate recommendations for remediation and usually also a risk assessment, are made available to the client. When commissioning a security assessment, you should consider the best way to process the results yourself. Some companies want a completed report in PDF format, as this represents a written third-party opinion that is as unbiased as possible. Also, the results are usually documented in detail and in a way that can be understood by different groups of people.

Other companies only share the results with the internal IT team, which has been working with IT security assessment results for some time, and there is no provision for sharing them with the company management. In this case, a detailed description of the results may not be necessary, so rough documentation in the form of a table will suffice. If you've been working with your pen testing partner for a long time anyway, a digital exchange of results is also possible, which minimizes the effort on both sides. Here,

you can make your wishes known to the contractor right at the start so that you also receive the results in the form in which you can best process them.

Also, when selecting a vendor, each potential contractor should be able to present you with a demo report. This allows you to evaluate whether the target groups in your company are being addressed appropriately, whether results come directly from a vulnerability scanner or additional value has been added by the expert knowledge of the testers, and how the risk calculation of the respective findings can be integrated into your own risk management.

Every company sets its focus differently, and the type of documentation can also vary. In general, however, a report can roughly be divided into the following sections:

■ Management summary

This section presents the scope and key findings of the test from a business perspective on one page. There, the goal and the time period in which the assessment took place are first described in two sentences. Also, positive aspects from the assessment should be highlighted and the most critical weaknesses should be presented. The management summary should always end with a recommendation for further course of action.

In most cases, the client knows the recipient of the management summary better than the contractor. For this reason, it can be advantageous to agree on the structure and content shortly before finalizing the report. This also gives the client the opportunity to include additional information: how much of a threat it is to the business, what points are particularly important to readers of the summary, and so on.

■ Technical summary

This section is intended for technical management and summarizes the actual findings in a table so that chief security officers (CSOs) and chief information security officers (CISOs), for example, can get a quick overview of the technical security status.

■ General recommendations

General recommendations and follow-up steps are presented either directly in the management summary together with the results or in a separate section. The goal of this section is to provide management with recommendations for subsequent steps.

■ Scope and organizational details

For the report to be understood as a stand-alone document even at a later date, it's necessary that all project agreements, scope info, contact persons, schedule details, and other organizational circumstances are written down.

■ Technical details

In this section, all results are documented in detail. This includes a general description, details of the specific security problem investigated, recommendations for remediation, and an assessment of the risk.

How the chapter is further structured depends on the results of the assessment and on the customer's wishes or how the customer can most easily process the data. This means that, for example, an outline of sections by host and then a listing of vulnerabilities by host can be made if the customer has defined responsible parties by host within the company. In other situations, it may be useful to distinguish between infrastructure and web applications. In any case, however, the same description form should be chosen for each documented vulnerability, which we'll explain in more detail ahead.

A description of a vulnerability itself is again divided into several sections, which should help different reader groups to extract the information content important for them more easily and to better understand the issue or the solution to it. The following outline is only an example and may vary from company to company:

■ Description

The description is intended primarily for people with basic technical knowledge, but without specific security know-how. They should understand the fundamental problem of the identified vulnerability and be able to assess the associated risk without having to delve into the specific technical details.

■ Technical utilization

This section explains the technical details of the vulnerability. With the information contained here, it should be possible for the person who is responsible for the system or application to understand the problem so that a suitable solution can be found.

In most cases, screenshots or code examples support the understanding. It's even better if the reader is provided with a detailed description of how to recreate the problem using open-source means. References based on results from commercial tools should be avoided, as the reader may not have the financial means to invest in paid tools.

■ Recommendations

The recommendations should provide the most accurate solutions possible to the specific vulnerabilities identified. If possible, dedicated configuration recommendations also should be provided, which the respective system owner can adopt directly.

■ Valuation

The vulnerability assessment can be based on different aspects and will vary from company to company.

The assessment is based either on the company's own benchmarks—for example, a quality-oriented classification into high, medium, and low categories—or on well-known approaches, such as Microsoft DREAD (damage, reproducibility, exploitability, affected users, and discoverability) or the Common Vulnerability Scoring System (CVSS).

Each of these methods has its advantages and disadvantages, and there is no clear industry standard. More important than choosing the perfect assessment method is that the assessment is consistent, understandable, and correct.

Figure 10.4 shows an example of a finding for using a default password. Because this is an unencrypted Telnet service, there should be at least one more finding in the report about using Telnet.

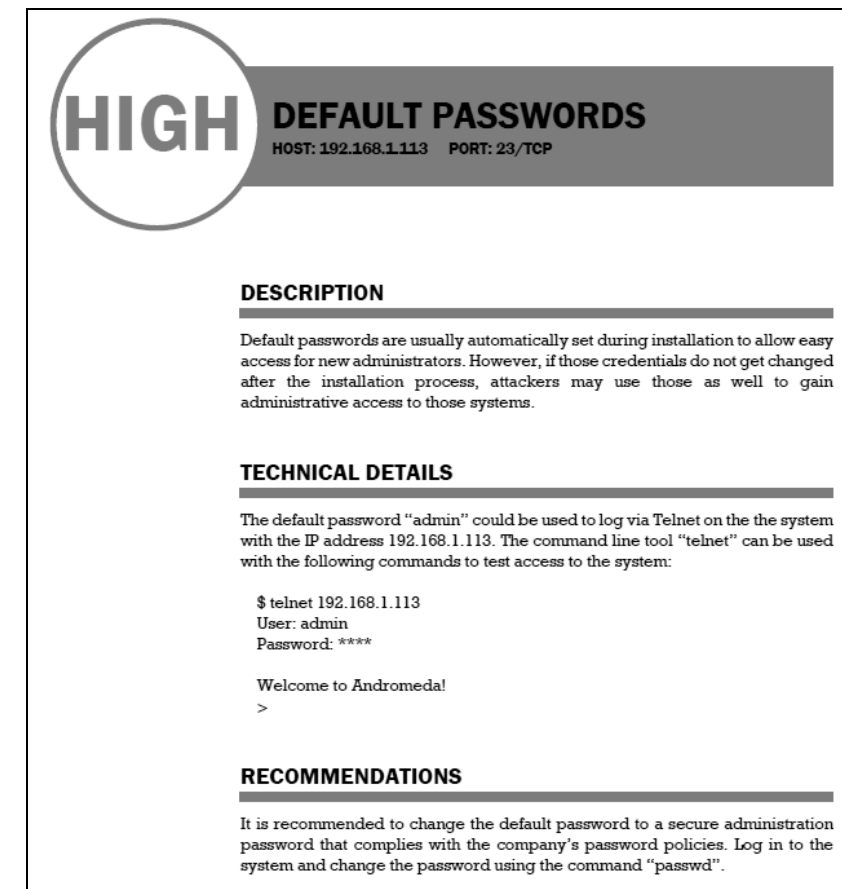


Figure 10.4 Example of a Finding in a Final Report

10.7 Selecting the Right Provider

Once you've made the decision to have the security assessment performed by an external provider, the search for the most suitable provider for your purposes begins. As in other areas, each provider has its own focus, so you should choose them based on the required objective. You should ask the vendor for experience in the specific task area in which you need assistance, such as reverse engineering, for example.

The assessment type can also play a role in the decision. If a simple vulnerability scan is required, the tester's experience is less important. However, if you're looking for a highly manual assessment, such as a penetration test, then the quality of the test is very much dependent on the knowledge and experience of the tester in question. These people are specialists who have to keep their know-how up to date on a daily basis, constantly expand it, and adapt it to current technologies. This also means that price can't be the sole reason for exclusion in these forms of assessment, as the quality of the results may also depend heavily on it.

At the very least, you should consider the following points when choosing the right provider:

- **Recommendations are important**

In addition to a technically flawless execution, the quality of a service also includes general aspects such as the organizational process, communication with the customer, understanding the customer's problems and the objective, and the preparation and communication of the results.

In most cases, the totality of these points provides an overall impression that can best be obtained via existing customers. You should proactively ask friends, acquaintances, and companies in your sector about their experiences with penetration testing companies. This way you can obtain honest feedback most of the time as well as an initial list of interesting providers. Referrals are one of the most valuable forms of evaluation for potential providers.

- **Customer names are optional**

Due to existing confidentiality agreements (NDA contracts), it's often not possible—or only possible with additional effort—for providers to name existing customers. Even if you're given customer names, a provider is most likely to list those customers who were satisfied with the service.

- **Description of previous experiences**

An adequate means of better assessing experience in connection with one's own task is to ask the provider for examples of similar projects and approaches used. This gives you an impression of how often and at what level of detail the provider has already covered these topics and how confident they feel in this area.

- **Description of the procedure**

You should get a description of the provider's approach. However, it's better to ask about the specific issues in the current project and how the provider intends to solve them than to mention known standards. The approach is interesting from both organizational and technical perspectives.

You can also check how the vendor handles critical systems that may be in place or how they typically work with internal security teams.

If critical systems are present, the call to handle them with care should also come from the provider side. If you have someone on the team with technical security expertise, you can also ask about the tools used to get a better sense of how advanced the technical understanding is for complex tasks.

- **Check the report**

Ask to see a demo report to assess whether it contains all the information you expect. This also allows you to react at an early stage if you still need to make appropriate adjustments so that you can process the results further in the company with as little effort as possible.

Contents

Preface	33
1 Introduction	39
1.1 Hacking	39
1.1.1 Hacking Contests, Capture the Flag	40
1.1.2 Penetration Test versus Hacking	41
1.1.3 Hacking Procedure	41
1.1.4 Hacking Targets	44
1.1.5 Hacking Tools	46
1.2 Security	47
1.2.1 Why Are IT Systems So Insecure?	48
1.2.2 Attack Vectors	49
1.2.3 Who Is Your Enemy?	53
1.2.4 Intrusion Detection	55
1.2.5 Forensics	55
1.2.6 Ten Steps to Greater Safety	56
1.2.7 Security Is Not Visible	57
1.2.8 Security Is Inconvenient	57
1.2.9 The Limits of This Book	58
1.3 Exploits	58
1.3.1 Zero-Day Exploits	60
1.3.2 The Value of Exploits	61
1.3.3 Exploit Types	61
1.3.4 Finding Vulnerabilities and Exploits	62
1.3.5 Common Vulnerabilities and Exposures	62
1.3.6 Common Vulnerability Scoring System	62
1.3.7 Vulnerability and Exploit Databases	63
1.3.8 Vulnerability Scanner	64
1.3.9 Exploit Collections	65
1.4 Authentication and Passwords	65
1.4.1 Password Rules	66
1.4.2 Phishing	66
1.4.3 Storage of Passwords (Hash Codes)	67
1.4.4 Alternatives to Passwords	68
1.4.5 Fast Identity Online	69

1.5	Security Risk IPv6	70
1.5.1	Security Complications	71
1.6	Legal Framework	72
1.6.1	Unauthorized Hacking Is Punishable by Law	72
1.6.2	Negligent Handling of IT Security Is Also a Criminal Offense	73
1.6.3	European General Data Protection Regulation	74
1.6.4	Critical Infrastructure, Banks	74
1.6.5	Security Guidelines and Standards	75
1.7	Security Organizations and Government Institutions	75
2	Kali Linux	77
2.1	Kali Alternatives	77
2.2	Trying Out Kali Linux without Installation	78
2.2.1	Verifying the Download	78
2.2.2	Verifying the Signature of the Checksum File	79
2.2.3	Trying Kali Linux in VirtualBox	80
2.2.4	Saving Data Permanently	83
2.2.5	Forensic Mode	83
2.3	Installing Kali Linux in VirtualBox	84
2.3.1	Option 1: Using a Prebuilt VirtualBox Image	85
2.3.2	Option 2: Installing Kali Linux Yourself	85
2.3.3	Installation	85
2.3.4	Login and sudo	88
2.3.5	Time Zone and Time Display	88
2.3.6	Network Connection	88
2.3.7	Using Kali Linux via SSH	89
2.3.8	Clipboard for Kali Linux and the Host Computer	91
2.4	Kali Linux and Hyper-V	91
2.5	Kali Linux in the Windows Subsystem for Linux	93
2.5.1	Kali Linux in Graphic Mode	94
2.5.2	WSL1 versus WSL2	95
2.5.3	Practical Experience	96
2.6	Kali Linux on Raspberry Pi	96
2.7	Running Kali Linux on Apple PCs with ARM CPUs	97
2.8	Simple Application Examples	99
2.8.1	Address Scan on the Local Network	100

2.8.2	Port Scan of a Server	101
2.8.3	Hacking Metasploitable	103
2.9	Internal Details of Kali	103
2.9.1	Basic Coverage	103
2.9.2	Package Sources	104
2.9.3	Rolling Release	104
2.9.4	Performing Updates	104
2.9.5	Installing Software	105
2.9.6	Python 2	105
2.9.7	Network Services and Firewall	106
2.9.8	kali-tweaks	106
2.9.9	Undercover Mode	107
2.9.10	PowerShell	107
3	Setting Up the Learning Environment: Metasploitable, Juice Shop	109
3.1	Honeypots	110
3.2	Metasploitable 2	110
3.2.1	Installation in VirtualBox	111
3.2.2	Network Settings	111
3.2.3	Host-Only Network	112
3.2.4	Using Metasploitable 2	113
3.2.5	Hacking Metasploitable 2	114
3.2.6	rlogin Exploit	115
3.3	Metasploitable 3 (Ubuntu Variant)	116
3.3.1	Why No Ready-Made Images?	117
3.3.2	Requirements	117
3.3.3	Installation	118
3.3.4	Starting and Stopping Metasploitable 3	120
3.3.5	Administrating Metasploitable 3	120
3.3.6	Network Configuration	121
3.3.7	Hacking Metasploitable 3	122
3.4	Metasploitable 3 (Windows Variant)	123
3.4.1	Administrating Metasploitable 3	124
3.4.2	SSH login	126
3.4.3	Internal Details and Installation Variants	126
3.4.4	Overview of Services in Metasploitable 3 (Windows Variant)	127
3.4.5	Hacking Metasploitable 3	129

3.5	Juice Shop	133
3.5.1	Installation with Vagrant	133
3.5.2	Installation with Docker	134
3.5.3	Docker in Kali Linux	135
3.5.4	Hacking Juice Shop	135
4	Hacking Tools	137
4.1	nmap	138
4.1.1	Syntax	138
4.1.2	Examples	140
4.1.3	Variants and Alternatives	141
4.2	hydra	142
4.2.1	Syntax	142
4.2.2	Password Lists	144
4.2.3	Examples	144
4.2.4	Attacks on Web Forms and Login Pages	145
4.2.5	Alternatives	146
4.3	sslyze, sslscan, and testssl	148
4.3.1	sslscan and sslyze	148
4.3.2	testssl	149
4.3.3	Online Tests	150
4.4	whois, host, and dig	151
4.4.1	whois	152
4.4.2	host	152
4.4.3	dig	153
4.4.4	dnsrecon	154
4.5	Wireshark	154
4.5.1	Installation	155
4.5.2	Basic Functions	156
4.5.3	Working Techniques	158
4.5.4	Alternatives	159
4.6	tcpdump	159
4.6.1	Syntax	160
4.6.2	Examples	161
4.6.3	ngrep	162
4.7	Netcat (nc)	163
4.7.1	Syntax	163

4.7.2	Examples	163
4.7.3	socat	166
4.8	OpenVAS	166
4.8.1	Installation	167
4.8.2	Starting and Updating OpenVAS	169
4.8.3	Operation	169
4.8.4	Alive Test	172
4.8.5	Setting Up Tasks Yourself	173
4.8.6	High Resource Requirements	175
4.8.7	Alternatives	175
4.9	Metasploit Framework	176
4.9.1	Operation in Kali Linux	177
4.9.2	Installation on Linux	177
4.9.3	Installation on macOS	178
4.9.4	Installation on Windows	179
4.9.5	Updates	180
4.9.6	The Metasploit Console (“msfconsole”)	180
4.9.7	A Typical “msfconsole” Session	181
4.9.8	Searching Modules	182
4.9.9	Applying Modules	183
4.9.10	Meterpreter	185
4.10	Empire Framework	187
4.10.1	Installation	188
4.10.2	Getting to Know and Setting Up Listeners	189
4.10.3	Selecting and Creating Stagers	190
4.10.4	Creating and Managing Agents	192
4.10.5	Finding the Right Module	193
4.10.6	Obtaining Local Administrator Rights with the Empire Framework	195
4.10.7	The Empire Framework as a Multiuser System	197
4.10.8	Alternatives	197
4.11	The Koadic Postexploitation Framework	197
4.11.1	Installing the Server	198
4.11.2	Using Helper Tools in the Program	199
4.11.3	Creating Connections from a Client to the Server	199
4.11.4	Creating a First Connection: Zombie 0	201
4.11.5	The Modules of Koadic	202
4.11.6	Extending Rights and Reading Password Hashes	203
4.11.7	Conclusion and Countermeasures	205
4.12	Social Engineer Toolkit	205
4.12.1	Syntax	206
4.12.2	Example	206

4.12.3	The dnstwist Command	210
4.12.4	Other SET Modules	211
4.12.5	Alternatives	212
4.13	Burp Suite	212
4.13.1	Installation and Setup	213
4.13.2	Modules	213
4.13.3	Burp Proxy	214
4.13.4	Burp Scanner	216
4.13.5	Burp Intruder	217
4.13.6	Burp Repeater	218
4.13.7	Burp Extensions	218
4.13.8	Alternatives	219
4.14	Sliver	219
4.14.1	Installation	220
4.14.2	Implants and Listeners	220
4.14.3	Other C2 Frameworks	224

5 Offline Hacking 227

5.1	BIOS/EFI: Basic Principles	228
5.1.1	The Boot Process	228
5.1.2	EFI Settings and Password Protection	229
5.1.3	UEFI Secure Boot	229
5.1.4	When the EFI Is Insurmountable: Remove the Hard Drive	230
5.2	Accessing External Systems	230
5.2.1	Booting the Notebook with Kali Linux	230
5.2.2	Reading the Windows File System	231
5.2.3	Vault Files	233
5.2.4	Write Access to the Windows File System	235
5.2.5	Linux	235
5.2.6	macOS	236
5.2.7	Does That Mean That Login Passwords Are Useless?	236
5.3	Accessing External Hard Drives or SSDs	236
5.3.1	Hard Drives and SSDs Removed from Notebooks	237
5.4	Resetting the Windows Password	237
5.4.1	Tools	238
5.4.2	Undesirable Side Effects	239
5.4.3	Resetting the Local Windows Password Using chntpw	240
5.4.4	Activating a Windows Administrator User via chntpw	242

5.5	Resetting Linux and macOS Passwords	244
5.5.1	Resetting a Linux Password	244
5.5.2	Resetting a macOS Password	245
5.6	Encrypting Hard Drives	246
5.6.1	BitLocker	246
5.6.2	Access to BitLocker File Systems on Linux (dislocker)	249
5.6.3	BitLocker Security	250
5.6.4	BitLocker Alternatives	251
5.6.5	macOS: FileVault	252
5.6.6	Linux: Linux Unified Key Setup	253
5.6.7	Security Concerns Regarding LUKS	253
5.6.8	File System Encryption on the Server	254

6 Passwords 255

6.1	Hash Procedures	256
6.1.1	Hash Collisions	257
6.1.2	SHA-2 and SHA-3 Hash Codes	258
6.1.3	Checksums or Hash Codes for Downloads	258
6.2	Brute-Force Password Cracking	259
6.2.1	Estimating the Time Required for Password Cracking	259
6.3	Rainbow Tables	260
6.3.1	Password Salting	261
6.4	Dictionary Attacks	262
6.5	Password Tools	263
6.5.1	John the Ripper: Offline CPU Cracker	264
6.5.2	hashcat: Offline GPU Cracker	265
6.5.3	Crunch: Password List Generator	268
6.5.4	hydra: Online Cracker	269
6.5.5	makepasswd: Password Generator	270
6.5.6	One-Time Secret: Send Passwords by Email	270
6.6	Default Passwords	271
6.7	Data Breaches	272
6.8	Multifactor Authentication	275
6.9	Implementing Secure Password Handling	276
6.9.1	Implementation Tips	277

7	IT Forensics	279
7.1	Methodical Analysis of Incidents	281
7.1.1	Digital Traces	281
7.1.2	Forensic Investigation	281
7.1.3	Areas of IT Forensics	282
7.1.4	Analysis of Security Incidents	284
7.2	Postmortem Investigation	284
7.2.1	Forensic Backup of Memory	284
7.2.2	Recovering Deleted Files by File Carving	286
7.2.3	Metadata and File Analysis	288
7.2.4	System Analyses with Autopsy	290
7.2.5	Basic System Information	292
7.2.6	Reading the Last Activities	295
7.2.7	Analyzing Web Activities	296
7.2.8	Tracing Data Exchanges	298
7.3	Live Analysis	300
7.3.1	Finding User Data	301
7.3.2	Called Domains and URLs	301
7.3.3	Active Network Connections	302
7.3.4	Extracting the TrueCrypt Password	302
7.4	Forensic Readiness	303
7.4.1	Strategic Preparations	303
7.4.2	Operational Preparations	304
7.4.3	Effective Logging	304
7.4.4	Protection against Tampering	305
7.4.5	Integrity Verification	305
7.4.6	Digital Signatures	305
7.5	Summary	305
8	Wi-Fi, Bluetooth, and SDR	307
8.1	802.11x Systems: Wi-Fi	307
8.1.1	Preparation and Infrastructure	308
8.1.2	Wireless Equivalent Privacy	310
8.1.3	WPA/WPA-2: Wireless Protected Access	315
8.1.4	Wireless Protected Setup	317
8.1.5	Wi-Fi Default Passwords	320
8.1.6	WPA-2-KRACK Attack	321

8.1.7	WPA-2 Enterprise	322
8.1.8	Wi-Fi Client: Man-in-the-Middle	323
8.1.9	WPA-3	325
8.2	Collecting WPA-2 Handshakes with Pwnagotchi	325
8.3	Bluetooth	332
8.3.1	Bluetooth Technology	332
8.3.2	Identifying Bluetooth Classic Devices	334
8.3.3	Hiding (and Still Finding) Bluetooth Devices	339
8.3.4	Bluetooth Low Energy (BTLE)	343
8.3.5	Listening In on Bluetooth Low Energy Communication	344
8.3.6	Identifying Apple Devices via Bluetooth	346
8.3.7	Bluetooth Attacks	347
8.3.8	Modern Bluetooth Attacks	349
8.4	Software-Defined Radios	349
8.4.1	SDR Devices	351
8.4.2	Decoding a Wireless Remote Control	353
9	Attack Vector USB Interface	359
9.1	USB Rubber Ducky	360
9.1.1	Structure and Functionality	360
9.1.2	DuckyScript	360
9.1.3	Installing a Backdoor on Windows 11	363
9.1.4	Use With Duck Encoder to Create the Finished Payload	366
9.2	Digispark: A Wolf in Sheep's Clothing	367
9.2.1	Downloading and Setting Up the Arduino Development Environment	368
9.2.2	The Script Language of the Digispark	370
9.2.3	Setting Up a Linux Backdoor with Digispark	371
9.3	Bash Bunny	375
9.3.1	Structure and Functionality	375
9.3.2	Configuring the Bash Bunny	377
9.3.3	Status LED	378
9.3.4	Software Installation	379
9.3.5	Connecting to the Bash Bunny	379
9.3.6	Connecting the Bash Bunny to the Internet: Linux Host	381
9.3.7	Connecting the Bash Bunny to the Internet: Windows Host	382
9.3.8	Bunny Script: The Scripting Language of the Bash Bunny	384
9.3.9	Using Custom Extensions and Functions	386

9.3.10	Setting Up a macOS Backdoor with Bash Bunny	387
9.3.11	The payload.txt Files for Switch1 and Switch2	390
9.3.12	Updating the Bash Bunny	394
9.3.13	Key Takeaways	395
9.4	P4wnP1: The Universal Talent	396
9.4.1	Structure and Functionality	396
9.4.2	Installation and Connectivity	397
9.4.3	HID Scripts	398
9.4.4	CLI Client	399
9.4.5	An Attack Scenario with the P4wnP1	399
9.4.6	Creating a Dictionary	400
9.4.7	Launching a Brute-Force Attack	401
9.4.8	Setting Up a Trigger Action	404
9.4.9	Deploying the P4wnP1 on the Target System	405
9.4.10	Key Takeaways	405
9.5	MalDuino W	406
9.5.1	The Web Interface of the MalDuino W	407
9.5.2	The Scripting Language and the CLI	408
9.5.3	An Attack Scenario with the MalDuino W	408
9.5.4	How Does the Attack Work?	409
9.5.5	Key Takeaways	412
9.6	Countermeasures	412
9.6.1	Hardware Measures	413
9.6.2	Software Measures	413

10 External Security Checks 419

10.1	Reasons for Professional Checks	419
10.2	Types of Security Checks	420
10.2.1	Open-Source Intelligence	420
10.2.2	Vulnerability Scan	422
10.2.3	Vulnerability Assessment	424
10.2.4	Penetration Test	425
10.2.5	Red Teaming	425
10.2.6	Purple Teaming	427
10.2.7	Bug Bounty Programs	428
10.2.8	Type of Performance	428
10.2.9	Depth of Inspection: Attacker Type	429
10.2.10	Prior to the Order	430

10.3	Legal Protection	430
10.4	Objectives and Scope	432
10.4.1	Sample Objective	432
10.4.2	Sample Worst-Case Scenarios	433
10.4.3	Sample Scope	433
10.5	Implementation Methods	433
10.6	Reporting	434
10.7	Selecting the Right Provider	437

11 Penetration Testing 441

11.1	Gathering Information	442
11.1.1	Searching for Information about a Company	442
11.1.2	Using Metadata of Published Files	445
11.1.3	Identifying the Structure of Email Addresses	447
11.1.4	Database and Password Leaks	449
11.1.5	Partial Automation with Maltego	450
11.1.6	Automating Maltego Transforms	456
11.1.7	Defense	458
11.2	Initial Access with Code Execution	459
11.2.1	Checking External IP Addresses of the PTA	459
11.3	Scanning Targets of Interest	463
11.3.1	Gathering Information via DNS	463
11.3.2	Detecting Active Hosts	465
11.3.3	Detecting Active Services with nmap	467
11.3.4	Using nmap in Combination with Metasploit	469
11.4	Searching for Known Vulnerabilities Using nmap	470
11.5	Exploiting Known Vulnerabilities Using Metasploit	472
11.5.1	Example: GetSimple CMS	474
11.6	Attacking Using Known or Weak Passwords	478
11.7	Email Phishing Campaigns for Companies	481
11.7.1	Organizational Preparatory Measures	481
11.7.2	Preparing a Phishing Campaign with Gophish	483
11.8	Phishing Attacks with Office Macros	490
11.9	Phishing Attacks with ISO and ZIP Files	494
11.9.1	Creating an Executable File with Metasploit	495
11.9.2	Creating a File with ScareCrow to Bypass Virus Scanners	499

11.9.3	Disguising and Deceiving: From EXE to PDF File	502
11.9.4	Defense	503
11.10	Attack Vector USB Phishing	504
11.11	Network Access Control and 802.1X in Local Networks	506
11.11.1	Getting to Know the Network by Listening	506
11.11.2	Network Access Control and 802.1X	507
11.12	Extending Rights on the System	509
11.12.1	Local Privilege Escalation	510
11.12.2	Bypassing Windows User Account Control Using the Default Setting	512
11.12.3	Bypassing UAC Using the Highest Setting	515
11.13	Collecting Credentials and Tokens	517
11.13.1	Reading Passwords from Local and Domain Accounts	518
11.13.2	Bypassing Windows 10 Protection against mimikatz	519
11.13.3	Stealing Windows Tokens to Impersonate a User	520
11.13.4	Matching Users with DCSync	521
11.13.5	Golden Ticket	522
11.13.6	Reading Local Password Hashes	523
11.13.7	Broadcasting within the Network by Means of Pass-the-Hash	524
11.13.8	Man-in-the-Middle Attacks in Local Area Networks	527
11.13.9	Basic Principles	527
11.13.10	LLMNR/NBT-NS and SMB Relaying	534
11.14	SMB Relaying Attack on Ordinary Domain Users	540
11.14.1	Command-and-Control	542
12	Securing Windows Servers	543
12.1	Local Users, Groups, and Rights	544
12.1.1	User and Password Properties	545
12.1.2	Local Admin Password Solution	548
12.2	Manipulating the File System	553
12.2.1	Attacks on Virtualized Machines	556
12.2.2	Protection	557
12.2.3	Attacking through the Registry	557
12.3	Server Hardening	558
12.3.1	Ensure a Secure Foundation	559
12.3.2	Harden New Installations	559
12.3.3	Protect Privileged Users	559

12.3.4	Threat Detection	560
12.3.5	Secure Virtual Machines as Well	560
12.3.6	Security Compliance Toolkit	560
12.4	Microsoft Defender	561
12.4.1	Defender Configuration	562
12.4.2	Defender Administration via PowerShell	563
12.5	Windows Firewall	564
12.5.1	Basic Configuration	565
12.5.2	Advanced Configuration	565
12.5.3	IP Security	567
12.6	Windows Event Viewer	568
12.6.1	Classification of Events	569
12.6.2	Log Types	570
12.6.3	Linking Actions to Event Logs	572
12.6.4	Windows Event Forwarding	573
12.6.5	Event Viewer Tools	575
13	Active Directory	579
13.1	What Is Active Directory?	579
13.1.1	Domains	580
13.1.2	Partitions	580
13.1.3	Access Control Lists	583
13.1.4	Security Descriptor Propagator	585
13.1.5	Standard Permissions	588
13.1.6	The Confidentiality Attribute	592
13.2	Manipulating the Active Directory Database or its Data	592
13.2.1	ntdsutil Command	593
13.2.2	dsamain Command	594
13.2.3	Accessing the AD Database via Backups	595
13.3	Manipulating Group Policies	596
13.3.1	Configuration Files for Group Policies	598
13.3.2	Example: Changing a Password	600
13.4	Domain Authentication: Kerberos	603
13.4.1	Kerberos: Basic Principles	603
13.4.2	Kerberos in a Theme Park	604
13.4.3	Kerberos on Windows	604
13.4.4	Kerberos Tickets	605
13.4.5	krbtgt Account	606

13.4.6	TGS Request and Reply	608
13.4.7	Older Authentication Protocols	610
13.5	Attacks against Authentication Protocols and LDAP	611
13.6	Pass-the-Hash Attacks: mimikatz	612
13.6.1	Setting up a Defender Exception	613
13.6.2	Windows Credentials Editor	614
13.6.3	mimikatz	617
13.6.4	The mimikatz “sekurlsa” Module	618
13.6.5	mimikatz and Kerberos	621
13.6.6	PowerSploit	623
13.7	Golden Ticket and Silver Ticket	624
13.7.1	Creating a Golden Ticket Using mimikatz	625
13.7.2	Silver Ticket and Trust Ticket	627
13.7.3	BloodHound	628
13.7.4	Deathstar	628
13.8	Reading Sensitive Data from the Active Directory Database	628
13.9	Basic Coverage	631
13.9.1	Core Server	631
13.9.2	Roles in the Core Server	632
13.9.3	Nano Server	633
13.9.4	Updates	633
13.9.5	Hardening the Domain Controller	634
13.10	More Security through Tiers	635
13.10.1	Group Policies for the Tier Model	636
13.10.2	Authentication Policies and Silos	636
13.11	Protective Measures against Pass-the-Hash and Pass-the-Ticket Attacks...	639
13.11.1	Kerberos Reset	639
13.11.2	Kerberos Policies	641
13.11.3	Kerberos Claims and Armoring	642
13.11.4	Monitoring and Detection	643
13.11.5	Microsoft Advanced Threat Analytics: Legacy	644
13.11.6	Other Areas of Improvement in Active Directory	647
14	Securing Linux	649
14.1	Other Linux Chapters	649
14.2	Installation	650
14.2.1	Server Distributions	650

14.2.2	Partitioning the Data Medium	652
14.2.3	IPv6	653
14.3	Software Updates	654
14.3.1	Is a Restart Necessary?	655
14.3.2	Automating Updates	655
14.3.3	Configuring Automatic Updates on RHEL	656
14.3.4	Configuring Automatic Updates on Ubuntu	656
14.3.5	The Limits of Linux Update Systems	657
14.4	Kernel Updates: Live Patches	658
14.4.1	Kernel Live Patches	659
14.4.2	Kernel Live Patches for RHEL	660
14.4.3	Kernel Live Patches on Ubuntu	660
14.5	Securing SSH	661
14.5.1	sshd_config	661
14.5.2	Blocking the Root Login	662
14.5.3	Authentication with Keys	663
14.5.4	Authenticating with Keys in the Cloud	664
14.5.5	Blocking IPv6	665
14.6	2FA with Google Authenticator	665
14.6.1	Setting Up Google Authenticator	666
14.6.2	2FA with Password and One-Time Code	668
14.6.3	What Happens if the Smartphone Is Lost?	669
14.6.4	Authy as an Alternative to the Google Authenticator App	670
14.7	2FA with YubiKey	670
14.7.1	PAM Configuration	671
14.7.2	Mapping File	671
14.7.3	SSH Configuration	672
14.8	Fail2ban	673
14.8.1	Installation	673
14.8.2	Configuration	674
14.8.3	Basic Parameters	676
14.8.4	Securing SSH	676
14.8.5	Securing Other Services	677
14.8.6	Securing Custom Web Applications	678
14.8.7	Fail2ban Client	678
14.9	Firewall	679
14.9.1	From Netfilter to nftables	680
14.9.2	Basic Principles	680
14.9.3	Determining the Firewall Status	682
14.9.4	Defining Rules	683

14.9.5	Syntax for Firewall Rules	685
14.9.6	Example: Simple Protection of a Web Server	687
14.9.7	FirewallID: RHEL	688
14.9.8	firewall-cmd Command	689
14.9.9	ufw: Ubuntu	691
14.9.10	Firewall Protection in the Cloud	693
14.10	SELinux	693
14.10.1	Concept	693
14.10.2	The Right Security Context	694
14.10.3	Process Context: Domain	695
14.10.4	Policies	696
14.10.5	SELinux Parameters: Booleans	696
14.10.6	Status	697
14.10.7	Fixing SELinux Issues	698
14.11	AppArmor	699
14.11.1	AppArmor on Ubuntu	700
14.11.2	Rules: Profiles	701
14.11.3	Structure of Rule Files	701
14.11.4	Rule Parameters: Tunables	703
14.11.5	Logging and Maintenance	703
14.12	Kernel Hardening	704
14.12.1	Changing Kernel Options Using sysctl	704
14.12.2	Setting Kernel Boot Options in the GRUB Configuration	706
14.13	Apache	706
14.13.1	Certificates	707
14.13.2	Certificate Files	708
14.13.3	Apache Configuration	709
14.13.4	HTTPS Is Not HTTPS	710
14.14	MySQL and MariaDB	712
14.14.1	MySQL versus MariaDB	712
14.14.2	Login System	713
14.14.3	MySQL and MariaDB on Debian/Ubuntu	714
14.14.4	Securing MySQL on RHEL	715
14.14.5	Securing MariaDB on RHEL	715
14.14.6	Hash Codes in the “mysql.user” Table: Old MySQL and MariaDB Versions	716
14.14.7	Privileges	717
14.14.8	Server Configuration	718
14.15	Postfix	719
14.15.1	Postfix: Basic Settings	719

14.15.2	Sending and Receiving Emails in Encrypted Form	720
14.15.3	Spam and Virus Defense	722
14.16	Dovecot	724
14.16.1	Using Custom Certificates for IMAP and POP	724
14.16.2	SMTP Authentication for Postfix	725
14.17	Rootkit Detection and Intrusion Detection	726
14.17.1	chkrootkit	727
14.17.2	rkhunter	728
14.17.3	Lynis	729
14.17.4	ISPProtect	730
14.17.5	Snort	731
14.17.6	Verifying Files from Packages	731
14.17.7	Scanning for Suspicious Ports and Processes	732
15	Security of Samba File Servers	735
15.1	Preliminary Considerations	735
15.1.1	Compiling Samba, SerNet Packages	736
15.2	Basic CentOS Installation	737
15.2.1	Partitions	737
15.2.2	Disabling IPv6	738
15.2.3	Installing Samba Packages on CentOS	741
15.3	Basic Debian Installation	741
15.3.1	The Partitions	741
15.3.2	Disabling IPv6	742
15.3.3	Installing Samba Packages on Debian	743
15.4	Configuring the Samba Server	743
15.4.1	Configuring the Kerberos Client	745
15.5	Samba Server in Active Directory	746
15.5.1	Joining the Samba Server	746
15.5.2	Testing the Server	748
15.6	Shares on the Samba Server	750
15.6.1	File System Rights on Linux	750
15.6.2	File System Rights on Windows	750
15.6.3	Special Shares on a Windows Server	751
15.6.4	The Admin Share on Samba	751
15.6.5	Creating the Admin Share	751
15.6.6	Creating the User Shares	752

15.7 Changes to the Registry	755
15.7.1 Accessing the Registry from Windows	757
15.8 Samba Audit Functions	758
15.9 Firewall	760
15.9.1 Testing the Firewall Script	763
15.9.2 Starting Firewall Script Automatically	764
15.10 Attack Scenarios on Samba File Servers	765
15.10.1 Known Vulnerabilities in Recent Years	766
15.11 Checking Samba File Servers	768
15.11.1 Tests with nmap	768
15.11.2 Testing the Samba Protocols	769
15.11.3 Testing the Open Ports	769
15.11.4 smb-os-discovery	771
15.11.5 smb2-capabilities	771
15.11.6 ssh-brute	772
16 Intrusion Detection Systems	775
16.1 Intrusion Detection Methods	775
16.1.1 Pattern Recognition: Static	775
16.1.2 Anomaly Detection (Dynamic)	777
16.2 Host-Based versus Network-Based Intrusion Detection	778
16.2.1 Host-Based IDS	778
16.2.2 Network-Based IDS	779
16.2.3 NIDS Metadata	780
16.2.4 NIDS Connection Contents	782
16.3 Responses	783
16.3.1 Automatic Intrusion Prevention	783
16.3.2 Walled Garden	784
16.3.3 Swapping Computers	784
16.4 Bypassing and Manipulating Intrusion Detection	785
16.4.1 Insertions	785
16.4.2 Evasions	786
16.4.3 Resource Consumption	786
16.5 Snort	787
16.5.1 Installation and Launch	787
16.5.2 Getting Started	789
16.5.3 IDS or IPS	790

16.5.4 Configuration	791
16.5.5 Modules	791
16.5.6 Snort Event Logging	792
16.6 Snort Rules	793
16.6.1 Syntax of Snort Rules	793
16.6.2 Service Rules	794
16.6.3 General Rule Options	795
16.6.4 Matching Options	797
16.6.5 Hyperscan	798
16.6.6 Inspector-Specific Options	799
16.6.7 Managing Rule Sets with PulledPork	800
17 Security of Web Applications	803
17.1 Architecture of Web Applications	803
17.1.1 Components of Web Applications	804
17.1.2 Authentication and Authorization	805
17.1.3 Session Management	806
17.2 Attacks against Web Applications	806
17.2.1 Attacks against Authentication	806
17.2.2 Session Hijacking	807
17.2.3 HTML Injection	808
17.2.4 Cross-Site Scripting	811
17.2.5 Session Fixation	815
17.2.6 Cross-Site Request Forgery	815
17.2.7 Directory Traversal	816
17.2.8 Local File Inclusion	817
17.2.9 Remote File Inclusion	819
17.2.10 File Upload	820
17.2.11 SQL Injection	821
17.2.12 sqlmap	823
17.2.13 Advanced SQL Injection: Blind SQL Injection (Boolean)	824
17.2.14 Advanced SQL Injection: Blind SQL Injection (Time)	825
17.2.15 Advanced SQL Injection: Out-of-Band Data Exfiltration	827
17.2.16 Advanced SQL Injection: Error-Based SQL Injection	827
17.2.17 Command Injection	828
17.2.18 Clickjacking	830
17.2.19 XML Attacks	832
17.2.20 Server Side Request Forgery	834
17.2.21 Angular Template Injection	835

17.2.22	Attacks on Object Serialization	835
17.2.23	Vulnerabilities in Content Management Systems	836
17.3	Practical Analysis of a Web Application	837
17.3.1	Information Gathering	838
17.3.2	Testing SQL Injection	840
17.3.3	Directory Traversal	845
17.3.4	Port Knocking	847
17.3.5	SSH Login	849
17.3.6	Privilege Escalation	850
17.3.7	Automatic Analysis via Burp	855
17.4	Protection Mechanisms and Defense against Web Attacks	859
17.4.1	Minimizing the Server Signature	860
17.4.2	Turning Off the Directory Listing	860
17.4.3	Restricted Operating System Account for the Web Server	861
17.4.4	Running the Web Server in a “chroot” Environment	861
17.4.5	Disabling Unneeded Modules	861
17.4.6	Restricting HTTP Methods	862
17.4.7	Restricting the Inclusion of External Content	862
17.4.8	Protecting Cookies from Access	863
17.4.9	Server Timeout	863
17.4.10	Secure Socket Layer	863
17.4.11	HTTP Strict Transport Security	864
17.4.12	Input and Output Validation	865
17.4.13	Web Application Firewall	866
17.5	Security Analysis of Web Applications	867
17.5.1	Code Analysis	868
17.5.2	Analysis of Binary Files	869
17.5.3	Fuzzing	869
18	Software Exploitation	871
18.1	Software Vulnerabilities	871
18.1.1	Race Conditions	871
18.1.2	Logic Error	872
18.1.3	Format String Attacks	873
18.1.4	Buffer Overflows	873
18.1.5	Memory Leaks	873

18.2	Detecting Security Gaps	874
18.3	Executing Programs on x86 Systems	874
18.3.1	Memory Areas	874
18.3.2	Stack Operations	876
18.3.3	Calling Functions	879
18.4	Exploiting Buffer Overflows	884
18.4.1	Analysis of the Program Functionality	884
18.4.2	Creating a Program Crash	886
18.4.3	Reproducing the Program Crash	888
18.4.4	Analysis of the Crash	889
18.4.5	Offset Calculation	891
18.4.6	Creating the Exploit Structure	893
18.4.7	Generating Code	895
18.4.8	Dealing with Prohibited Characters	896
18.5	Structured Exception Handling	899
18.6	Heap Spraying	901
18.7	Protective Mechanisms against Buffer Overflows	903
18.7.1	Address Space Layout Randomization	903
18.7.2	Stack Canaries or Stack Cookies	904
18.7.3	Data Execution Prevention	905
18.7.4	SafeSEH and Structured Exception Handling Overwrite Protection	906
18.7.5	Protection Mechanisms against Heap Spraying	907
18.8	Bypassing Protective Measures against Buffer Overflows	907
18.8.1	Bypassing Address Space Layout Randomization	907
18.8.2	Bypassing Stack Cookies	908
18.8.3	Bypassing SafeSEH and SEHOP	908
18.8.4	Return-Oriented Programming	908
18.8.5	DEP Bypass	911
18.9	Preventing Buffer Overflows as a Developer	914
18.10	Spectre and Meltdown	915
18.10.1	Meltdown	915
18.10.2	Defense Measures	916
18.10.3	Proof of Concept (Meltdown)	917
18.10.4	Spectre	918
18.10.5	Proof of Concept (Spectre)	919
18.10.6	The Successors to Spectre and Meltdown	921

19 Bug Bounty Programs	923
19.1 The Idea Behind Bug Bounties	923
19.1.1 Providers	923
19.1.2 Variants	924
19.1.3 Earning Opportunities	925
19.2 Reporting Vulnerabilities	926
19.2.1 Testing Activities	926
19.3 Tips and Tricks for Analysts	927
19.3.1 Scope	927
19.3.2 Exploring the Response Quality of the Target Company	927
19.3.3 Take Your Time	927
19.3.4 Finding Errors in Systems or Systems with Errors	928
19.3.5 Spend Money	928
19.3.6 Get Tips, Learn from the Pros	928
19.3.7 Companies Buy Companies	928
19.3.8 Creating a Test Plan	929
19.3.9 Automating Standard Processes	929
19.4 Tips for Companies	930
20 Security in the Cloud	931
20.1 Overview	931
20.1.1 Arguments for the Cloud	932
20.1.2 Cloud Risks and Attack Vectors	933
20.1.3 Recommendations	934
20.2 Amazon Simple Storage Service	935
20.2.1 Basic Security and User Management	937
20.2.2 The aws Command	938
20.2.3 Encrypting Files	939
20.2.4 Public Access to Amazon S3 Files	941
20.2.5 Amazon S3 Hacking Tools	942
20.3 Nextcloud and ownCloud	943
20.3.1 Installing Nextcloud	944
20.3.2 Blocking Access to the “data Folder”	946
20.3.3 Performing Updates	947

20.3.4 File Encryption	948
20.3.5 Security Testing for ownCloud and Nextcloud Installations	949
20.3.6 Brute-Force Attacks and Protection	950
21 Securing Microsoft 365	953
21.1 Identities and Access Management	954
21.1.1 Azure Active Directory and Microsoft 365	954
21.1.2 User Management in AAD	957
21.1.3 Application Integration	958
21.2 Security Assessment	960
21.3 Multifactor Authentication	961
21.3.1 Preliminary Considerations	962
21.3.2 Enabling Multifactor Authentication for a User Account	962
21.3.3 User Configuration of Multifactor Authentication	963
21.3.4 App Passwords for Incompatible Applications and Apps	965
21.4 Conditional Access	969
21.4.1 Creating Policies	970
21.4.2 Conditions for Policies	972
21.4.3 Access Controls	973
21.5 Identity Protection	975
21.5.1 Responding to Vulnerabilities	975
21.6 Privileged Identities	976
21.6.1 Enabling Privileged Identities	977
21.6.2 Configuring a User as a Privileged Identity	979
21.6.3 Requesting Administrator Permissions	980
21.7 Detecting Malicious Code	982
21.7.1 Protection for File Attachments	986
21.7.2 Protection for Files in SharePoint Online and OneDrive for Business	988
21.7.3 Protection for Links	989
21.7.4 Protection for Links in Office Applications	991
21.8 Security in Data Centers	992
21.8.1 Encryption of Your Data	992
21.8.2 Access Governance	994
21.8.3 Audits and Privacy	995

22 Mobile Security	997
22.1 Android and iOS Security: Basic Principles	997
22.1.1 Sandboxing	998
22.1.2 Authorization Concept	998
22.1.3 Protection against Brute-Force Attacks when the Screen Is Locked	999
22.1.4 Device Encryption	1000
22.1.5 Patch Days	1001
22.2 Threats to Mobile Devices	1003
22.2.1 Theft or Loss of a Mobile Device	1003
22.2.2 Unsecured and Open Networks	1004
22.2.3 Insecure App Behavior at Runtime	1004
22.2.4 Abuse of Authorizations	1006
22.2.5 Insecure Network Communication	1007
22.2.6 Attacks on Data Backups	1009
22.2.7 Third-Party Stores	1013
22.3 Malware and Exploits	1014
22.3.1 Stagefright (Android)	1019
22.3.2 Pegasus (iOS)	1023
22.3.3 Spy Apps	1024
22.4 Technical Analysis of Apps	1025
22.4.1 Reverse Engineering of Apps	1025
22.4.2 Automated Vulnerability Analysis of Mobile Applications	1031
22.5 Protective Measures for Android and iOS	1036
22.5.1 Avoid Rooting or Jailbreaking	1036
22.5.2 Update Operating Systems and Apps	1037
22.5.3 Device Encryption	1038
22.5.4 Antitheft Protection and Activation Lock	1038
22.5.5 Lock Screen	1039
22.5.6 Antivirus Apps	1041
22.5.7 Two-Factor Authentication	1042
22.5.8 Critical Review of Permissions	1044
22.5.9 Installing Apps from Alternative App Stores	1045
22.5.10 Using VPN Connections	1046
22.5.11 Related Topic: WebAuthn and FIDO2	1046
22.5.12 Using Android and iOS in the Enterprise	1048
22.6 Apple Supervised Mode and Apple Configurator	1048
22.6.1 Conclusion	1055
22.7 Enterprise Mobility Management	1055
22.7.1 Role and Authorization Management	1057

22.7.2 Device Management	1058
22.7.3 App Management	1059
22.7.4 System Settings	1061
22.7.5 Container Solutions Based on the Example of Android Enterprise	1062
22.7.6 Tracking Managed Devices	1062
22.7.7 Reporting	1063
22.7.8 Conclusion	1064
23 Internet of Things Security	1065
23.1 What Is the Internet of Things?	1065
23.2 Finding IoT Vulnerabilities	1067
23.2.1 Shodan Search Engine for Publicly Accessible IoT Devices	1067
23.2.2 Using Shodan	1068
23.2.3 For Professionals: Filtering Using Search Commands	1069
23.2.4 Printer Exploitation Toolkit	1071
23.2.5 RouterSploit	1073
23.2.6 AutoSploit	1077
23.2.7 Consumer Devices as a Gateway	1081
23.2.8 Attacks from the Inside via a Port Scanner	1081
23.2.9 Sample Port Scan of an Entertainment Device	1082
23.2.10 Local Network versus Internet	1083
23.2.11 Incident Scenarios with Cheap IoT Devices	1083
23.2.12 Danger from Network Operator Interfaces	1084
23.3 Securing IoT Devices in Networks	1085
23.4 IoT Protocols and Services	1086
23.4.1 MQ Telemetry Transport	1087
23.4.2 Installing an MQTT Broker	1089
23.4.3 MQTT Example	1091
23.4.4 \$SYS Topic Tree	1092
23.4.5 Securing the Mosquitto MQTT Broker	1094
23.5 Wireless IoT Technologies	1097
23.5.1 6LoWPAN	1098
23.5.2 Zigbee	1098
23.5.3 LoRaWAN	1099
23.5.4 NFC and RFID	1100
23.5.5 NFC Hacking	1101
23.6 IoT from the Developer’s Perspective	1102
23.6.1 Servers for IoT Operation	1103

23.6.2	Embedded Linux, Android, or Windows IoT Devices	1104
23.6.3	Embedded Devices and Controllers without Classic Operating Systems	1105
23.7	Programming Languages for Embedded Controllers	1107
23.7.1	C	1107
23.7.2	C++	1108
23.7.3	Lua	1108
23.8	Rules for Secure IoT Programming	1109
23.8.1	Processes as Simple as Possible	1110
23.8.2	Short, Testable Functions	1111
23.8.3	Transfer Values Must Be Checked in Their Entirety	1112
23.8.4	Returning Error Codes	1113
23.8.5	Fixed Boundaries in Loops	1115
23.8.6	No Dynamic Memory Allocation (or as Little as Possible)	1115
23.8.7	Make Dimensioning Buffers or Arrays Sufficiently Large	1116
23.8.8	Always Pass Buffer and Array Sizes	1116
23.8.9	Use Caution with Function Pointers	1117
23.8.10	Enabling Compiler Warnings	1118
23.8.11	String Copy for Few Resources	1118
23.8.12	Using Libraries	1119
The Authors		1121
Index		1123

Index

/etc/nsswitch file	747
/etc/sysctl.conf	705
2FA (two-factor authentication)	69, 275, 665
<i>Authy app</i>	670
<i>GitHub</i>	961, 1042
3rd-party app store	1013
6LoWPAN (wireless IoT protocol)	1098
802.11 standard (WiFi)	307
802.1X standard (EAP)	508, 509

A

aa-enforce command	701
aa-status command	700
Access point	1004
Acion (Windows Event Viewer)	572
ACL (Access Control List)	583, 750, 753
<i>Active Directory</i>	583
Activation lock	1039
Active Directory	579
<i>Access Control List</i>	583
<i>audit</i>	639
<i>authentication</i>	603
<i>configuration partition</i>	581
<i>database</i>	579
<i>Default Security Descriptor</i>	583
<i>domain</i>	580
<i>DSInternals</i>	629
<i>group policy</i>	596
<i>join</i>	746
<i>manipulating the database</i>	592
<i>mimikatz</i>	617
<i>pass-the-hash attack</i>	612
<i>Samba</i>	746
<i>schema partition</i>	580
<i>securing</i>	631
<i>standard rights</i>	588
<i>subdomain</i>	580
<i>test environment</i>	110
Add-MpPreferences-Cmdlet	564
Administrative share	
<i>Samba</i>	752
Admin share	
<i>Windows</i>	751
ADSI editor	589
Adversary-in-the-middle attack	527
AFH map (Bluetooth)	340
aircrack-ng command	310

airmon-ng command	309
airodump-ng command	309, 311
Alive test (OpenVAS)	172
AlmaLinux	650
AMQP (Advanced Message Queueing Protocol)	1087
<i>IoT server</i>	1103
Analysis	
<i>apps</i>	1025, 1031
<i>binary files</i>	869
<i>web application</i>	837
<i>web apps</i>	212, 855
Android	997
<i>Android Enterprise</i>	1055, 1062
Angular	835
Anomaly detection	777
Antispam	
<i>Linux/Postfix</i>	722
Anti-theft protection	
<i>mobile security</i>	1038
Antivirus	
<i>Android</i>	1041
<i>Linux/Postfix</i>	722
<i>Windows</i>	557
<i>Windows, outsmarting</i>	499
Apache	706
<i>disabling modules</i>	861
<i>key</i>	710
<i>mod-ssl</i>	709
<i>SSL</i>	709, 710
<i>SSL configuration</i>	709, 863
<i>TLS</i>	709
APFS (file system, macOS)	236
apksigner	1018
apktool	1028
AppArmor	699
<i>audit</i>	703
<i>tunables</i>	703
Apple	97
<i>Apple Configurator</i>	1048
<i>supervised mode</i>	1048
Apple bleee	346
App-MpPreference cmdlet	613
apt command	104
Arduino (Digispark)	367
ARMS (Azure Rights Management Services)	993

ARP protocol	465, 507	AWS (S3) (Cont.)	
<i>arp command</i>	528	<i>key</i>	937
<i>ARP request/reply</i>	527	AWSBucketDump command	942
<i>arp-scan command</i>	100, 508	aws command	938
<i>arpspoof command</i>	529	Azure	
<i>netdiscover command</i>	466	<i>Azure Active Directory application</i>	
ASLR (Address Space Layout		<i>integration</i>	958
Randomization)	903	<i>Azure Active Directory Connect</i>	956
<i>bypassing</i>	907	<i>Azure Active Directory user</i>	
ATA (advanced threat analytics)	644	<i>management</i>	957
Atlas (Mongo)	1104	<i>Identity Protection</i>	975
Atmel IoT controllers	1105	<i>Message Queue Telemetry Transport</i>	1104
ATP (Advanced Threat Protection)		<i>Privileged Identities</i>	976, 977, 979, 980
<i>Detonation Chamber</i>	983	Azure Active Directory	953
<i>Dynamic Delivery</i>	983		
<i>file attachments</i>	986		
<i>links</i>	989		
Attack			
<i>on SSL/TLS connections</i>	533		
Attacker type	40, 429		
ATTACKMODE command (Bash Bunny)	384		
Attack vector	49		
Audio recording	1006		
Audit			
<i>Active Directory</i>	639		
<i>AppArmor logging</i>	703		
<i>lynis tool</i>	729		
<i>Microsoft 365</i>	995		
<i>Samba</i>	758		
Authentication	65, 806		
<i>Active Directory</i>	579, 603		
<i>attacks</i>	612		
<i>Authentication Service (AS)</i>	604		
<i>Kerberos</i>	603		
<i>NTLM and LM</i>	610		
<i>web applications</i>	805		
Authentication policy (Windows)	636		
Authentication Service (Kerberos)	604		
Authentication silo (Windows)	636		
Authorization	65, 1004		
<i>abuse (mobile security)</i>	1006		
<i>mobile security</i>	998		
<i>web applications</i>	805		
authorized_keys file	663		
Authy app	670		
Autopsy	290		
AutoSploit toolkit	1077		
AWS (S3)			
2FA	937		
<i>Bucket</i>	935		
<i>bucket-finder command</i>	942		
<i>bucket-stream command</i>	942		

B

back command (Metasploit)	183		
Backdoor	42		
<i>Android/iOS</i>	1014		
<i>setting up with Bash Bunny</i>	387		
<i>setting up with Digispark</i>	371		
<i>vsftpd (Linux)</i>	183		
Backup			
<i>as a security risk, Active Directory</i>	595		
<i>as a security risk, backup scripts</i>	510		
<i>as a security risk (mobile security)</i>	1009		
Backup as a security risk			
<i>Active Directory</i>	595		
<i>backup scripts</i>	510		
Bad character problem	896		
Badlock bug (Samba)	767		
banaction parameter (Fail2ban)	676		
Bash Bunny	375		
<i>ATTACKMODE command</i>	384		
<i>QUACK command</i>	385		
<i>setting up a backdoor</i>	387		
Beamgun project	417		
BEEF framework	813		
bettercap command	529		
BIAS vulnerability	349		
Binary file			
<i>analyzing</i>	869		
bind-address (MySQL)	718		
Biometric procedures	68		
BIOS	228		
BitLocker	246		
<i>access on Linux</i>	249		
<i>Microsoft 365</i>	992		
BlackArch	77		
Black box check	428		
Blade RF device	351		

BLESA vulnerability	349	Burp (Cont.)	
Blind SQL injection	824, 825	<i>proxy</i>	214
BloodHound	526, 628	<i>Repeater</i>	218
BlueBorne vulnerability	349	<i>Scanner</i>	216
BlueHydra	336	BYOD (Bring Your Own Device)	1055
blue hydra command	336, 345	<i>mobile security</i>	1055
bluelog command	337		
blueranger command	337		
Blue team	425		
Bluetooth	332		
<i>AFH map</i>	340		
<i>analysis adapter</i>	339		
<i>attacks</i>	347		
<i>device discovery</i>	334		
<i>hiding devices</i>	339		
<i>key</i>	333		
<i>low energy (LE)</i>	343, 1066		
<i>low energy communication</i>	344		
<i>scanner</i>	337		
<i>service discovery</i>	340		
Boolean blind attack	843		
Boot options	706		
Boot process	228		
Botnet	43		
Breach detection gap	55		
Bring Your Own Key (Microsoft 365)	992		
Brute-force attack			
<i>password cracking</i>	259		
<i>Samba</i>	760		
<i>smartphones</i>	999		
BSides	187		
btcrack command	347		
btmon command	338		
btscanner command	335		
Bucket (AWS S3)	935		
<i>bucket-finder command</i>	942		
<i>bucket-stream command</i>	942		
Buffer overflow	873, 884		
<i>protective mechanisms</i>	903		
<i>safe coding</i>	914		
Bug	1006		
Bug bounty	428		
<i>program</i>	61, 923		
Bugcrowd	924		
Bugdoor	42		
bulk-extractor	300		
Bunny Script (programming language)	384		
Burp	212, 1025		
<i>BApp Store</i>	218		
<i>example</i>	855		
<i>Extensions</i>	218		
<i>Intruder</i>	217		

C

C/C++ (IoT development)	1107, 1108		
C2 (command-and-control)	542		
C2 server	219		
cadaver command	129		
Cain and Abel	544		
Canary	904		
canonical-livepatch command	660		
CAPI2 log (Windows)	570		
Captive portal (intrusion detection)	784		
Capture the Flag	40		
CDP (Commissioned Data Processing)	995		
CentOS	652		
<i>disabling IPv6</i>	738		
<i>Samba installation</i>	737		
CEO fraud attack	481		
certbot command	708		
Certificate			
<i>HTTPS</i>	707		
cewl command	262, 480		
CGI exploit	185		
Changing the icon of an EXE file	502		
Chaos Computer Club (CCC)	76		
chcon command	695		
Checkm8	1036		
chkrootkit command	727		
chntpw command	240		
<i>activating an administrator account</i>	242		
<i>resetting the Windows password</i>	240		
chroot command	244, 861		
CLEO (corporate liable employee			
owned)	1055		
Click-jacking attack	830		
<i>preventing</i>	862		
Cloud	931		
<i>Amazon S3</i>	935		
<i>Cloud AMQP</i>	1104		
<i>encryption</i>	939, 948, 993		
<i>encryption gateway</i>	993		
<i>Microsoft 365</i>	953		
<i>Nextcloud</i>	943		
<i>securing</i>	931		
CMS (content management system)	836		

CoAP (Constrained Application Protocol) 1087

IoT server 1103

Cobalt Strike 224

Code analysis 874

example 884

IoT 1118

web applications 868

Command-and-control 542

Command-and-control server 219

Command injection 828

Common vulnerabilities and exposures (CVE) 62

Common vulnerability scoring system (CVSS) 62, 172

Computer Emergency Response Team (CERT) 76

Conditional access

Microsoft 365 969

Configuration partition (Active Directory) 581

Cookie

stack 904

Cookie protection 863

COPE (corporate owned personally enabled) 1055

Core Impact scanner 175

Core Server (Windows) 631

Covenant 224

CPU vulnerability 44

Meltdown 915

Spectre 915

Cracker 40

Cracking 42

crackle command 348

CrackMapExec 525

Credential Manager (Windows) 233

Critical infrastructure 74

Crunch 400

crunch command 268, 480

Cryptography 1004

Crypto scam 53

CSRF (Cross-Site Request Forgery) 815

CSS

reflected cross-site scripting 813

stored cross-site scripting 813

Cuda Cracker 263

Curses (Snort) 795

Cyber-observable Object 776

Cybersecurity Act 74

Cyberwarfare 54

CYOD (Choose Your Own Device) 1055

D

DaRT (Windows tool) 238

Data, personal 74

Data breach 272

Data economy (mobile security) 1004

Data leak 449

Data protection laws 74

Data storage 1004

DAV protocol 129

db-nmap command 475

DC (domain controller) 579

dc3dd 285

DCSync 521

Deathstar 628

Debian 104, 650

disabling IPv6 742

Samba 741

Debugger 889

Default Security Descriptor (Active Directory) 583

Defender (Windows) 490, 498, 561

PowerShell cmdlets 563

setting up exceptions 613

Defender for Identity 647

Delegation token 520

Denial-of-service attack 43

DEP (Data Execution Prevention) 905

Deserialization 835

Detection time span 55

DHCP-NAC 508

Dictionary attack 142, 262

dig command 153, 463, 465

Digispark 367

Digital forensics 279

Directory listing (web security) 860

Directory traversal 845

web security 816

Disabling netbios (smb.conf file) 744

Disk imager program 78

dislocker package (Kali Linux) 249

Distributed denial-of-service attack 43

DLP (Data Loss Prevention) 993

DMARC 724

dm-crypt module 253

dnf-automatic package 656

DNS (Domain Name System) 151, 463

brute forcing 464

determining information 151

zone transfer 465

dnsrecon command 464

dnstwist command 210

Docker in Kali Linux 135

DocuSign 958

Domain

Active Directory 580

Domain authentication 603

DomainKeys Identified Mail (DKIM) 724

doona command 886

DoS/DDoS attack 43

Dovecot 724

Dragonblood 325

Drupal 836

dsain command 594

DSC (Desired State Configuration) 558

DSInternals 629

Duck Encoder 366

Duckhunter project 417

Ducky (USB hacking device) 360

DuckyScript programming language 360

Dynamics 365 956

E

EAP (Extensible Authentication Protocol) 508

802.1X standard 509

EFI 228

UEFI Secure Boot 229

EFS (Encrypted File System) 251

EIP (instruction pointer) 890

Email

phishing 481

postfix 719

SpamAssassin 722

Embedded system 765

EMET (Enhanced Mitigation Experience Toolkit) 914

EMM (Enterprise Mobility Management) 1055

Emotet 43

Empire

agent 192

launcher 190

listener 189

payload 204

stager 190

Empire Framework 187

example 387

Encryption

Amazon S3 939

BitLocker 246

Buckets (AWS S3) 935

EFS (Windows) 251

FileVault 252

Encryption (Cont.)

hard drive 246

LUKS 253

mobile 1000

NAS device 254

Nextcloud 948

protection against ransomware (Samba) 738

ransomware 43

EPCIP Guideline 74

EPEL package source 657

ESP (stack pointer) 890

Etcher (USB Image Writer) 78

Ethereal tool 154

Ethical hacking 40

ettercap command 529, 533

Event

eventvwr program, Windows 568

Windows 568

Windows, Active Directory 643

Windows, eventvwr program 568

Windows, forwarding 573

Windows, linking to actions 572

Evil Twin attack 322

Excel macro 490

EXE file

changing the icon 502

exiftool 290

Exploit 58, 871

Android and iOS 1014

buffer overflow 873, 884

database 63

format string attack 873

local 61

logic error 872

memory leak 873

off-by-one error 872

race conditions 871

remote 61

Structured Exception Handling (SEH) 899

exploit command (Metasploit) 183

ext4 mount options 741

Extended attributes 750

External security check 419

F

Fail2ban 673

FAST (Flexible Authentication Secure Tunneling) 642

Fast IDentity Online (FIDO) 69

Microsoft 365 967

Fast IDentity Online (FIDO) (Cont.)	
<i>smartphones</i>	1046
fdesetup command (macOS)	252
Federal Data Protection Act	74
File carving	286
File server	735
File system	
<i>changing</i>	235
<i>reading</i>	231
File system encryption	
<i>server</i>	254
File upload (web security)	820
FileVault (macOS)	252
Firewall	679
<i>example</i>	687
<i>firewall-cmd command</i>	689
<i>FirewallD (Linux)</i>	688
<i>for USB devices</i>	413
<i>Samba</i>	760
<i>Windows</i>	564
Firmware	
<i>password (macOS)</i>	245
<i>vulnerability</i>	44
Foca program	445
Foremost	287
Forensic readiness	303
Forensics	55, 279
Format string attack	873
Forum of Incident Response and	
Security Teams (FIRST)	76
FTK (Forensic Tool Kit)	300
Fuzzing	874
<i>example</i>	886
<i>web applications</i>	869
G	
G DATA USB Keyboard Guard	415
GDPR	995
General Data Protection Regulation	
(GDPR)	74, 431
German Federal Office for Security in	
Information Technology (BSI)	75
getent command	748
getsebool command	696
GetSimple CMS	474
Get-WindowsFeature	632
GhostCrypt (encryption software)	251
GNU Radio	349
Golden ticket	624
<i>detecting</i>	646
Golden ticket attack	522
Google Authenticator	275, 665
Google Play Crawler	1025
Google Play Downloader	1025
Google Play Protect	1041
GoPhish program	483
GoPhish toolkit	483
GPC (Group Policy Container)	596
gpedit module (Windows)	246, 545
gpg command	79
GPS tracking	1006
GPT (Group Policy Template)	596
Gray box check	429
Greenbone OpenVAS	166
Grey box check	429
Greylisting (Postfix)	723
Group management, Windows	546
Group policy	544, 636
<i>Active Directory</i>	596
Group Policy Caching	603
GRUB	706
GSAD service (OpenVAS)	169
gvm-start	169
H	
Hacker article	72
HackerOne	924
Hacking	39
<i>tools</i>	137
Hacking device	46
<i>Bluetooth</i>	339
<i>Software-Defined Radio</i>	351
<i>USB devices</i>	359
Hacking devices, WiFi	324
Hack RF project	351
Half-open scan	467
Ham-it-up converter	351
handler exploit	497
Hard drive	
<i>data access</i>	236
<i>encrypting</i>	246
<i>encrypting (Linux)</i>	253
<i>encrypting (macOS)</i>	252
<i>encrypting (Windows)</i>	246
<i>external</i>	236
Hardware authentication	68
Hardware vulnerability	44
Hash code	67, 256
<i>collisions</i>	256
<i>cracking</i>	263
<i>hashcat command</i>	265
<i>hashdump command</i>	523

Hashkiller	630
Haveibeenpwned	449
hciconfig command	334
hcidump command	338
hcitool command	334, 345
Heap	874
<i>protection against heap spraying</i>	907
<i>spraying</i>	901
Heap spraying	901
<i>protection against</i>	907
Heimdal Kerberos	743
Hibernation file	235
hide unreadable (Samba option)	754
HID hacking gadget	359
HIDS (host-based IDS)	778
Honeypot	110
Host	
<i>detecting active hosts</i>	465
hostap-wpe package	322
HostKeyAlgorithms	114
Host-only network (VirtualBox)	112
hosts command (Metasploit)	181
Hotspot	1004
HSTS (HTTP Strict Transport Security)	864
HTML, injection	808
HTTPS	
<i>Apache configuration</i>	706, 863
<i>certificate</i>	707
<i>checking the configuration</i>	148
Human Interface Device (USB)	359
hydra command	142, 269
<i>example</i>	478
Hyperscan	798
Hyper-V	91, 634
<i>Shielded VM</i>	634
<i>VirtualBox</i>	84
I	
IDS (Intrusion Detection System)	775
<i>inline</i>	785, 790, 801
<i>passive</i>	790
iFrame (click jacking)	830
Image Writer	78
Immunity debugger	889
Impacket library	536
Impersonation token	520
incognito extension (Metasploit)	520
Industry 4.0	1066
Information gathering	463, 838
Infrastructure	
<i>critical</i>	74
Injection, HTML	808
Input validation	865
Inspectrum (SDR tool)	354
Installation log (Windows)	570
Instruction pointer	890
Intel Management Engine	44
Intercepting proxy	214, 1025
Interfaces (smb.conf file)	744
Intrusion detection	55
<i>captive portal</i>	784
<i>Linux</i>	726
<i>prevention</i>	783
<i>walled garden</i>	784
IoC (Indicator of Compromise)	776
IoE (Internet of Everything)	1067
iOS	997
IoT	1065
<i>6LoWPAN</i>	1098
<i>C/C++</i>	1107, 1108
<i>code analysis</i>	1118
<i>controllers</i>	1105
<i>developer principles</i>	1109
<i>development</i>	1102
<i>LoRaWAN</i>	1099
<i>Lua</i>	1108
<i>MQTT</i>	1087, 1103
<i>NFC</i>	1100
<i>protocols</i>	1086
<i>qpit</i>	1103
<i>securing a device</i>	1085
<i>server implementations</i>	1103
<i>servers</i>	1103
<i>SSL</i>	1094
<i>wireless technologies</i>	1097
<i>XMPP</i>	1087
<i>ZigBee</i>	1098
IoT Device	
<i>securing</i>	1085
IP packet filters	679
IPsec	1046
IPSec settings (Windows Firewall)	567
iptables command	506, 680
<i>blocking outbound traffic</i>	506
<i>Samba</i>	760
IPv6	70
<i>disabling, CentOS</i>	738
<i>disabling, Debian</i>	742
<i>Fail2ban</i>	673
<i>Linux</i>	653
<i>SSH server</i>	665
ISO/OSI layer model	527
ISO 14443 standard	1100

ISO 15693 standard	1100
ISPProtect scanner	730
IT forensics	55, 279
<i>digital traces</i>	281
<i>file carving</i>	286
<i>forensic backup</i>	284
<i>forensic investigation</i>	281
<i>forensic readiness</i>	303
<i>live analysis</i>	300
<i>post mortem</i>	284
<i>system analysis</i>	290
IT Security Act	74
iwconfig command	309
J	
Jailbreak	
<i>Android</i>	1025, 1036
<i>iOS</i>	1025, 1036
JEA (Just Enough Administration)	559
JIT (Just-In-Time Administration)	559
John the Ripper	263, 264
<i>example</i>	479
Join (Samba/Active Directory)	746
Juice Shop	133
K	
KAISER protection method	916
Kali Linux	77
<i>booting a foreign notebook computer</i>	230
<i>default password</i>	83
<i>dislocker package</i>	249
<i>installing OpenVAS</i>	167
<i>internal details</i>	103
<i>live mode</i>	81
<i>SSH</i>	89
<i>UEFI Secure Boot</i>	229
<i>verifying the download</i>	78
<i>WSL variant</i>	93
kali-tweaks	106
KARMA toolkit	323
KASLR mechanism	916
KDC (Key Distribution Center)	603
Kerberos	603, 745
<i>armoring/FAST</i>	642
<i>Authentication Service</i>	604
<i>Long-Term Session Key (LTSK)</i>	606
<i>mimikatz</i>	621
<i>policies</i>	641
<i>Privilege Access Certificate (PAC)</i>	606
<i>reset</i>	639
Kerberos (Cont.)	
<i>Ticket Granting Service</i>	604
<i>Ticket Granting Service, Golden Ticket</i>	624
<i>Ticket Granting Service, Silver Ticket</i>	624
<i>Ticket Granting Service, tickets</i>	604, 605
<i>tickets</i>	605
<i>Windows</i>	604
Kerberos, Login Session Key	606
Kerckhoff's principle	1008
Kernel	
<i>boot options</i>	706
<i>hardening</i>	704
<i>Page Table Isolation (KPTI)</i>	916
Kernel (Linux)	
<i>live patches</i>	659
<i>Meltdown attack</i>	915
<i>updates</i>	658
Key	
<i>Amazon S3</i>	937
<i>Android/iOS</i>	1000
<i>Apache</i>	710
<i>BitLocker</i>	246
<i>Bluetooth</i>	333
<i>Bluetooth/crackle</i>	348
<i>DKIM/Linux</i>	724
<i>gpg command</i>	79, 940
<i>Kerberos</i>	603
<i>openssl command</i>	940
<i>Samba</i>	757
<i>SSH authentication</i>	663
<i>WiFi/KRACK attack</i>	321
<i>WiFi/WEP</i>	310
<i>WiFi/WPA</i>	315
<i>WiFi/WPS</i>	317
Key logger	517
Key logging	42
Keystroke injection attack	360
KillerBee project	1098
King Phisher	212
kirbi file (Kerberos ticket)	626
Koadic	
<i>implants</i>	202
<i>modules</i>	202
<i>obfuscate</i>	201
<i>server</i>	198
<i>stagers</i>	202
<i>zombies</i>	201
kPatch kernel updates	659
Kr00k	45
KRACK (Key Reinstallation Attack)	321
krb5.conf file	745

krb5-user	743
Ksplice kernel updates	659
L	
Landing zone (NOP sled)	902
LAPS (Local Admin Password Solution, Windows password management)	548
<i>LAPS UI</i>	551
Lazarus Group	442
LDAP	579, 611
LDAP Simple Bind	611
Let's Encrypt	707
Leviathan	449
LFI (Local File Inclusion)	817
Liability agreement	431
LibEseDB tool	595
libpam-heimdal	743
Lightweight Directory Access Protocol	611
Linux	649
<i>BitLocker</i>	249
<i>changing the file system</i>	235
<i>detecting rootkits</i>	726
<i>encrypting the hard drive</i>	253
<i>firewall</i>	679
<i>intrusion detection</i>	726
<i>IPv6</i>	653
<i>kernel updates</i>	658
<i>Metasploitable</i>	118
<i>Metasploit installation</i>	177
<i>reading the file system</i>	235
<i>resetting a password</i>	244
<i>Samba</i>	735
<i>securing SSH</i>	661
<i>securing the root server</i>	649
<i>setting up a backdoor with Digispark</i>	371
<i>updates</i>	654
Live analysis (forensics)	300
Live host detection	465
Live mode (Kali Linux)	81
LLMNR multicast message	535
LM authentication protocol	610
Local exploit	61
Local Group Policy Object	598
Lock screen	1039
Log4Shell	45
Logging	
<i>Fail2ban</i>	673
<i>linking Windows to actions</i>	572
<i>SELinux</i>	698
<i>Windows</i>	568
<i>Windows, forwarding</i>	573
Login cracker	142
Login Session Key (Kerberos)	606
logonpasswords (mimikatz)	619
Long-Term Session Key (LTSK, Kerberos) ...	606
Long-Term Support (LTS)	651
Lookup table (MySQL/MariaDB)	716
LoRaWAN (wireless IoT protocol)	1099
LPGO command	598
LSA (Local Security Authority)	618
lsa command (mimikatz)	625
LSAiso function (Windows 10)	519
LSASS.exe program	618
lsblk command	231
lsuf command	733
lsusb command	416
Lua (IoT development)	1108
LUCY (phishing tool)	212
LUKS (Linux encryption)	253
lynis command	729
M	
M1/M2-CPU	97
MAC address	528
<i>arp-scan-Kommando</i>	101
<i>setting (Linux)</i>	508
MAC filter	508
Machine	
<i>Maltego</i>	456
macOS	
<i>backdoor with Bash Bunny</i>	387
<i>FileVault</i>	252
<i>FileVault encryption</i>	252
<i>firmware password</i>	245
<i>Metasploit installation</i>	178
<i>reading the file system</i>	236
<i>resetting the password</i>	245
<i>T2-Chip</i>	236
Mail encryption (Postfix)	720
makepasswd command	270
Malicious software	43
Maltego	
<i>machine</i>	456
<i>program</i>	450
<i>transform</i>	450
Malware	43
<i>Android and iOS</i>	1014
MAM (Mobile Application Management)	1055
Mandatory Access Control (MAC)	694
Man-in-the-middle attack	1004
<i>AD certificates</i>	581

Man-in-the-middle attack (Cont.)	
<i>ettercap</i> command	529
WiFi	323
MariaDB	712
Marshalling	835
mat2	289
Matching options (Snort)	797
MD5 hash code	256
MDM (Mobile Device Management)	973, 1055
ME (Management Engine)	246
Meltdown vulnerability	44, 915
memdump command	918
Memory area	
<i>x86</i> systems	874
Memory isolation	915
Memory leak	873
Message Analyzer	159
Metasploit	65, 176, 472
Android/iOS	1014
<i>back</i> command	183
Community	176
console	180
database	177, 180
<i>db_nmap</i>	469
Digispark example	371
<i>exploit</i> command	183
Express	176
Framework	176
handler <i>exploit</i>	497
<i>hosts</i> command	181
<i>incognito</i> extension	520
<i>msfvenom</i> command	495
<i>nmap</i> command	181
<i>outsmarting the virus protection</i>	491, 499
<i>phishing</i> example	495
<i>run</i> command	183
<i>services</i> command	181
<i>show</i> command	183
<i>updates</i>	180
<i>use</i> command	183
Metasploitable	109
<i>version 2 (Ubuntu)</i>	110
<i>version 3 (Windows)</i>	116, 123
Meterpreter	65, 185
Android	1016
iOS	1016
<i>phishing</i> attack	495
Microsoft	
Advanced Threat Analytics (ATA)	644
Intune	956
Message Analyzer	159
Microsoft (Cont.)	
Network Monitor	159
Online ID	954
Planner	953
Teams	953
Trust Center	995
Microsoft 365	953
AAD	954
AAD Connect	956
<i>access governance (data centers)</i>	994
<i>access management</i>	954
App Launcher	958
ATP	982
Bring Your Own Key (BYOK)	992
conditional access	969
data centers (security)	992
encryption (data centers)	992
Exchange Online	953
FIDO	967
identities	954
Identity Protection	975
Live ID	954
Microsoft Account (MSA)	954
Microsoft Online ID	954
Microsoft Planner	953
Mobile Device Management (MDM)	973
plans	953
privileged identities	976
security assessment	960
Microsoft Defender for Office 365	982
Mimikatz, logonpasswords	619
mimikatz program	518
mimikatz tool	617
<i>golden ticket</i>	625
Kerberos	621
PowerSploit	623
SekurlSA	618
usage example	518
MiniShare sample analysis	884
min protocol (smb.conf file)	744
MIT-Kerberos	743
MITRE ATT&CK Framework	428, 441
<i>command-and-control</i>	542
<i>initial access</i>	459, 472
<i>privilege escalation</i>	509
<i>reconnaissance</i>	441, 463
Tactics, TA0003	441
Tactics, TA0007	524
Tactics, TA0011	542
Techniques, T1003	518
Techniques, T1003.006	521
Techniques, T1056.001	517

MITRE ATT&CK Framework (Cont.)		Multipattern search engine	793
Techniques, T1068	510	MySQL	712
Techniques, T1078	478, 510		
Techniques, T1091	504	N	
Techniques, T1110	524		
Techniques, T1133	478	NAC (Network Access Control)	506
Techniques, T1136.001	441	802.1X	507
Techniques, T1190	472	Nano Server (Windows)	633
Techniques, T1200	506	NAS device	765
Techniques, T1548.002	512	encryption	254
Techniques, T1552	517	National Institute of Standards and	
Techniques, T1555	517	Technology (NIST)	75, 434
Techniques, T1557	527	National Vulnerability Database (NVD)	63
Techniques, T1558	522	nc command	163
Techniques, T1558.001	522	example	539, 820, 830
Techniques, T1566	481	ncrack command	146
Techniques, T1590.002	463	needs-restarting command	655
Techniques, T1591	442	Nessus	166
Techniques, T1592	445	Netcat program (nc)	163, 820
Techniques, T1594	442	example	820, 830
Techniques, T1594.001	449	net command	544
Techniques, T1594.002	447	netdiscover command	466
Techniques, T1594.003	447	netstat command	732
Techniques, T1595.001	465	Network communication	1004
Techniques, T1595.002	470	Network hash	534
mkpasswd command	270	Network login cracker	142
Mobile encryption	1000	Network Monitor (Microsoft)	159
Mobile security	997	Network scanner	100, 138
Bring Your Own Device	1055	Network service	
BYOD	1055	identifying	467
MobSF (Mobile Security Framework)	1031	Nexpose scanner	175
ModSecurity (Web Application Firewall)	866	Nextcloud, encryption	948
mod-ssl (Apache module)	709	NFC (wireless IoT technology)	1100
Monitoring (Windows)	643	nft	680
mount-command	232	example	687
Windows is hibernated	235	nftables	680
Windows snapshots	594	ngrep command	162
mount-options (Samba)	738	NIDS (network-based IDS)	779
MPsSvc service (Windows Firewall)	564	NIST SP 800-115	434
MQTT (Message Queue Telemetry		nmap command	138
Transport)	1068	alternatives	141, 472
IoT protocol	1087	example	466
IoT server	1103	Metasploit	469
SSL	1094	NSE	768
msfconsole command	177, 180	penetration testing on networks	467
msfdb	177	Samba	760, 768
msfupdate	180	SambaCry test	766
msfvenom command	371, 495, 499, 537, 895, 1014	scripting engine	470
Multi-factor authentication	275, 961	smb2-capabilities	771
Multilevel security (SELinux)	696	smb-os-discovery	771
		ssh-brute	772
		Nmap command (Metasploit), db-nmap	181

Nmap	142
Non-Disclosure Agreement (NDA)	431
NOP sled	902
Notebook	
<i>BIOS/EFI settings</i>	228
<i>changing the Windows file system</i>	235
<i>reading the Windows file system</i>	231
<i>resetting the Windows password</i>	237
NSE	
nmap command	768
NSS (Nameservice Switching Daemon)	747
nsswitch file	747
NTDS.DIT file	579
ntdsutil command	593
NTDSXtract tool	595
NTLM authentication protocol	610
ntlmrelayx command	537, 539
nullok option (PAM/Linux)	669
NVT (Network Vulnerability Test)	166
<hr/>	
O	
<hr/>	
Obfuscation	1025
Object serialization	835
Off-by-One error	872
Office macro	490
Offline hacking	227
Offset calculation (program analysis)	891
OneDrive for Business Online	974
One-time password (OTP)	670
One-time secret	270
ÖNORM A 7700	434
Open Security Foundation (OSF)	64
Open Source Vulnerability	
Database (OSVDB)	64
openssl command	708, 940
OpenVAS	166
alive test	172
GSAD service	169
gvmcmd command	168
gvm-feed-update command	169
gvm-start command	169
severity	172
Open Web Application Security	
Project (OWASP)	433, 859
Oracle Linux	650
Order processing contract	431
OSINT (Open Source Intelligence)	420, 442
analysis	420
osmocom-fft command	355
Out-of-order execution	915
<hr/>	
P	
<hr/>	
P4wnP1	
basic principles	396
CLI	399
HID scripts	398
installation	397
trigger	404
Packer tool	118
Packet filters	679
Parrot OS	77
parted command	231
Partition	
Active Directory	580
Samba installation	736
Passkey (Apple)	70
Pass-the-hash attack	524, 612
protective measures	639
Pass-the-ticket attack	612
protective measures	639
passwd command	244
Password	65, 255
attacks	478
cracker	142
cracking	255, 259, 263
creating a cracking word listing	480
default passwords	271
FIDO2	69
generating random passwords	270
hash code	255, 523
hydra command	142
in memory	518
LAPS configuration	548
leak	449
Linux password reset	244
macOS password reset	245
password lists	144
properties (Windows)	545
protecting (Windows)	545
resetting the Windows password	237
reuse	272
rules	66
secure handling	276
security	478
spraying	262
two-factor authentication	275
WiFi	320
Windows password reset	244

Patch day	1001	Privilege escalation	509, 850, 1036
Payload	65, 176, 185	<i>exploit suggerer</i>	510
<i>encoder</i>	499	<i>preventing through the tier model</i>	635
pcap library	159	<i>UAC</i>	512, 515
PCRE (Snort)	798	Process context (SELinux)	695
PCUnlocker program	238	Processor	
Penetration test	41, 425	<i>vulnerability</i>	44
Penetration Testing Execution		Processor vulnerability	44
Standard (PTES)	434	<i>Meltdown</i>	915
PentestBox	77	<i>Spectre</i>	915
Permissions		Program analysis, offset calculation	891
<i>Samba</i>	753	Program execution, x86 systems	874
Personal data	74	Protection	
Phishing	66	<i>Active Directory</i>	631
<i>defense</i>	503	<i>hard drives</i>	246
<i>Gophish toolkit</i>	483	<i>Linux</i>	649
<i>sending emails</i>	481	<i>SSH</i>	661
<i>Social-Engineer Toolkit (SET)</i>	206	<i>web applications</i>	859
<i>USB phishing</i>	504	Proxy	
PHP/CGI exploit	185	<i>Netcat</i>	165
Pickling	835	<i>proxy listener (Burp)</i>	214
PIN-cracking brute-force attack	347	<i>Zed Application Proxy</i>	219
Port knocking	847	proxychains command	541
Port scanner	138	psexec module	524
POSIX-ACL	750	PTA (Permission-to-Attack)	431, 459
Post exploitation		PTT (Platform Trust Technology)	246
<i>Android</i>	1016	PUOCE (Private Use of Company	
<i>Empire Framework</i>	187	Equipment)	1055
<i>iOS</i>	1016	Purple teaming	427
<i>ScareCrow</i>	499	pwgen command	270
Postfix	719	Pwn2Own event	61
<i>antispam</i>	722	Pwnagotchi	325
<i>antivirus</i>	722	pwsh	107
<i>DKIM</i>	724	Python 2	105
<i>greylisting</i>	723	Python EmPyre	187
<i>spam defense</i>	722		
<i>SPF (Sender Policy Framework)</i>	724	Q	
<i>virus protection</i>	722		
Post-mortem investigation	284	QoD (Quality of Detection)	172
PowerShell		qpit (IoT server)	1103
<i>Defender administration</i>	563	QR code generator	211
<i>event log administration</i>	576	QUACK command (Bash Bunny)	385
<i>Get-ACL and Set-ACL cmdlets</i>	583	Quakbot	494
<i>Get-ADObject cmdlet</i>	588	Quantum Insert	781
<i>PowerSploit tools</i>	602		
<i>send-MailMessage</i>	572	R	
PowerShell (Kali Linux)	107		
PowerSploit tools	623	Race condition	871
PRET (Printer Exploitation Toolkit)	1071	Rainbow table	260
Prevention (intrusion detection)	783	Ransomware	43
Privilege Access Certificate (PAC,		Raspberry Pi Zero W	396
Kerberos)	606	realm (Samba)	744

reaver command	319	RSS Rapid Response	778
reboot-required file	655	Rubber Ducky (USB hacking device)	360
Red Hat	650	Rules, Snort	793
<i>firewall</i>	688	run command (Metasploit)	183
<i>kernel live patches</i>	660		
Red teaming	425	S	
Reflected cross-site scripting	813		
REG file	598	S/MIME (Microsoft 365)	993
Registry	744	SafeSEH protection (Visual Studio)	906
<i>attack (Windows)</i>	557	Samba	735
<i>Samba</i>	751, 755, 757	<i>Active Directory</i>	746
Regular expressions, Snort	798	<i>administrative share</i>	752
Remote deletion (mobile security)	1038	<i>audit functions</i>	758
Remote exploit	61	<i>Badlock bug</i>	767
remove-hiberfile-Option	235	<i>brute-force attack</i>	760
Remove-MpPreferences-Cmdlet	564	<i>CentOS</i>	737
Reporting	434	<i>Debian</i>	741
Repository, Samba	736	<i>encryption</i>	738
resetpassword command	245	<i>firewall</i>	760
Resource Hacker tool	502	<i>hide unreadable</i>	754
responder command	535, 538	<i>iptables command</i>	760
Responsible disclosure	60	<i>IPv6</i>	737
REST (Representational State Transfer)	804	<i>join</i>	746
restorecon command	695	<i>key</i>	757
Reverse engineering	1025	<i>mount options</i>	738
reverse-tcp-Payload	496	<i>nmap command</i>	760, 768
RFCAT library	352	<i>permissions</i>	753
RFI (Remote File Inclusion)	819	<i>protection against ransomware</i>	738
RFID standard	1100	<i>realm</i>	744
RHEL		<i>registry</i>	751, 755, 757
<i>automatic updates</i>	656	<i>repository</i>	736
<i>firewall</i>	688	<i>rid backend</i>	744
rhosts configuration file	115	<i>RPC error</i>	767
RID 500 account	544	<i>SambaCry vulnerability</i>	766
rid backend (Samba)	744	<i>SerNet</i>	737
rkunter command	728	<i>shares</i>	750
rlogin command	115	<i>special authorizations</i>	751
Rocky Linux	650	<i>tcpdump command</i>	760
Role, Windows	632	<i>TDB database</i>	755
Rolling release	104	<i>user share</i>	752
Rooting		<i>winbind</i>	744
<i>Android</i>	1025, 1036	SAM file (Windows)	237, 239, 240, 544
<i>iOS</i>	1025, 1036	Sample DC9 security environment	837
Rootkit	65	Sandboxing	
<i>detection (Linux)</i>	726	<i>mobile security</i>	998
Root server	649	Sandworm Team	442
ROP (Return-Oriented Programming)	908	SCADA (Supervisory Control And Data	
Router hack (Deutsche Telekom)	1084	Acquisition)	1066
Router Keygen command	271	ScareCrow	499
RouterSploit framework	1073	Schema partition (Active Directory)	580
RPC error (Samba)	767	Script kiddy	40, 430
rpm command	731		

SDDL (Security Descriptor Definition		Set-NetFirewallProfile command,	
Language)	589	blocking outbound traffic	506
SDProp (Security Descriptor Propagator) ...	585	setoolkit command	206
sdptool command	340	settroubleshoot-server package	698
SDR (Software-Defined Radio)	349	setsebool command	696
<i>hacking devices</i>	351	Severity (OpenVAS)	172
sealert command	698	SHA (Secure Hash Algorithm)	256
searchsploit command	472	<i>MySQL/MariaDB</i>	716
Seccubus	175	<i>SHA-1</i>	256
Security		<i>SHA-2</i>	258
<i>Amazon S3</i>	935	Share	
<i>cloud</i>	931	<i>Samba server</i>	750
<i>IoT devices</i>	1085	SharePoint Online	974
<i>Microsoft 365</i>	953	Shielded VM (Hyper-V)	634
<i>ownCloud</i>	943	Shodan search engine	1067
<i>Samba</i>	735	<i>SambaCry vulnerability</i>	766
<i>Windows</i>	543	show command (Metasploit)	183
Security Account Manager	237, 240, 544	Side-channel attack	915
Security assessment (Microsoft 365)	960	Silver ticket	624
Security check, external	419	skip-networking (MySQL/MariaDB)	718
Security Compliance Toolkit	560	Sliver	219
Security context (SELinux)	694	Slow Loris attack	863
Security log (Windows)	570	smart-hashdump command	523
Security token	670	Smart Lock (mobile security)	1040
SEH (Structured Exception Handling)	899	Smartphone	997
sekurlsa module (mimikatz)	618	<i>Bring Your Own Device</i>	1055
SELinux	693	<i>BYOD</i>	1055
<i>Google Authenticator</i>	668	<i>finding</i>	1038
<i>logging</i>	698	smb.conf file	744, 757
<i>multilevel security</i>	696	<i>changing to the registry</i>	755
<i>Password reset</i>	244	<i>disabling netbios</i>	744
<i>process context</i>	695	<i>interfaces</i>	744
<i>security context</i>	694	<i>min protocol</i>	744
<i>targeted policy</i>	696	<i>vfs objects</i>	744
Semi-Annual Channel (SAC, Windows)	631	<i>winbind</i>	748
Sender Policy Framework (SPF)	724	smbclient command	748
SerNet (Samba packages)	737	SMB log	736, 744
Server		SMB protocol	
<i>file system encryption</i>	254	<i>authentication via NTLM</i>	536
<i>Linux protection</i>	649	<i>relaying</i>	536, 540
<i>securing Windows</i>	543	<i>SMB daemon</i>	765
<i>signature (HTTP)</i>	860	SMB scan	469
<i>timeout (HTTP)</i>	863	smtp parameter (Postfix)	720
services command (Metasploit)	181	Snapshot, Windows	593
Session (web security)	806	Snort	787
<i>fixation</i>	815	<i>curses</i>	795
<i>hijacking</i>	807	<i>matching options</i>	797
<i>management</i>	806	<i>PCRE</i>	798
SET (Social-Engineer Toolkit)	205	<i>regular expressions</i>	798
<i>Social engineering</i>	52, 205	<i>rules</i>	793
setenforce command	699	SOAP (Simple Object Access Protocol)	804
sethc program	553	socat command	166

Software exploitation	871	STIX (Structured Threat Information	
SpamAssassin	722	Expression)	776
Spam defense, Linux/Postfix	722	stkeys command	320
Spear phishing	1042	Stored cross-site scripting	813
Special authorizations (Samba)	751	Structured Exception Handling	
spectool program	342	Overwrite Protection	906
Spectre vulnerability	44, 915	Stuxnet	1066
SpeedPhish Framework	212	Subdomain, Active Directory	580
SPN (Service Principal Name)	609	sudo	103
Spraying, heap spraying	901	<i>Kali Linux</i>	88
Spyware	1024	Supervised mode (Apple)	1048
SQL (SQL injection)	821	SUSE	650
<i>blind SQL injection</i>	824, 825	SYN scan	467
SQL injection	840	sysctl	704
sqlmap	842	System log (Windows)	570
sqlmap command	823	SYSVOL folder	597
SSD, encrypting	246		
SSD data access	236	T	
ssh-copy-id command	663	T2 chip (Apple)	236, 252
ssh-dss	114	Tablet	997
ssh-keygen command	663	targeted policy (SELinux)	696
ssh-rsa	114	Task Scheduler (Windows)	572
SSH server		Task Scheduler program (Windows)	572
<i>authentication with keys</i>	663	taskset command	917
<i>configuration file</i>	661	TCP Connect scan	468
<i>firewall script</i>	760	tcpdump command	159
<i>Kali Linux</i>	89	<i>Samba</i>	760
<i>securing</i>	661	TCP SYN scan	467
<i>two-factor authentication</i>	665	TDB database (Samba)	755
SSH Server, IPv6	665	Template Injection	835
SSL (Secure Socket Layer)	148, 863	testssl command	149
<i>Apache configuration</i>	709, 710, 863	Theft	
<i>checking the configuration</i>	148	<i>mobile security</i>	1003
<i>interception</i>	1009	<i>mobile security, ant-theft protection</i>	1038
<i>SSLCipherSuite (Apache)</i>	710	Third-party store	1013
<i>SSLPolicy (Apache)</i>	711	Threat Intelligence	776
<i>SSLProtocol (Apache)</i>	710	Threat modeling	867
<i>Virtual Private Network</i>	1046	Ticket Granting Service (Kerberos)	604
SSL configuration	709	<i>golden ticket</i>	624
SSL connection		<i>silver ticket</i>	624
<i>attack</i>	533	<i>tickets</i>	604, 605
SSRF (Server Side Request Forgery)	834	Tier model	635
Stack	874	TLS (Transport Layer Security)	148, 720
<i>canaries</i>	904	<i>Apache configuration</i>	709, 863
<i>cookies</i>	904	<i>checking the configuration</i>	148
<i>pointer</i>	890	<i>interception</i>	1009
Stagefright exploit	1019	<i>Microsoft 365</i>	992
Standard rights (Active Directory)	588	TLS connection attack	533
Starkiller	187	Token	670
STARTTLS (Postfix)	720	TPM (Trusted Platform Module)	246
stat command	288	Tracking, mobile devices	1062

Transform (Maltego)	450	V	
TrueCrypt (encryption software)	251	Vagrant	117
Trust ticket	628	Vault file (Windows)	233
TTP	441	<i>vaultcmd command</i>	233
tunables (AppArmor)	703	<i>VaultPasswordView tool</i>	234
two-factor authentication	275, 961	VDS (Vulnerability Disclosure Program)	924
Two-factor authentication (2FA)	69, 275, 665, 937	VeraCrypt (encryption software)	251
		vfs objects (smb.conf file)	744
		VirtualBox	80
		<i>Hyper-V</i>	84
		<i>installing Kali Linux</i>	84
		<i>network connection</i>	88
		VPN (Virtual Private Network)	
		<i>Always-on VPN</i>	1046
		<i>mobile security</i>	1046
		Vulnerability	44, 59
		<i>analyzing mobile apps</i>	1031
		<i>assessment</i>	424
		<i>database</i>	63
		<i>management</i>	422
		<i>scan</i>	422
		<i>scanner</i>	64
		<i>scan with nmap</i>	470
		<i>scan with OpenVAS</i>	166
		<i>software exploitation</i>	871
		W	
		WAF (Web Application Firewall)	866
		<i>ModSecurity</i>	866
		Walled garden (intrusion detection)	784
		WannaCry bug	766
		wash command	318
		wbinfo command	747
		wce command	614
		wdigest command	518
		Web application	
		<i>architecture</i>	803
		<i>attacks</i>	806
		<i>code analysis</i>	868
		<i>components</i>	804
		<i>fuzzing</i>	869
		<i>inclusion of external content</i>	862
		<i>protection mechanisms</i>	859
		<i>restricting methods</i>	862
		<i>sample analysis</i>	837
		<i>security</i>	434, 803
		<i>security analysis</i>	867
		Web application security (BSI)	434
		WebAuthn	1046
		WebDAV protocol	129

Webproxy (Netcat)	165	Windows (Cont.)	
Web security		<i>Metasploitable</i>	116, 123
<i>directory listing</i>	860	<i>monitoring</i>	643
<i>directory traversal</i>	816	<i>Nano Server</i>	633
<i>file upload</i>	820	<i>password, properties</i>	545
<i>session</i>	806	<i>password protection</i>	545
Web server	706	<i>reading the file system</i>	231
WECUtil tool (Windows)	575	<i>registry</i>	557
weevely command	130	<i>resetting the password</i>	237, 240
WEF (Windows Event Forwarding)	573	<i>role</i>	632
WEP (Wireless Equivalent Privacy)	310	<i>SAM file</i>	544
White box check	429	<i>security log</i>	570
whois command	152	<i>server</i>	543
Whois protocol	461	<i>Server Update Service (WSUS)</i>	633
WiFi	307	<i>sethc program</i>	553
<i>attack</i>	307	<i>system log</i>	570
<i>client attack</i>	323	<i>Task Scheduler</i>	572
<i>default configuration</i>	320	<i>update</i>	559
<i>detecting networks</i>	309	<i>User Account Control</i>	512
<i>infrastructure</i>	308	<i>users and groups</i>	544
<i>password</i>	320	<i>Utilman program</i>	553
<i>Pineapple hacking device</i>	324	<i>vault files</i>	233
winbind	747	<i>WECUtil tool</i>	575
winbind (Samba)	744	Windows Authentication token	520
Windows		Windows Exploit Suggester	510
<i>Active Directory</i>	579	Windows Subsystem for Linux (WSL)	93
<i>admin share</i>	751	WinDump program	159
<i>antivirus</i>	557	Wireshark tool	154
<i>authentication</i>	603	<i>Bluetooth</i>	341, 345
<i>authentication policy</i>	636	<i>example</i>	507
<i>authentication silo</i>	636	WLAN	307
<i>BitLocker</i>	246	WordPress	836
<i>CAPI2 log</i>	570	WPA (Wireless Protected Access)	315
<i>Changing the file system</i>	235	<i>KRACK attack</i>	321
<i>Core Server</i>	631	<i>WPA-2</i>	315
<i>Credential Manager</i>	233	<i>WPA-2 Enterprise</i>	322
<i>Credentials Editor</i>	614	<i>WPA-3</i>	325
<i>Defender</i>	490, 498, 561	WPS (Wireless Protected Setup)	317
<i>encrypting the hard drive</i>	246	WPScan	836
<i>events</i>	568	WSL (Windows Subsystem for Linux)	93
<i>Event Viewer</i>	568		
<i>Firewall</i>	564		
<i>forwarding events</i>	573		
<i>forwarding logging</i>	573		
<i>gpedit module</i>	545		
<i>group management</i>	546		
<i>hardening</i>	558, 634		
<i>hibernation file</i>	235		
<i>installation log</i>	570		
<i>Kerberos</i>	603		
<i>locking USB devices</i>	413, 414		
<i>Metasploit</i>	179		

XPath injection	832	Zenmap	142
XSS (cross-site scripting)	811	Zentyl Linux	651
		Zero-day exploit	60
		Zerodium (exploit market)	61
		ZigBee	
		<i>IoT wireless protocol</i>	1066
		<i>wireless IoT protocol</i>	1098
		zipalign	1018

Y

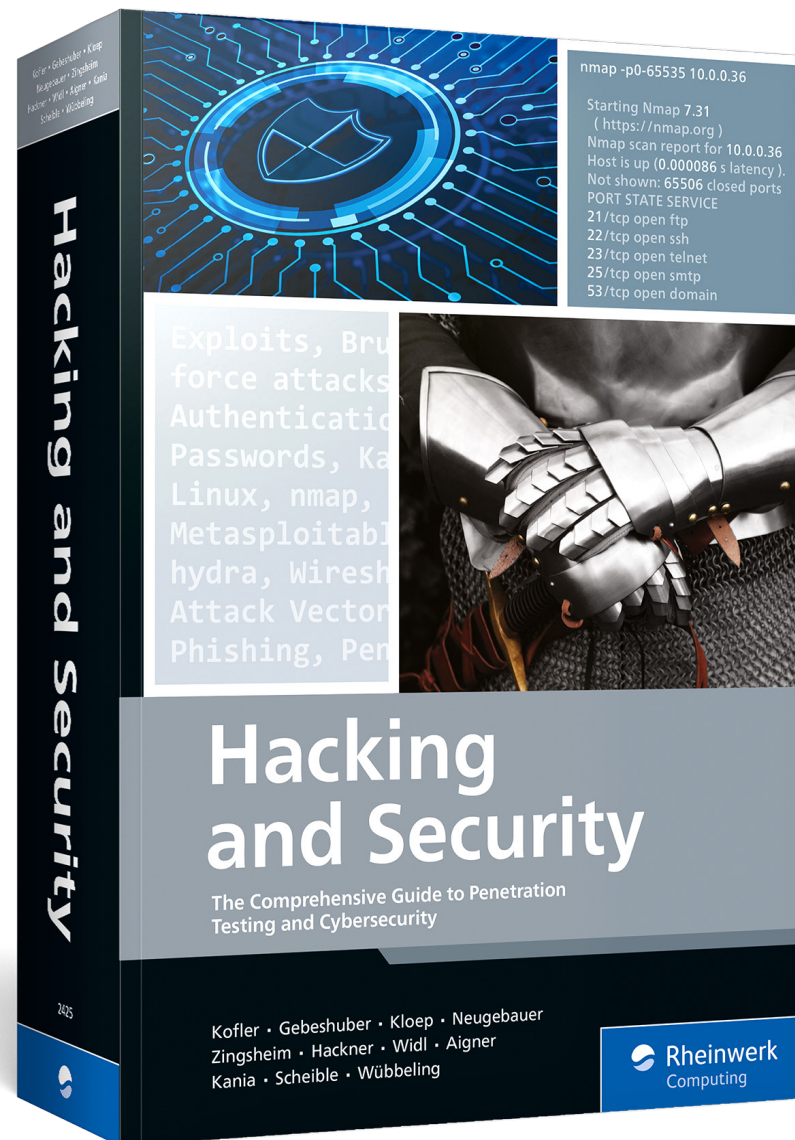
Yard Stick One	351
ysoserial	835
YubiKey	670

Z

Zadig program	350
ZAP (Zed Application Proxy)	219, 1025
Zed Application Proxy (ZAP)	219, 1025

X

X.500 protocol	579
X-Frame-Options	862
xfs mount options	738
xHydra user interface	146
XML Attack	832
<i>XML External Entity Attack (XXE)</i>	833
XMPP (IoT protocol)	1087
XPath Bruter	833



This book was written by a team of security specialists who have decades of experience working as penetration testers, administrators, and developers. Their unique expertise has taught them how hackers operate and they use this insight to teach others how to effectively defend systems against attacks.

- Michael Kofler
- Klaus Gebeshuber
- Peter Kloep
- Frank Neugebauer
- Andrè Zingsheim
- Thomas Hackner
- Markus Widl
- Roland Aigner
- Stefan Kania
- Tobias Scheible
- Matthias Wübbeling

Michael Kofler et al.

Hacking and Security

The Comprehensive Guide to Penetration Testing
and Cybersecurity

1141 pages | 07/2023 | \$59.95 | ISBN 978-1-4932-2425-8

 www.rheinwerk-computing.com/5696

We hope you have enjoyed this reading sample. You may recommend or pass it on to others, but only in its entirety, including all pages. This reading sample and all its parts are protected by copyright law. All usage and exploitation rights are reserved by the author and the publisher.