Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

A zero trust security approach with FIDO2

Stefan Bavendiek (stefan.bavendiek@mailbox.org)

- https://orcid.org/0000-0002-9404-0358

Research Article

Keywords: Zero Trust, Network Security, Security Architecture, Fido2, Webauthn

Posted Date: November 8th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-2022891/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

A zero trust security approach with FIDO2

01.09.2022

Abstract

Common IT infrastructures are built with a perimeter firewall, that separates the internal from the external network environment. Systems within the internal network are considered as trusted by conventional security architectures, while everything outside is considered hostile. However, the IT Security threats within our modern IT environments are often targeting end devices, with attack vectors such as phishing. The consequences from this mismatch between security architecture and threat landscape, become apparent with the increasing number of infrastructures that are getting completely devastated by low level malware such as common ransomware. Modern security requirements demand an IT architecture, that mitigates the risk from compromised end devices. The zero trust architecture is an approach to solve this problem and relieve the conventional perimeter security architecture. This paper will present an approach for a new security architecture that uses modern zero trust solutions to mitigate the risks from common security threats and increase the usability at the same time by replacing passwords as authentication method with the FIDO2 Web Authentication standard.

Keywords— Zero Trust, Network Security, Security Architecture, Fido2, Webauthn

Contents

1 Introduction **4**
- 1.1 Problem Statement . . . 4
- 1.2 Research Object . . . 4
- 1.3 Approach and Organization . . . 4

2 Conventional security architectures **4**
- 2.1 The perimeter problem . . . 4
- 2.2 The privileged service problem . . . 5
- 2.3 The authentication problem . . . 5
- 2.4 The central service problem . . . 5
- 2.5 End device security . . . 6

3 State of research **6**
- 3.1 The Zero Trust Architecture . . . 6
  - 3.1.1 What it is . . . 6
  - 3.1.2 What it is not . . . 6
  - 3.1.3 Zero Trust Architecture Approaches . . . 6
- 3.2 Authentication Factors . . . 8
  - 3.2.1 Knowledge Factor - Something you know . . . 8
  - 3.2.2 Possession Factor - Something you have . . . 8
  - 3.2.3 Biometric Factor - Something you are . . . 8
- 3.3 The Web Authentication Standard . . . 9
  - 3.3.1 FIDO2 . . . 9
  - 3.3.2 CTAP . . . 9
  - 3.3.3 WebAuthn . . . 9
- 3.4 Authorization protocols . . . 10
  - 3.4.1 Delegated authorization . . . 10
  - 3.4.2 SAML . . . 10
  - 3.4.3 OAuth . . . 10
  - 3.4.4 OpenID Connect . . . 11

4 Zero trust components **11**
- 4.1 Identity Aware Proxies . . . 11
- 4.2 Single Sign-on Identity and Access Management . . . 12

5 Proposed security architecture **12**
- 5.1 Example Scenario . . . 12
- 5.2 Risk analysis . . . 12
  - 5.2.1 Authentication and Phishing . . . 12
  - 5.2.2 Network Service Exposure . . . 13
  - 5.2.3 End Device Security . . . 13
- 5.3 Core Technologies and solutions . . . 13
  - 5.3.1 FIDO2 Authentication . . . 13
  - 5.3.2 Zero Trust Network Security . . . 14
  - 5.3.3 Central Software Management and End Device Security . . . 15
  - 5.3.4 Intrustion Detection, Integrity Monitoring and Central Logging . . . 16
  - 5.3.5 Automatic Remote Backup . . . 16
  - 5.3.6 Device Data Encryption . . . 16

6 Conclusion **17**

7 Further research **17**

List of Figures

1 Zero trust structure 7
2 Zero trust access process 7
3 FIDO2 Authentication process 10
4 OAuth work flow 11
5 Zero Trust Architecture 14

1 Introduction

1.1 Problem Statement

With our modern economy being increasingly dependent on reliable IT infrastructure [1], the security of these environments is becoming a critical business factor. The rise of ransomware in recent years [2] has led to an increased number of cyber attacks that completely devastate IT infrastructures of modern organizations and companies [3]. Common ransomware such as Wannacry [4] and Petya [5] have proven to be one of the biggest threats to digital infrastructure. Conventional IT Security architectures, that consist of a perimeter security approach and a trusted internal network, provide an optimal environment for ransomware and other worm based malware, as the majority of the internal services are accessible as soon as access to a single internal device is established. With the Windows operating system and the Microsoft Active Directory service as the dominant infrastructure software, the requirements for operating these infrastructures include an extensive number of services that need to be able to communicate within the internal network. The consequences of this architecture approach are significant risks originating from the compromise of a single device within the Active Directory domain. As one of the most dominant exploits for ransomware attacks, EternalBlue [6] is an convenient tool to compromise entire Microsoft environments, since the exploited SMB protocol is usually permitted within the internal network. The dominating IT Security architecture today, still revolves around a hard border, that separates the trusted internal network, from the untrusted external network. As has been shown by modern threats like ransomware, this approach is not suitable to mitigate the existing IT security risks. Additional risk areas such as weak authentication protocols and vulnerable authentication interfaces, present additional attack vectors that are often used to compromise additional infrastructure components. As such, the common security architectures do not mitigate the existing risks that modern organizations face during everyday operations.

1.2 Research Object

A modern IT security architecture has to provide effective measures to mitigate the risks of current and future cyber security threats without limiting functionality or increasing the burden on its users. This paper aims to design such an IT security architecture by using a zero trust approach and integrating the FIDO2 Web Authentication Protocol as a central authentication method. Based on existing architectures like the BeyondCorp approach developed by Google [7], the objective is to design a secure infrastructure that delivers appropriate countermeasures against modern day cyber security threats. At the same time, the strain on the users within this infrastructure will be reduced by eliminating outdated security concepts like password authentication and using single sign-on authentication schemes with FIDO2 as the core technology. The resulting architecture aims to be secure against common security threats, as well as being convenient to use.

1.3 Approach and Organization

The first part of this paper will deal with a theoretical approach to IT security architectures and introduce the conventional perimeter security architecture and the most relevant technologies that are present in modern day infrastructures. The following chapter will present the current research regarding zero trust architectures and passwordless authentication technologies. Additionally, practical security architecture will be presented, based on the introduced technologies. The second part will describe a proof of concept implementation of the newly introduced architecture. During this phase, the challenges and problems that became apparent during the implementation of the POC will be documented. The last chapters will describe the reduced attack surface and remaining risks as well as possible solutions for additional security measures that go beyond the scope of this paper.

2 Conventional security architectures

2.1 The perimeter problem

Conventional IT security architectures consist of a strict security perimeter and a trusted internal intranet. Many security technologies in this architecture, focus on the transition between the external and internal network, by using firewall and VPN solutions. Howevern these security measures are not suitable to defend against the current threat landscape [8]. With the rise of ransomware [2] as a major threat to modern organisations, this conventional security architecture approach is no longer effective. Conventional IT architectures are commonly build with perimeter firewalls separating the external internet from internal network segments, while forming a demilitarized zone (DMZ) [9] for externally available services. Since devices within the internal administrative domain are considered trustworthy, there are a number of services within the internal network that are commonly accessible without further authentication [10]. While some of these services may implement their own authentication interface, the services itself are already accessible and may be vulnerable to pre authentication exploits. Exposing these internal services to all devices within the network, presents a significant risk. Depending on the default permissions for internal members of the organisation, services like writeable fileshares are a valuable target for ransomware. However, even without default permissions on commonly shared resources, the network interfaces of these internal servers are reachable. This means, that vulnerabilities that are present within these services are exposed to all internal devices, even if there are no default permission available. As a result, malware or adversaries that have compromised a single device, can easily target additional resources within the internal network. The huge attack surface that this architecture presents to internal devices make further attacks a lot easier. Part of this issue is the result of the design of the Microsoft Active Directory. While some mitigations are possible, an effective implementation of the active directory service, requires extensive communication between services, resulting in a flat network design [10]. Another issue results from the available information within the directory service. As it is by default possible for domain members to query information about available resources, information harvesting results in an abundance of valuable data such as information about all domain users, their

member groups and permissions. There are a number of common services that are usually available on the intranet of many organizations and that can be targeted by an attacker who compromised a single domain joined device. Maintaining a strong perimeter security between the internal and external network within the design components of conventional security architectures is already close to impossible with the rise of cloud services. With the increased use of cloud based services and applications within enterprise networks, the traditional security approach does no longer work as intended. With common client devices using cloud services as part of the intranet, the separation between internal and external network is increasingly hard to maintain and brings new challenges for security [11]. To mitigate the risks introduced by these new dependencies, the approach to network security needs to be adapted to work with an environment that tightly integrates services from different networks and organisations.

2.2 The privileged service problem

The available attack surface within common internal networks is also often extended to administration interfaces of network and security devices. Unless all network interfaces and services used for administration are confined to a privileged network, reachable by a jump server solution or comparable measure, vulnerabilities in these core services can have devastating consequences. This is especially the case with management interfaces for network devices and firewalls, but also other core services like domain controllers, hypervisors and out of band management interfaces. The security architecture needs to ensure, that privileged services cannot be reached on a network level by unprivileged users and devices. Relying on the native authentication implementation of these critial services, will not only make them vulnerable to any pre authentication exploits, but it will also increase the attack surface significantly due to potentially weak authentication methods and lack of user and permission management. One established method to reduce the exposure of privileged services is via jump servers. Also known as administrative hosts, these systems are hardened bastion hosts that do not have normal application software installed to reduce the attack surface. To connect to central security critical services, the administrator needs to take an extra step and connect to the jump server in order to reach a privileged network segment. Only after successfully authentication with privileged user credentials, the administrator can further connect to the privileged target services. There are a number of methods to implement this kind of jump server, but one of the most common software applications is the Microsoft remote desktop service. While this solution can mitigate the risk of exploitable vulnerabilities in privileged service interfaces that are isolated by this measure, the RDP service itself is highly complex and is not designed as a security perimeter guard [12]. The critical services that are protected by these RDP jump hosts, will still be accessed from these servers that act as clients themselfes and handle the credentials required to authenticate to the target service. RDP server have become one of the most privileged targets itself as a result [13] and a compromise of these jump hosts will have devastating consequences. The attack surface problem originating from complex network protocols can be solved by alternative protocols that were designed for untrusted environments, such as secure shell [14]. However, the jump server solution can only reduce the exposure of privileged services for unprivileged user. Once the user is authenticated and has accessed the jump host and administrative target service, any attack or malware present on the users end device is still able to access all those services and information. While end device security is a separate topic in itself, it is crucial to highlight the remaining risks. This is especially relevant since jump hosts are also used as a separation between different organisations, like internal and customer networks. This however is a misconception, as there is only a weak separation within the application layer. The data entered and retrieved within the application is still displayed on the end device. The jump server solution is therefore able to reduce the exposure of privileged services, but still requires a trusted end device that needs to be permitted to access all accessible data.

2.3 The authentication problem

To this day, the most dominant authentication methods are still based on user defined passwords, even though its weakness has been known for more then two decades [15]. Attacks on password authentication, such as brute force attacks, credential stuffing and phishing play a significant role in modern cyber attacks [16]. Requiring an increasing number of complex passwords for different services, also puts a burden on the users, which leads to password reuse and weak user passwords [15]. Phishing attacks are also the dominating method for modern ransomware attacks [2]. As a consequence, modern IT security architectures need to combat this issue by deploying authentication methods, that prevent phishing attacks altogether. There are a number of alternatives to password based authentication methods [17] that could be deployed in modern IT environments. Although an increased number of services already require additional authentication factors [18], the use of password based authentication is still the most common authentication method today [19]. One of the reasons for this is the easy implementation and existing compliance regulation [18]. To avoid the well known issues with weak authentication protocols, modern IT security architectures need to deploy and enforce alternative solutions.

2.4 The central service problem

Complex IT infrastructures are commonly build using a central identification service, such as Active Directory or alternative LDAP implementations. This service is used to manage users, groups, permissions and access to the different organizational networks and services. Consequently, the users who manage this system, such as the help desk, need to work with highly privileged user accounts, that can change the security settings of most users and devices. The responsible administrators of these central directory systems, are often the highest privileged members of the organization. This results in a significant security risks, as the compromise of a single privileged user that has the permission to manage this central identification system or its resources, will result in the loss of integrity for the complete IT organization. This issue is most relevant in large organizations, with multiple locations or even different continental offices. To mitigate the risks resulting from these security critical central services, the security architecture needs to reduce both the consequences of compromise as well as the initial risk of compromise through reduction of the attack surface. One of the key components to consider when designing solutions for this issue, is the exposure of network protocols to these central services. Implementations such as Microsoft Active Directory offer some methods to both reduce the risks by implementing a forest structure, as well as offering read

only services for authentication in less protected networks. However, as with other conventional implementations, there are extensive requirements for network communication via complex protocols [10] that can potentially be exploited. The challenge for new architectures is therefore, to design a system that allows an indirect communication from less secure network devices to highly classified services, without compromising security.

2.5 End device security

End devices, like desktop PCs or corporate mobile phones, are one of the most prominent sources for malware infections [2]. The reasons for this are manifold, with end users being one common factor for infection. Another relevant impact on end device security is the various complex applications that can be found on common enterprise devices, such as mail clients, office products and internet browsers. These software products do have a large attack surface and are among the most affected software products in regards to vulnerabilities [20]. Due to the requirements of online connectivity for many business processes in the digital age, as well as the common and widespread use of historic communication protocols such as email messaging, a dominant intrusion method against end users and devices is via phishing attacks [21]. End device security has been an important factor in conventional security architectures. While modern security concepts like zero trust will be able to mitigate or reduce many risks coming from compromised or otherwise untrusted end devices within the network, they will continue to be a major factor for a trustworthy IT infrastructure. Eventually the data that is handled within an organization will need to go through the devices that the users are utilizing to access and modify this information. All input and output data from digital services that the end user works with will at one stage pass the end device or even originate from it. As a consequence, it is crucial that these devices are a trustworthy basis for keeping this data secure. With the large attack surface that end devices present, this is one of the major challenges for future security infrastructures.

3 State of research

3.1 The Zero Trust Architecture

3.1.1 What it is

The zero trust architecture implements the principle of least privilege within the complete IT infrastructure of an organization [22]. Unlike with conventional architectures, there is no default permission based on the used device or location. All permissions need to be granted based on the identity of the user and only after authentication has taken place. Additional factors that influence the authentication process can include the current location and the state of the device compliance [23]. The zero trust paradigm is a response to the evolving IT landscape that is becoming a collection of distributed services from various vendors and locations. A modern IT infrastructure consist of services that are provided not only by the internal IT department, but rather are distributed across the globe and managed by different organizations. The rise of cloud based applications and services has sped up this development to a point, where conventional security architectures, that protect networks instead of individual assets, are no longer effective. The zero trust approach is an effective answer to the security challenges of modern IT environments. If implemented correctly, this architecture will eliminate the risk of lateral movement and greatly reduce the damages that can result from a successful cyber attack. The zero trust paradigm extends not only to the area of network services, but also to data access and transmission. There are a number of technologies that can be used to implement zero trust in different areas. In consequence of the broader definition, even end to end encryption in modern messaging systems [24] can be considered as a part of a zero trust architecture. Emerging technologies such as homomorphic encryption [25] also have the potential to greatly reduce risks by reducing the exposure of information to unauthenticated entities. In the end, a zero trust security architecture is a collection of technologies and paradigms that enforce the principle of least privilege and minimal access within an organization or infrastructure as a whole.

3.1.2 What it is not

There are a number of definitions for zero trust and different vendors provide various products that are intended to provide a zero trust concept as a service or feature [26]. It is however crucial to understand, that zero trust is not a specific technology or measure to be implemented into an existing infrastructure. The zero trust security architecture is a paradigm that serves as a basic concept for all design and architecture decisions regarding security and permission models and it cannot be considered as a single product or solution. Similar to the secure software development lifecycle framework, the zero trust architecture needs to be integrated into the architectural design of the IT environment as a whole. While it is possible to implement this architecture into a subset of infrastructure compartments, the resulting security gain will greatly depend on the remaining attack surface and can therefore make a partial implementation ineffective. It is also crucial to note that a zero trust architecture on its own does not eliminate all threats to the IT environment. One of the most prominent attack methods in the form of phishing, is still a valid threat [22]. It is therefore advised to consider this approach as a basis for a secure IT infrastructure and not as a final solution. Besides the issue of weak and phishing prone password based authentications, another remaining issue lies with end point security. Although the zero trust architecture will greatly reduce the consequences of compromised devices within the network, it will not reduce the risk of compromised end devices itself.

3.1.3 Zero Trust Architecture Approaches

There are a number of approaches to implement a zero trust architecture. The central components relevant for the implementation of a zero trust architecture are the policy decision point (PDP), responsible for authorization and the policy enforcement point (PEP), which is responsible for authentication [22].

As one of the core principles of zero trust, all access permissions are granted based on a set of predefined rules. The exact definition of these rules and the components that define and determine them can vary. The same is true for the enforcement

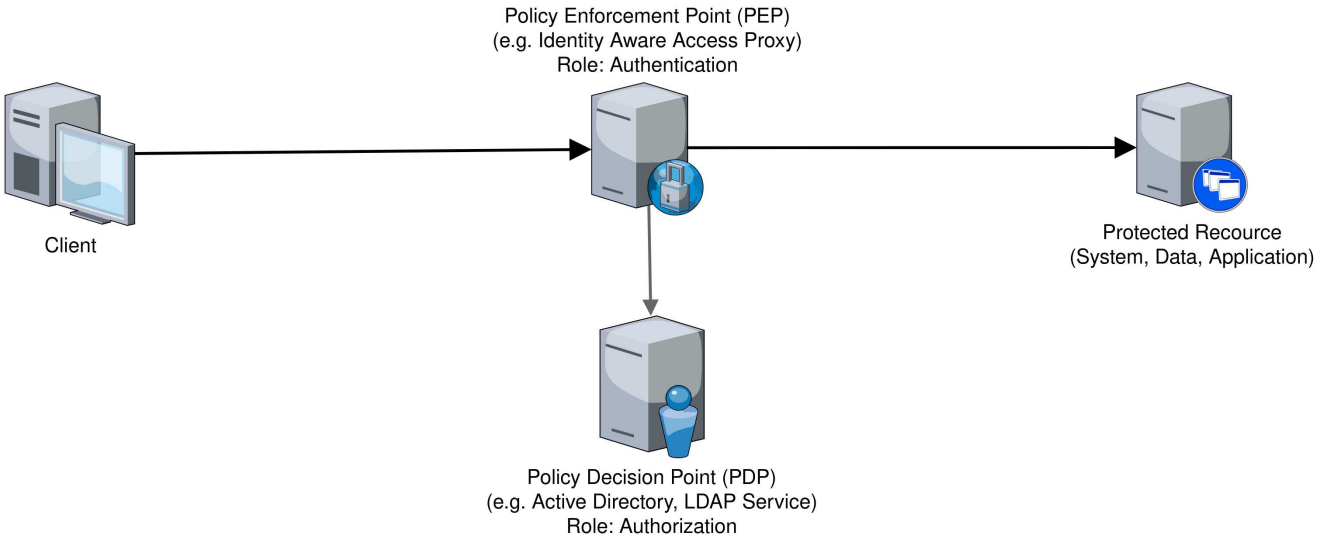


Figure 1: Zero trust structure

point of these rules. The implementation that is most appropriate depends on the structure of the target organization. While all approaches implement the core principles of zero trust, the used technologies can be different [22]. The **identity based** approach enforces the decision rules by using authentication portals as policy decision points, that grant access permissions to services based on the identity of the user that is requesting access. Additional factors such as used device and asset compliance status can also be considered. This approach is often implemented by using an open network model, which on its own does not by default permit access to any internal services, but can be accessed by internal and external devices alike. Access is granted to requested services, only if the identity of the user has a specific permission for this resource. Therefore the identity management is one of the major components for this approach. Alternative implementations use **micro segmentation** that place either individuals or groups of components into network segments. All transition points between segments are guarded by next generation firewalls (NGFW). These gateways dynamically manage all access requests from clients. Part of this approach can be an agent on the client device that aids the decision enforced by the firewall devices between the micro segments. This approach requires a central identity and permission management as well, but relies on the gateway devices to enforce the defined access rules. **Software defined perimeters** use an overlay network to implement zero trust. The most common deployment of this model uses an agent on the client and a resource gateway as a policy enforcement point. Based on the defined rulesets and device conditions determined by the agent, a secure channel between client and the requested resource is established. While the mentioned zero trust approaches are different implementations, practical implementations are not necessarily confined within these definitions. A security architecture can also be designed using multiple different approaches for different segments of the infrastructure. For example it is feasible to implement an identity based approach using software defined perimeters with or without agents for the client environment and at the same time use a micro segmentation approach for the communication between the server backends. The common condition for zero trust is that the principle of least privilege is enforced throughout the complete IT infrastructure.

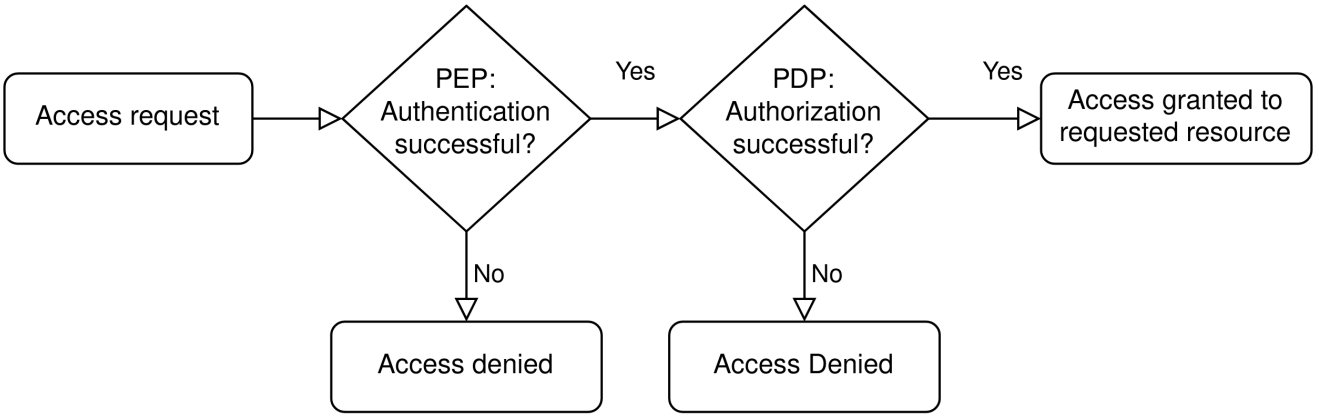


Figure 2: Zero trust access process

3.2 Authentication Factors

3.2.1 Knowledge Factor - Something you know

Authentication factors can be separated into multiple different groups. The most well known group is the knowledge factor, that uses a shared secret like a password to authenticate a user. The security problems that result from this authentication method have been well known for more than two decades [15] and the problems continue to be a major threat to account security [27]. Passwords are not only easy targets for phishing attacks [21] but also need to be stored securely by the authentication service, which presents additional challenges [28]. The widespread use of password based authentication and the resulting security issues have already been well researched. Newly implemented authentication systems and architectures should therefore depend on alternative factors as additional authentication methods or as replacement for passwords. Closely related to passwords are personal identification numbers (PIN) [29]. A PIN can be a numeric or alphanumeric passcode. This method is mostly used in combination with a smart card or local device and is usually shorter then passwords. Personal identification numbers or passcodes are not directly used to authenticate against a remote system, but rather are used in combination with a smartcard as a possession factor. This kind of two factor authentication system originates from the introduction of automated teller machines in 1967 [30]. The main difference between passwords and PINs is based on the fact that a PIN is never transmitted to a remote system but only serves as a local authentication factor in combination with a secure device such as a smart card. By ensuring physically that the number of authentication attempts are limited, the PIN can be kept short and still serve as a secure knowledge factor. As such PINs are still used today and present a reliable authentication method when used in combination with a smart card or secure device. Most attacks on PINs are targeted at the smart card or security device used [31]. PINs are still vulnerable to phishing attacks in principle, although the physical component makes this attack vector significantly harder to exploit. Another common knowledge factor exists in the form of security questions. While there are some differences in the risks as compared to password based authentication, the underlining security problems are the same, as this is still a knowledge factor which depends on the users choice and also has the be store securely by the remote authentication system.

3.2.2 Possession Factor - Something you have

Security tokens and devices are a common possession factor used for authentication. Smart cards are one common and widespread authentication factor [31]. The main benefits of smart cards it the temper resistance and the possibility to protect the confidentiality and integrity of the stored data. Smart cards are often protected with an additional security PIN to protect against theft. They are used to securely store a unique cryptographic key that is associated with the users identity. A smart cards can be used to perform a cryptografic operation within the temper resistant device and thereby perform that authentication process such as by signing a challenge with the stored private key. There are also smart card based security tokens that produce a time based code for authentication by implementing the time-based one-time password algorithm (TOTP) [32]. The TOTP algorithms is an improved version of the HMAC-based one-time password algorithm. While this method improves the authentication security by eliminating many weaknesses of password based authentication methods, it is not immune to some of the previous attack vectors such as phishing [33]. While security tokens based on signing authentication challenges require a connection to the end device, tokens that implement TOTP can be separated and kept offline. Since the use of physical tokens for authentication purposes present an additional challenge due to the procurement and the management of these additional components, an alternative implementation of TOTP and similar authentication schemes are based on soft tokens. Soft tokens are basically software applications that implement the same algorithms that smart cards and security tokens offer. The downside of soft tokens is the limited protection against attacks on the end device they are installed on. Alternative authentication systems using out of based communication with separated devices belong into this category as well. One of the most common methods is SMS verification using a mobile phone. Although using SMS codes as a second authentication factor is quite common due to the ease of implementation, the remaining security risks in this method are still exploitable [34], due to the historic security weaknesses build into this communication protocol. Common for all listed authentication methods is the use of ephemeral tokens. All authentication methods that either work by signing a authentication challenge or use a one time password to be transmitted, avoid the risk of replay attacks as well as the requirement to store an individual secret on the authenticating service.

3.2.3 Biometric Factor - Something you are

Biometric authentication factors use inherent attributes of a user as proof of identity. The biometric factor itself is not a modern invention. Identifying individuals by their physical attributes has been an established method outside of digital applications for many years [35] and the use of hand signatures is even older [17]. The use of biometric identification factors for online authentication however has been established in more recent years [36]. The widespread use of biometric authentication has been started with the introduction of Apple Touch ID in 2013 [37]. Utilizing facial recognition technology for authentication has also been introduced to the commercial marked by Apple in 2017 [38]. The established biometric authentication methods are using local devices and hardware to measure the physical attributes of its users and compare the calculated values to the locally stored data [37]. The authentication process does not require online services and no biometric information needs to be transmitted. While using biometric factors can result in a reliable authentication method, there are a number of potential issues that limit the practical security as well as the trust users put into this technology [37]. While the possibility of spoofing a fingerprint has been a known threat since its introduction [39] there is also the trust issue regarding the information storage. Since the fingerprint or facial structure is not only unique but also very private information, one legitimate concern for users is whether the device or software manufacturer abuses the access to this data and extracts the information. From a security perspective, the use of biometric authentication as a second factor can improve the authentication security, while the use as a sole authentication method introduces some significant risks and is not advised [39]. With fingerprint and face recognition, there are already two established biometric factors. There are

however multiple additional biometric factors, that have not yet been introduced into the mainstream market for end users. Among these biometric factors are voice recognition, eye and iris scanning, as well as recognition of movement.

3.3 The Web Authentication Standard

3.3.1 FIDO2

While there are a number of established multifactor authentication methods, like TOTP and SMS based out of band verification, these previous technologies were still vulnerable to a number of attack vectors, like phishing [40]. The W3C web standard Fido2 (Fast IDentity Online) aims to remove those remaining threats by defining a new interface to authenticate users to web based applications based on public key cryptography [41]. The key objective is to provide a secure alternative to existing multifactor authentication methods as well as to passwords itself. The core component of FIDO2 are the Web Authentication (WebAuthn) standard [42] and the Client to Authenticator Protocol (CTAP) [43] which are described in the next chapters. The origin of the FIDO2 authentication specification comes from the Universal 2nd Factor authentication standard (U2F). The U2F and FIDO Standards are developed by the FIDO Alliance and the World Wide Web Consortion (W3C). Initially, the protocol was developed by Google and Yubico which formed the FIDO Alliance in conjunction with other IT companies. The practical security gained by the use of U2F has already been proven, as Google has been using U2F based security tokens for its internal services since early 2017. Google confirmed in 2018, that since its implementation there has not been a single account takeover [44]. There are three relevant components for U2F/Fido authentification. The service that the user needs to authenticate to, takes the role of the Relying Party. The Browser has the role of the proxy and the authenticator is managing the cryptographic keys of the user in a secure manner. The Communication between the relying party and the browser is done by using the WebAuthn protocol, while the communication between the browser and the authenticator is done using the CTAP protocol. The authenticator is an abstract definition and its implementation can vary. One common form of authenticator is a hardware based security token, that communicates with the end device by using a local protocol, such as USB, NFC or Bluetooth. The authenticator can also be implemented by using an external device such as a smartphone. It is also possible for the operating system to provide an authenticator service by using the Trusted Platform Module (TPM) of the end device for managing the cryptographic keys. The advantages of the FIDO authentication standard are based on multiple solutions for long known issues with other authentication methods. Since the FIDO authentication process relies on well established cryptographic algorithms instead of user defined passwords, the weaknesses of password based authentication are avoided. It is no longer possible for a user to use a weak authentication secret, reuse it or share it with other parties. Since the FIDO authentication process also validates the identity of the relying party, phishing is no longer a valid attack vector as the authenticator will not work on services that have not been previously registed with the authenticator. The management of shared secrets by the service is also no longer necessary as the only information that are saved by the relying party are the public keys of the users. Compromised web services will therefore no longer lead to compromised authentication information. Although the FIDO2 standard has only been officially introduced in 2018 [45], there is already widespread support for this authentication method. All major operating systems, and web browsers already support the FIDO standard [46]. At the same time, all major cloud service providers offer support for web authentication, acting as a relying party. Since a simple form of the authenticator can be a hardware device there are a number of USB and NFC devices available, such as the ones from Yubico [47] or the open source tokens from Nitrokey [48]. Google has also made their own internally used Titan authenticator available [49]. A full list of supported plattforms and hardware tokens can be found here [50]. Aside from authenticators based on external hardware tokens, it is also possible to use internal hardware like a TPM or Apples Secure enclave to manage the cryptographic keys. As a result, a smartphone like the Apple IPhone can also serve as an authenticator [51]. Android phones starting with version 7, are also able to serve as a FIDO2 authenticator [52]. In addition to the authenticator hardware, in the form of a separate device or internal security chip, there is also a number of local authentication methods available to serve as an additional authentication factor. In case of most hardware tokens, this can be an additional pin to be entered for the authentication process, a touch button on the local device or a fingerprint sensor. On Apple IOS, the TouchID and FaceID authentication methods can be used as additional factor. As part of the authentication process, the relying party can be configured to demand an additional factor in addition to the actual authenticator, or it can allow the authentication process with only the authenticator available on its own. Unlike the previous U2F standard, the FIDO2 authentication method can be used both as a second factor in combination with a password or other authentication factor, as well as passwordless. Since it is possible to configure the target service to demand different numbers of authentication methods, it is possible to use FIDO2 both as a replacement of password based authentication methods, as well as one of multiple authentication factors, based on the circumstances or target services.

3.3.2 CTAP

The Client to Authenticator Protocol (CTAP) defines a communication protocol for the user agent on the client device, to communicate with the authenticator device [43]. As part of the authentication registration, a pair of public key credentials is created by the authenticator as requested by the web application. When a user authenticates to a remote web service, the verification of the authentication request is done on the authenticator. The process of forwarding the relevant authentication data to the authenticator, is definded by the CTAP protocol. Originally the CTAP protocol was defined as part of the U2F authentication standard, also known as CTAP1/U2F. The later specification of CTAP2 was adopted by the FIDO project. Authenticators that implement the CTAP2 protocol, are FIDO2 authenticators, while the implementation if the previous CTAP protocol, allows only the use of the U2F authentication method.

3.3.3 WebAuthn

The Web Authentication (WebAuthn) specification is a W3C standard, defining an API for the creation and use of strong authentication as part of the FIDO authentication protocol [42]. The WebAuthn specification is used to communicate

between a Relying Party and the User Agent on an end device, like a web browser. The access to the registration and authentication is provided via a javascript API, implementing the WebAuthn standard. If a web service offers WebAuthn as an authentication method, the user can register a compatible authenticator for this service. As part of this process, a public key is created on the authenticator and registered at the web service as part of the user identity. The user also receives a identification data from the web service itself, which includes the public key with the associated domain name of the web service. When the user initiates the authentication process and receives a request for authentication from the remote service, the domain name that initiates this authentication request is compared to the saved domain name on the authenticator. This prevents other services, imitating the targeted web application as part of a phishing attack, from initiating the authentication process. Since the validation of the remote service request is done by the authenticator, this attack vector is effectively made unfeasible. The verification of the authentication request is done using the CTAP protocol for communication between the user agent (Browser) and the Authenticator.

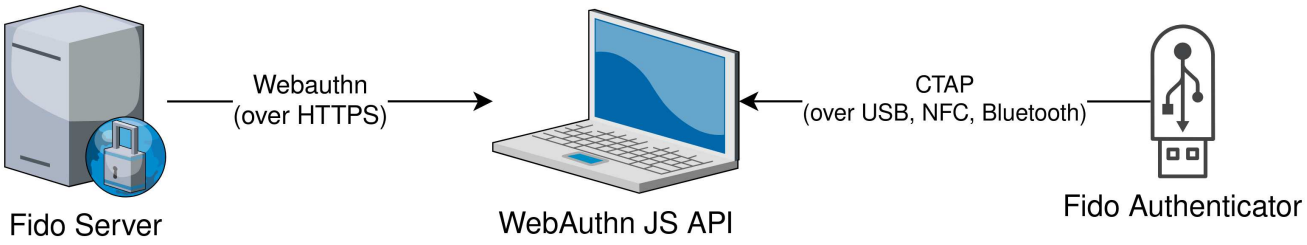


Figure 3: FIDO2 Authentication process

3.4 Authorization protocols

3.4.1 Delegated authorization

The authentication method used by an application is relevant in regards to the attack surface presented by the user interface. Weak authentication can compromise the identity of the affected user and as a consequence all the data that is protected by this method. In contrast to this, the authorization process is responsible for managing what a authenticated user is permitted to do within the system. This can extend to the information visible to the user or the actions permitted to be executed. By managing the authorization of user permissions tightly the principle of least privilege can be implemented. A well structured minimal privilege setup can reduce the risk of compromised accounts. This is especially relevant in large organizations where a single user often only requires access to a small subset of resources and services within the infrastructure. The authentication and authorization of users is a security critical element of an IT infrastructure and can include a significant amount of effort for every application. Single sign on services (SSO) and identity aware proxies [53] can help to minimize the effort necessary to implement and manage user authentication and authorization. This however makes it necessary to communicate the user identity and authorization information to all relevant applications within an organization. To do this in a effective and secure manner, a number of protocols have been established for this purpose. One of the most widespread standards for exchanging authentication and authorization information between identity and service providers is the Security Assertion Markup Language (SAML) [54]. An modern alternative and designed specially for the HTTP protocol is the OAuth 2.0 standard [55]. Based on OAuth is the OpenID standard which is at its core an extension of OAuth and manages the authentication identity in addition to the authorization information [56].

3.4.2 SAML

The Security Assertion Markup Language (SAML) is an open standard for exchanging authentication and authorization information between identity providers and service providers like applications [54]. SAML uses the Extensible Markup Language (XML) as an data exchange format and a primary use case for SAML is the transfer and delegation of authentication and authorization information in the context of single sign on applications. While there are several versions available, the current version is SAML 2.0 which became a standard in 2005. The work flow of the SAML protocol is initiated by the client (User Agent) and triggers a number of communication processes with the service provider and the identity provider. At its core, the user who request access to a service is redirected to the identity provider to authenticate, before the identity provider refers the user back to the requested resource at the service provider if the user is successfully authenticated and authorized to access the requested resource.

3.4.3 OAuth

The OAuth standard aims to allow services to provide access to specific resources for third party applications and providers [55]. This allows users to share selected information managed by a service with other services in a secure manner. While SAML manages identities and single sign-on in federated environments, OAuth is used for the authorization of resources and is not intended for authentication. OAuth is especially suitable to provide access to resources but can also be used for single sign-on use cases. OAuth is also designed specifically for the HTTP protocol and can easily be implemented on mobile platforms. One of the most significant security benefits of OAuth is the use of back channel communication for sensitive information.

When a user authenticates to an authorization service, requesting access for an application to a specified resource, the authorization service issues an authorization code, that is send back to the users client. Since this communication takes

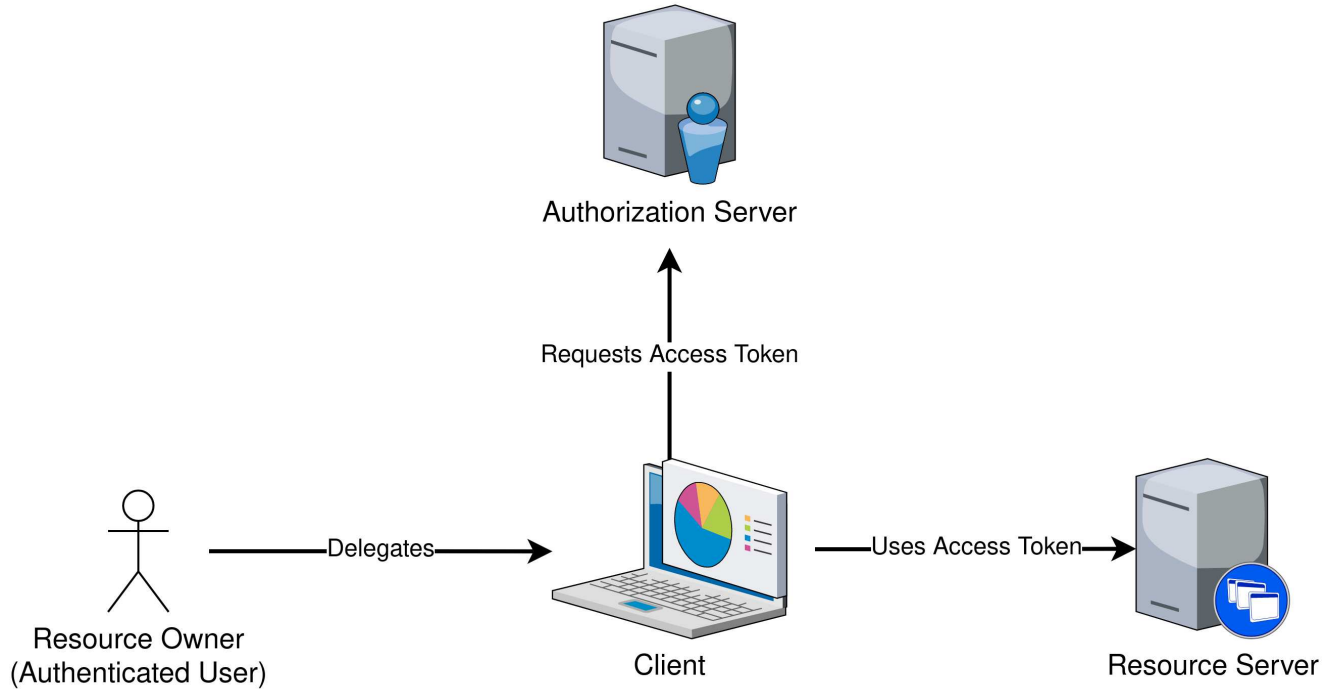


Figure 4: OAuth work flow

place on the front end, using complex applications like web browsers with a large attack surface, this authorization code cannot be used directly to access the requested resources. Instead the authorization code is send to the applications backend server, which in turn will contact the authorization server and request an access token, while providing the authorization code and a shared client secret. If the code and secret are valid, the application is provided with the access token, which can be used to access only the resources the user has granted access to. Since the access token is never transmitted to the frontend, a compromised web client will not provide access to the requested resources. For use cases that do not involve a backend application server, the access token can also be send through the frontend without the additional security benefits.

3.4.4 OpenID Connect

While is is possible to transmit authentication information using OAuth, the protocol is not intended for this purpose and should not be used for anything other then authorization. The OpenID Connect protocol, is an extension of the OAuth 2.0 protocol and also provides authentication information [56]. OAuth can authorize an application to receive specified information after authentication, but there is no intended way to transmit authorization information about the user that permitted the access in the first place. OpenID allows clients to verify the identity of the user based on the information provided to the authorization server. This allows OpenID Connect to be used for the same use cases that SAML provides, while also providing the advantages of the OAuth 2.0 protocol.

4 Zero trust components

4.1 Identity Aware Proxies

While single sign-on services have been an established technology for years, a core component used to implement a zero trust architecture, are identity aware access proxies (IAP). This infrastructure element is designed to validate the user trying to access internal services by referring to an authentication and authorization service. After the user has been authenticated and the authorization service has confirmed the required permission to access the requested resource are granted, the access proxy will allow a network connection to the targeted service.

To implement this architecture effectively, it is necessary to block all direct network connections to internal services and only allow a connection to and from the access proxy. This structure is similar to the setup for jump servers, that are also designed to only permit connections to and from the jump box. Since most identity aware proxies are new products, some do not yet allow the use of non HTTP based services like SSH. A zero trust implementation using such proxy services, can require the use of traditional jump servers for non HTTPS services, like SSH or RDP. The risks involved in such an approach need to be assessed depending on the actual implementation, and will be the subject of a later chapter. There are a number of suitable projects available, that offer core components for the implementation of a zero trust architecture. The most suited product depends on the requirements present in the target infrastructure and will be discussed in the example scenario in the next chapter. Ideally the access proxy is the only network service that is reachable for any unauthenticated network participant. Depending on the requirements, this proxy service can either be part of an internal network, that is accessed through a VPN service, or it can be made directly available, as it is the case with Googles identity aware proxy service [57]. Since the access proxy is a web application based on http server software that was designed to be exposed to the internet, this proxy service can also be used as an replacement of traditional enterprise VPN services. The internal

infrastructure attack surface presented to any unauthenticated user or device, is therefore reduced to that of the access proxy. When a user is authenticated, the authorization session can be limited to only the currently requested service. This reduces the attack surface towards authenticated users to only the currently accessed service, while other network services cannot be reached on a network level. In case of compromised clients, any possibly present malware can therefore only access the data available on the local device and the currently accessed service. This makes it possible to reduce the potential damage caused by compromised devices significantly, depending on the access permissions of the user and the implemented incidence detection and response mechanism. Should an attacker be able to compromise a service the user is authorized to access, any internal service should be prevented to access any other unrelated services by a strict firewall setting.

4.2 Single Sign-on Identity and Access Management

One of the key components required in addition to an identity aware proxy, is a source for identity information, authentication and authorization. This kind of application service has been around even before the idea of zero trust was specified in 2010 [58]. One established method for the implementation of identity and access management is via LDAP servers like Microsofts Active Directory (AD). Application using this kind of historic service, will ask the user for their AD credentials and forward this information to the central directory service. In case the credentials are valid, the AD server will respond with the users information, like group memberships. The application service can thus authorize the user depending on the success of the authentication process, as well as based on the user information like group memberships. One major security weakness in this kind of authorization, is that the users needs to send their credentials to all application services they authenticate against. If a single application server has been compromised, the attacker record all credentials used to access the service. This risk has been mitigated with modern alternative implementations for true single sign-on, such as Microsofts Active Directory Federation Services (AD FS) [59] or Red Hats Keycloak [60]. These SSO services can be implemented using authentication and authorization protocols such as SAML and OpenID. By using this kind of established protocols, it it possible to implement a software architecture that avoids sending any confidential information such as credentials, to application services that user are accessing. When users in such an environment need to login to a network service, they will be redirected to the SSO service to authenticate themselves. The SSO service will then send back the users authentication and authorization information to the targeted application using a protocol like SAML or OpenID. In case the WebAuthn protocol is used as authentication method, the authentication service is refered to as the WebAuthn Relying Party. A zero trust implementation using Identity aware proxies, will work similar, with the only two major differences. Any request to access an application service is send to the access proxy instead of the backend application and the authorization and authentication information are send to the requested application by the SSO service, instead of the access proxy. The IAP service will therefore never receive authentication requests or authentication information. The exposure of network services in this setup, regarding unauthenticated users, is limited to the IAP service. As the authentication request need to be forwarded to the SSO service, the used authentication APIs will also add to the attack surface.

5 Proposed security architecture

5.1 Example Scenario

To illustrate the practical setup of a zero trust security architecture using FIDO2 for authentication, this chapter will describe an example implementation. In this scenario, we have an IT department that has an elevated risk of being targeted and that works on several highly sensitive IT infrastructures for different organizations. As such, the members of this infrastructure are in possession of multiple sets of privileged accounts, used for either working on data or for administrating IT services. It is further assumed, that only the members of this department have administrative access to the infrastructure components, while external customers will also need to access the data and frontend applications, such as ticket systems. Both the members and the external users, require access from different remote locations. User information for internal members are managed locally and no internal application data may leave the local datacenter. External cloud services are used to deliver some services for the customer. External users will also be permitted to use identity information managed from outside of the organization, as long as the requirements for authentication are met. The landscape of internal applications, consists mostly of web based applications that manage unspecified data from external customers. Additional services used for internal infrastructure management and administration include LDAP servers for user identity management as well as SSH for administration. The available hardware resources include a number of servers forming a hypervisor cluster. All application services run on virtual servers within this cluster. While the full description of the IT architecture of this infrastructure involves a more comprehensive specification, including details to specific services and used technologies like required hardware, hypervisor software and high availability configurations, this proposal will only include the description of the security architecture.

5.2 Risk analysis

The provided example scenario presents a fairly common use case for IT infrastructures. As such, there are a number of common security risks that need to be considered when developing applicable security solutions. The following risks, are assumed to be the main threat vectors for the described environment, based on the previous analyses of IT security architectures and authentication methods.

5.2.1 Authentication and Phishing

As the previous chapters have already described the risks involved in common authentication methods and associated threats like phishing, it is clear that the central authentication methods used to access the internal services within this infrastructure

presents one of the main security risks. Strong authentication technologies are therefore a central requirement for a modern security architecture.

5.2.2 Network Service Exposure

Conventional security architectures are not suitable to protect large internal networks from modern security threats, as has previously been discussed in detail. The exposure of internal network applications presents a major attack surface and needs to be mitigated using a modern network security approach. These risks extends not only to the network service itself, but also includes the authentication methods, interfaces and the user management.

5.2.3 End Device Security

Independent of the security measure taken to protect the data and services on the network, the security of the end devices used to access the internal systems is critical. Compromised end devices allow attackers to access any data and system that the user of the infected end device accesses. Since the user will authenticate against the accessed systems as a legitimate user, most security measures will not be effective against a compromised end device. A strict minimal privilege and access architecture can significantly limit the consequences of a successful compromise, but all data the effected user access legitimately will be exposed to the attacker.

5.3 Core Technologies and solutions

5.3.1 FIDO2 Authentication

The most central technology to mitigate the risk from weak authentication and phishing, is the FIDO2 protocol. By providing hardware tokens for FIDO authentication to all users, the foundation for strong authentication has been established. As fallback, as well as for external users, alternative FIDO2 implementations, such as provided by IOS and Android devices can be used as well. The advantage provided by separate hardware devices is especially the tough option for authentication. While FIDO2 allows the authentication process to success with just the hardware token attached to the end device, this configuration can be exploited to authenticate a user without interaction even when it is not intended. Compromised end devices could use this to access all services that user is permitted to use. With the touch option, the user has to interact with the device to authenticate with a service. This adds additional security to the authentication method, while still being a comfortable method for the user. This authentication method will be required for all applications and accounts required for work on normal internal and customer data. With FIDO2 it is possible to set a timer for reauthentication. This timer will be set to 12 hours, allowing for a normal work day without the need to reauthenticate, as long as the FIDO2 token is present. At the start of the work day, when the device is started or when the 12 hour timer has expired, the user will be required to unlock the FIDO2 token using a PIN code as additional local authentication against the token itself. This PIN will also be required when the end device is rebooted or the hardware token has lost connection with the end device. This PIN will only be required once during a normal work day and therefore it will not add any significant inconvenience for the user. Authentication for services that are used for administrative purposes, require an additional factor. To authenticate against any service used for administration, the user will be required to use a TOTP code as additional method after the FIDO2 authentication has succeeded. This will prevent any phishing attack surface present with TOTP authentication. Since some products for zero trust implementations like the identity aware proxy Pomerium [61], do not yet fully support non http based applications, the administration interfaces using SSH, will require an additional, parallel access device until arbitrary TCP services are supported. As the example IT infrastructure uses SSH for administration access, a jump server for SSH connections will be required. This jump server will also implement the same authentication measures as the access proxy does. The common OpenSSH server software is not only widely used for security critical systems that are exposed to the internet, it also supports FIDO2 authentication since version 8.2 [62]. As such it is suitable to serve as an alternative access service within a zero trust architecture, since the required authentication method is supported and the used network protocol is known to be reliable in regards to security. The OpenSSH server also supports another feature critical for zero trust, which is provided by the ProxyJump option [63]. By using ProxyJump, the user authenticates against the OpenSSH jump server first, before an additional SSH connection is forwarded to the target server. The connection between the users client and the target server is encrypted along the way, without the jump server being able to see the traffic, which is the case with the traditional ProxyCommand option [63] that initiates the final connection from the jump server itself. Using ProxyJump, will result in the same security architecture as is the case with a zero trust access proxy. Using this option will result in the user having to authenticate using FIDO2 first, before an additional SSH authentication is requested. The Fido authentication for administrative SSH access will require the user to enter a PIN code for local authentication to the FIDO2 token. The login to the target server will require key based SSH certificates with the private SSH keys stored on a secure token device. A possible FIDO2 compatible token with the ability to store SSH keys, is available with the YubiKey 5 Series [47]. Alternatively, both the FIDO2 secrets, as well as the SSH key material, can safely be stored using the TPM of the end device, although this will not allow the use of touch login with FIDO2. One requirement for all internal services will be the compatibility to the OAuth2, OpenID connect or SAML2 protocols. Since enterprise applications commonly support at least one of these protocols, compatibility issues are unlikely. The IT security architecture resulting from this setup, will not require the use of traditional passwords at all. While it is still possible to allow authentication with passwords as secondary factor in addition to FIDO2 or WebAuthn respectively, it is not recommended.

5.3.2 Zero Trust Network Security

The central component for the implementation of a zero trust security architecture, is the identity aware access proxy, that is used to permit authenticated users to connect to the requested services. Identity aware access proxies will shield the infrastructure services from all unauthenticated users and restrict authenticated users to those services they are authenticated and authorized for. A product like Pomerium [61] provides the required functionality to implement this kind of central security application. Pomerium provides a single sign-on gateway that can fulfill the role of the access proxy within this example infrastructure. While it is possible to use this access proxy as an alternative to traditional VPN services, for the purpose of this security architecture, a VPN will be used as an additional layer. One reason to avoid traditional VPN services is their complexity, especially when used in large environments with a high number of users. The attack surface of VPN protocols like IPsec is also significant [64] which adds to the involved risks of operating a VPN service that implements the IPsec protocol. As an alternative to traditional VPN implementations, the Wireguard VPN protocol uses a new approch to VPN security [65]. The Wireguard protocol is significantly less complex then any alternative VPN implementation and used modern cryptographer algorithms. Additional attributes like high reliability make this protocol a good choice for modern VPN services. Within this example IT infrastructure, the Wireguard VPN protocol will be used as an initial access gateway for all internal end devices. To keep the implementation complexity as low as possible, the VPN connection will not require user authentication, but will only limit the access to internal service to clients with a valid VPN key. The client key material can be stored in the TPM module of the end device to prevent the extraction of secret keys by malware. In addition to the access proxy, a central user identity service is required, to provide the authentication and authorization interfaces. A suitable product to fulfill this role, can be found in Keycloak [60] or the commercial version Red Hat Single Sign-On (RH-SSO) [66]. User information can also be managed by a external LDAP server such as FreeIPA or Microsoft Active Directory. The LDAP protocol is only used between the LDAP server and the Keycloak or RH-SSO server. Application services are not permitted to communicate with LDAP servers like FreeIPA at all, to avoid exposing this protocol interface to other services. The reasoning for exposing network protocols to different areas of the infrastructure is based on the associated attack surface of the protocols. Network protocols like HTTP and SSH have been used on the internet for decades and are fairly well audited and therefore are deemed suitable for production critical services. Other protocols like NTP and DNS are also well established and commonly provided on the internet. While security issues in these protocol implementations are more common, the information provided by these protocols are usually public. The confidentiality of the provided information by NTP and DNS servers is not as important as their integrity, as any manipulation of the information provided can lead to a number of possible attacks [67]. Other protocols, like SMB and RDP, that are traditionally used in conventional infrastructures, provide a significant attack surface and should be avoided. The internal infrastructure services like DNS and NTP can be provided, using secure implementations like NTPsec and DNSSEC, by a central network security device like a pfSense Firewall appliance [68]. This network device will also provide the Wireguard VPN gateway service. As part of the VPN service, information traditionally provided by DHCP servies, such as dynamic internal IP addresses, can be distributed to the client devices.

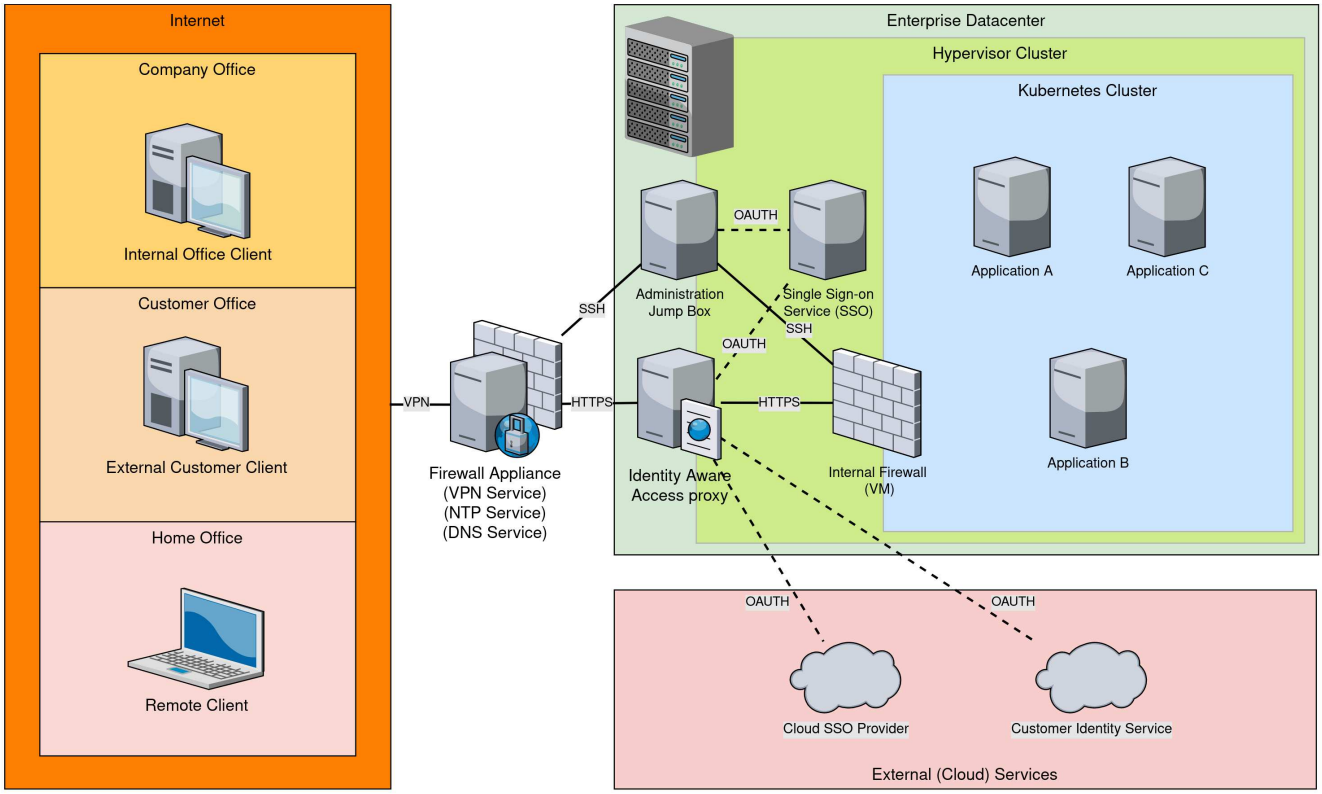


Figure 5: Zero Trust Architecture

In addition to NTP and DNS information, a HTTP service will be provided within the outer zone, between the VPN service and the access proxy. This service will also only provide public information and serve as a packet mirror for the client operating system. This makes it possible to implement a client software compliance check, that requires all clients to

have up to date software versions installed, before successful authentication with the central proxy is allowed. An additional, virtual pfSense firewall will be used to prevent any access to internal systems, allowing only HTTPS connections from the central access proxy and SSH connections from the SSH jump server. Additionally, a log collector needs to be allowed to receive syslog traffic, used mutual TLS certificates for device authentication and confidentiality. All internal applications, that are not part of the core infrastructure, such as those provided by the pfSense servers, are implemented as containers within a Kubernetes cluster. The reason behind this design decision is, that the Kubernetes service already implements a network security architecture, that is similar to the micro segmentation zero trust approach [69]. The central virtual firewall, will therefore only need to manage the traffic between the Kubernetes loadbalancers, serving as reverse proxies for the internal container applications, and the access proxy, as well as the required access between the administration interfaces and the SSH jump server. All other network control mechanisms are implemented using the native Kubernetes features and do not require additional firewall rules. This makes both the implementation and maintenance of the firewall services straightforward and simple. While the Pomerium access proxy implements a identity based zero trust approach at the front end, the Kubernetes cluster can provide a network micro segmentation zero trust approach for the backend services. The use of Kubernetes will add some amount of complexity o the infrastructure that will itself present an additional attack surface, that will need to be hardened and monitored. Any network switch devices that are used to connect the hardware components on the second OSI layer, are required to implement private Vlans, to enforce all traffic to be managed by the described network security services. By using this network security approach, conventional DMZ designs are no longer required and firewall management requires only a static rule set with low maintenance. The central security requirement for zero trust, that demands all users and services need to be authenticated and authorized, while permissions are set for least privileges, is therefore implemented by these listed technologies. Since all connections permitted by the access proxy for authenticated and authorized users, need to use the HTTPS protocol, only encrypted traffic is allowed within the internal network.

5.3.3 Central Software Management and End Device Security

While the described zero trust and authentication technologies provide the central security advantage compared to conventional IT security architectures, there are a number of remaining security measures that are still significant in this example environment. The used operating systems within the infrastructure, will be Linux Red Hat, as the requirements for modern secure systems are met by this Linux distribution. This is mostly the case, because the administration of Linux systems can be done using automation systems like Ansible [70], that do not require complex protocols like Microsoft environment do [4]. Instead all administration and automation can be managed through the already present SSH services. Since the application services for end users as web based, there is not requirement for specific operating systems either. For additional hardening and compliance checks the Lynis auditing tool is used on all Linux operating systems [71]. This will allow to verify the compliance with central security requirements such as installed updates and hardening measures. For additional security, sandboxing technologies should be deployed for all network applications such as web browsers. This can be implemented using well established tools like SELinux. The reduction of the attack surface on users end devices continues to be one of the most significant security measures, even with a zero trust architecture in place. That is because the end device is used by the users to access all applications and their data within the infrastructure. A compromised client will therefore circumvent all security measures designed to prevent unauthorized access to data and systems. As the root of trust for all security sensitive actions within the network, end device security is a most crucial component within the security architecture. The key measure to reduce the risk of compromised endpoint security is the reduction of the available attack surface. This involves for example, restrictions of end user applications that are prone to lead to phishing and misuse, such as email. Other applications like web browser need to be tightly sandboxed and monitored. The amount of available end user applications and operating system hardening is a significant factor for the overall attack surface of end devices. To implement a secure end device security architecture, one recommendation involves the use of multiple devices for different use cases and security levels. Separating devices for internal communication and general use, from those used to manage services using privileged accounts is an established security measure [72]. The exact strategy for privileged access management can vary and does not always involve dedicated devices. This is especially the case in environments where users need to access a number of different systems with different security levels. Using dedicated separated devices in these environments is not always an efficient measure and is both expensive and bothersome for the users. Using virtual machines and virtual desktop environments for administrating services of different security levels is one common alternative approach. However the security gained by this approach is very limited when the end users device is still used for common use cases such as web browsing or email communication. Implementing a thin client policy that is based on end user devices with a minimal attack surface and using cloud based services can improve this approach. Since access to web based cloud services or remote desktops still involves complex protocols and applications, such as web browsers and remote desktop clients, there is still a significant attack surface involved. An approach to endpoint security that involves cost effective and convenient measures while also improving the security of end devices when working on different security environments, can be archived with an architecture as implemented by Qubes OS [55]. The Qubes operating system is based on a hypervisor approach, that separates different VMs for different applications and security levels. By using the Xen hypervisor as the base operating system, the security architecture is based on a very small, minimal codebase that needs to be trusted by all other VMs and applications. Since no end user applications are running on the hypervisor itself, the attack surface on the base system itself is minimal. Inside the Qubes VMs can run a number of common operating sytems, such as Linux or Windows. Hardening and security measures inside thes VMs is still relevant, as it is possible to attack the hypervisor through a fully compromised VM, even with only a small attack surface available. By using the Qubes OS approach, it is posisble to use multiple end user operating ssytems and applications of different trust and security levels on the same device, without a significantly lower end device security as would be gained by using different devices.

5.3.4 Intrustion Detection, Integrity Monitoring and Central Logging

In case a infrastructure component, like a server or end user device has been successfully attacked and compromised, it is crucial to be able to detect the compromise and possible following lateral intrusion attempts. To make this possible a SIEM application can help detect suspicious behavior and possible compromised clients. It crucial for the effectiveness of SIEM applications to receive all security relevant information within the network, that could indicate security breaches. This also allows to retrace security events after a security breach has been detected and follow up on the activity of the malicious actor. There are two kinds of sources for such information. The first is active security logging utilities, such as auditd, that can be configured to trace activity on system resources like users or files based on defined rules. In combination with logs from user space applications, it is possible to trace all activity on a system, even after a compromise occurred, as long as the logging information is stored on a secure remote system. The second measure is integrity checks using cryptographic hash sums. An application like AIDE can be used to calculate the has of critical system files, which allows the detection of manipulated files, assuming the checksum databases are also stored on a secure remote location that cannot be accessed by the users system after the information has been stored. Scanning physical end devices for manipulated or malicious files can be problematic if the device is already compromised. The weaknesses of traditional antivirus applications has been known for some time [73] and the same weaknesses apply to many aspects of intrusion detection since this is a error prone approach to security that requires many resources. The most significant issue with intrustion detection as well as antivirus solutions, it that they are based on a black list security approach. Monitoring files and activities on what is known to be malicious, is predestined to fail in an environment where new malicious programs and attack vectors are discovered every day, often weeks after they have been deployed. Using a whitelist approach by application whitelisting and artificial ignorance [74] can improve intrustion detection measures significantly. While intrusion detection by auditing sensitive system activities and monitoring log files is one integral part for end point security, integrity monitoring can be an effective and efficient solution in some environments. One condition to implement integrity monitoring securely is a trusted base system that is independent of the monitored system. This can be the case with virtual machines and hypervisors. While it is not recommended to implement any additional functionality into the hypervisor, to keep the attack surface of this bases operating system minimal, virtual machines can be stopped and extracted to be scanned by remote security applications. Antivirus and similar security software rely on their ability to detect a threat before the system itself is compromised. If the system and the used kernel is however compromised, the security software, like antivirus, relies on the infected system itself. Any competent malware that infected the operating systems kernel itself can interfere with the security application and render it useless. By using a system like Qubes OS that runs all user applications inside virtual machines, it is possible to extract these VMs even during active operation, and scan them by an independent system for threats, malicious activity and integrity. It is also possible to extract the current state of memory inside the VM, which can lead to a detection of high level malware that resides in memory only and is never written to persistent storage. When using central software management applications, all used systems should be consistent with the specified template. Detection of any discrepancy within the extracted VMs, is therefore efficient and reliable. While this approach can be implemented on virtual systems, it is not easily possible to use this kind of fully automated process with traditional bare metal devices as no independent system exists that can extract the operating system environment. On the other hand, container based systems such as Docker, are especially suitable to be monitored for integrity. Since containers are based on images and should not change during normal operation, any anomaly within the integrity of a container can easily be detected. Since the describes example IT infrastructure includes a Kubernetes cluster for most server applications, effective integrity monitoring should be a central part in its operations.

5.3.5 Automatic Remote Backup

Many common security threats like ransomware targets the valuable user data on the compromised device to make it inaccessible. To prevent any major damage to the data stored on end devices, an automatic backup system, that saves the users data in a way that avoids deletion or manipulation is crucial. Systems like git have been using an approach that involves appending changes rather then overwriting existing files. This effectively prevents the threat posed by malware like ransomware without major additional effort. For the example environment, it is recommended to implement a git server for backup purposes. All clients contain a script that saves the users data to the remote server without user interaction. Alternatively the data can then be saved incremental on a remote SFTP server running OpenSSH, using the rsync application. This can be implemented in a way that any write permissions are removed from the saved data during the operation. While the users are now still able to read their backup data for recovery purposes, it is no longer possible to manipulate the backup data. While these backup strategies for end users devices are simple to implement and manage, they are lacking attributes like encryption. An advanced and secure approach to automatic backup management is available with duplicity [75]. In environments where cloud services may be used, the highly secure Tarsnap [76] is a valid alternative.

5.3.6 Device Data Encryption

While the previous measures keep the data and infrastructure secure from network based threats, the involved systems also need to be protected from physical threats like the evil maid attack. If an adversary has physical access to a device, it would be possible to access and manipulate the stored information by mounting the persistent storage to another device, that the attacker controls. To prevent the extraction of sensitive information, all data needs to be encrypted using full disk encryption (FDE). Since FDE required the user to enter a shared secret, the encryption keys should not be derived from a password, to avoid the already discussed security issues with password authentication. Instead the key material should be saved on the TPM of the device. The user will be required to enter a PIN code to unlock access to the device, with the TPM serving as the trusted element to protect the encryption keys. instead of the entropy of a user defined encryption password, the security of the disk encryption systems depends on the security of the TPM. To further prevent the exploitation of the well known weaknesses of full disk encryption systems, the verification of the boot stack is also required. This should also be implemented by using the devices TPM and the secure boot process. While secure boot also has known security issues [77], it is an available and easily implemented security feature that makes attacks on the physical device harder.

6 Conclusion

The solutions presented in this paper are designed to solve long existing security issues in conventional IT security architectures. Traditional password based authentication has been known to be weak for several decades by now, and the conventional DMZ based security architectures were designed to protect a network based on assumptions that are no longer valid in modern IT environments. The proposed architecture has illustrated how the zero trust paradigm in combination with modern, phishing resistant FIDO2 authentication can mitigate most of the common risks that are present in modern IT infrastructures. Implementing these measures effectively requires significant changes to the IT environment of any organization intending to adapt this approach. However the reduction in complexity as well as management efforts compared to traditional architectures, will make this an economic investment as well in addition to a significant improvement of the IT security. While there are still ways to compromise a network even with a zero trust architecture and FIDO2 authentication in place, both the likelihood and the consequences of a successful compromise are significantly lower. With Google spearheading these new security strategies very successfully, it should serve as a good example of how well designed security measures can significantly improve the security of an IT infrastructure with limited resources and efforts.

7 Further research

This paper has presented a solution to modern network security by using the zero trust architecture and solid passwordless FIDO2 authentication. While the security of IT infrastructures implementing the describes measures is significantly improved, there are a number of remaining security risk, that need effective and efficient solutions. While the remaining risks have been discussed to some extend, many issues and solutions require further research. Endpoint security is a widely known issue and yet the solution present today, like antivirus applications, are known to be ineffective and inefficient. New security approaches to endpoint security based on the reduction of software attack surfaces and integrity monitoring could lead to the next steps in the development of more secure IT infrastructures. Artificial ignorance based SIEM approaches also promise to be a possible solution to intrusion detection and security monitoring.

DECLARATION

Availability of data and materials: Not applicable
Funding: Not applicable
Acknowledgements: Not applicable

References

[1] Industrie und Handelskammer NRW. Digitale transformation und industrie 4.0. Technical report, IHK NRW, NRW, Germany, 10 2015.

[2] Muhammad Kiru and Aman Jantan. *Ransomware Evolution: Solving Ransomware Attack Challenges*, pages 193–229. Apple Academic Press, 01 2020.

[3] The untold story of notpetya, the most devastating cyberattack in history. `https://www.wired.com/story/notpetya-cyberattack-ukraine-russia-code-crashed-the-world/`, Online; accessed 5-November-2020.

[4] Shou-Ching Hsiao and Da-Yu Kao. The static analysis of wannacry ransomware. In *20th International Conference on Advanced Communication Technology (ICACT)*, pages 153–158. IEEE, 02 2018.

[5] Jagmeet Aidan, Harsh Verma, and Lalit Awasthi. Comprehensive survey on petya ransomware attack. In *2017 International Conference on Next Generation Computing and Information Systems (ICNGCIS)*, pages 122–125. IEEE, 12 2017.

[6] Dylan Davis Sean Dillon. Eternalblue - exploit analysis and port to microsoft windows 10. Technical report, Sunnyvale, CA, USA, 6 2017.

[7] Rory Ward and Betsy Beyer. Beyondcorp: A new approach to enterprise security. *usenix, ;login:*, Vol. 39, No. 6:6–11, 2014. Portland, CA, USA.

[8] Sun Jingyao, Sonali Chandel, Yu Yunnan, Zang Jingji, and Zhang Zhipeng. Securing a network: How effective using firewalls and vpns are? In *Advances in Information and Communication*, pages 1050–1068. Springer, Springer International Publishing, 01 2020.

[9] Kirori Mindo, Caroline Sogomo, and Nickson Karie. Analysis of network and firewall security policies in dynamic and heterogeneous networks. *International Journal of Advanced Research in Computer Science and Software Engineering*, 6:141–146, 04 2016.

[10] Service overview and network port requirements for windows. `https://docs.microsoft.com/en-in/troubleshoot/windows-server/networking/service-overview-and-network-port-requirements`, Online; accessed 5-November-2020.

[11] Gurudatt Kulkarni. Cloud security challenges. In *2012 7th International Conference on Telecommunication Systems, Services, and Applications (TSSA)*, pages 88–91, 10 2012.

[12] Remote desktop - allow access to your pc. `https://docs.microsoft.com/en-us/windows-server/remote/remote-desktop-services/clients/remote-desktop-allow-access`, Online; accessed 5-November-2020.

[13] Ben Jones Matt Boddy and Mark Stockley. Rdp exposed - the threat that's already at your door. Technical report, Sophos, 2019.

[14] Niels Provos, Markus Friedl, and Peter Honeyman. Preventing privilege escalation. In *USENIX Security Symposium*, 09 2003.

[15] Moshe Zviran and William Haga. Password security: An empirical study. *J. of Management Information Systems*, 15:161–186, 03 / 1999.

[16] BSI. The state of it security in germany in 2019. Technical report, Federal Office for Information Security, Bonn, Germany, 10 2019.

[17] Syed Zulkarnain Syed Idrus, Estelle Cherrier, Christophe Rosenberger, and Jean-Jacques Schwartzma nn. A review on authentication methods. *Australian Journal of Basic and Applied Sciences*, 7:95–107, 06 / 2013.

[18] Sarah Miller Al Pascual, Kyle Marchini. State of authentication report. Technical report, JAVELIN, 2017.

[19] State of password and authentication security behaviors report. `https://www.yubico.com/blog/yubico-releases-the-2019-state-of-password-and-authentication-security-behaviors-report`, Online; accessed 5-November-2020.

[20] Vasilis Katos, Shahin Rostami, Panagiotis Bellonias, Nigel Davies, Agnieszka Kleszcz, Shamal Faily, Arnolnt Spyros, Alexandros Papanikolaou, Christos Ilioudis, and Konstantinos Rantos. State of vulnerabilities 2018/2019. Technical report, enisa - european union agency for cybersecurity, 12 2019.

[21] Jason Hong. The state of phishing attacks. *Commun. ACM*, 55:74–81, 01 2012.

[22] Scott Rose, Oliver Borchert, Stu Mitchell, and Sean Connelly. Zero trust architecture. *NIST Special Publication 800-207*, 02 2020.

[23] Stephan Wiefling, Markus Dürmuth, and Luigi Lo Iacono. More than just good passwords? a study on usability and security perceptions of risk-based authentication. *arXiv preprint arXiv:2010.00339*, abs/2010.00339, 10 2020.

[24] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. A formal security analysis of the signal messaging protocol. In *2017 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 451–466, test123, 04 2017.

[25] Monique Ogburn, Claude Turner, and Pushkar Dahal. Homomorphic encryption. *Procedia Computer Science*, 20:502–509, 12 2013.

[26] Top zero trust security solutions. `https://www.esecurityplanet.com/products/zero-trust-security-solutions.html`, Online; accessed 5-November-2020.

[27] Katha Chanda. Password security: An analysis of password strengths and vulnerabilities. *International Journal of Computer Network and Information Security*, 8:23–30, 07 2016.

[28] Alena Naiakshina, Anastasia Danilova, Christian Tiefenau, Marco Herzog, Sergej Dechand, and Matthew Smith. Why do developers get password storage wrong? a qualitative usability study. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA, 08 2017. Association for Computing Machinery.

[29] Carlisle Adams. *Personal Identification Number (PIN)*, pages 927–927. Springer US, Boston, MA, 01 2005.

[30] J. Wonglimpiyarat. *Strategies of Competition in the Bank Card Business: Innovation Management in a Complex Economic Environment*, chapter ., page 124. Sussex Academic Press, 2004.

[31] Konstantinos Markantonakis, Keith Mayes, and F. Piper. Smart cards for security and assurance. In *Managing Information Assurance in Financial Services*, pages 166–189. IGI Global, 01 2007.

[32] Totp: Time-based one-time password algorithm. `https://tools.ietf.org/html/rfc6238`, Online; accessed 5-November-2020.

[33] Has two-factor authentication been defeated? a spotlight on 2fa's latest challenge. `https://blog.malwarebytes.com/cybercrime/2019/01/two-factor-authentication-defeated-spotlight-2fas-latest-challenge/`, Online; accessed 5-November-2020.

[34] Markus Jakobsson. Two-factor in authentication – the rise in sms phishing attacks. *Computer Fraud and Security*, 2018:6–8, 06 2018.

[35] A brief history of biometrics. `https://bioconnect.com/a-brief-history-of-biometrics/`, Online; accessed 5-November-2020.

[36] Nalini Ratha, Jonathan Connell, and Ruud Bolle. *Secure Fingerprint Authentication*, pages 263–288. Springer US, 07 2011.

[37] Stephen J Tipton, Daniel J White II, Christopher Sershon, and Young B Choi. ios security and privacy: Authentication methods, permissions, and potential pitfalls with touch id. *International Journal of Computer and Information Technology*, 3(03), 2014.

[38] Andrew Bud. Facing the future: the impact of apple faceid. *Biometric Technology Today*, 2018:5–7, 01 2018.

[39] Chaos computer club breaks apple touchid. `https://www.ccc.de/en/updates/2013/ccc-breaks-apple-touchid`, Online; accessed 5-November-2020.

[40] Roger Grimes. The many ways to hack 2fa. *Network Security*, 2019:8–13, 09 2019.

[41] Fido2: Webauthn and ctap. `https://fidoalliance.org/fido2/`, Online; accessed 5-November-2020.

[42] Web authentication: An api for accessing public key credentials level 1. `https://www.w3.org/TR/webauthn/`, Online; accessed 5-November-2020.

[43] Client to authenticator protocol (ctap). `https://fidoalliance.org/specs/fido2/fido-client-to-authenticator-protocol-v2.1-rd-20191217.html`, Online; accessed 5-November-2020.

[44] Google: Security keys neutralized employee phishing. `https://krebsonsecurity.com/2018/07/google-security-keys-neutralized-employee-phishing`, Online; accessed 5-November-2020.

[45] Fido alliance and w3c achieve major standards milestone in global effort towards simpler, stronger authentication on the web. `https://fidoalliance.org/fido-alliance-and-w3c-achieve-major-standards-milestone-in-global-effort-towards-simpler-stronger-authenticat` Online; accessed 5-November-2020.

[46] Support for fido2: Webauthn and ctap. `https://fidoalliance.org/fido2/fido2-web-authentication-webauthn/`, Online; accessed 5-November-2020.

[47] The yubikey. `https://www.yubico.com/products/`, Online; accessed 5-November-2020.

[48] Nitrokey fido2. `https://shop.nitrokey.com/shop/product/nk-fi2-nitrokey-fido2-55`, Online; accessed 5-November-2020.

[49] Titan security key. `https://store.google.com/product/titan_security_key`, Online; accessed 5-November-2020.

[50] Usb-dongle authentication. `https://www.dongleauth.info/`, Online; accessed 5-November-2020.

[51] Expanded support for fido authentication in ios and macos. `https://fidoalliance.org/expanded-support-for-fido-authentication-in-ios-and-macos/`, Online; accessed 5-November-2020.

[52] News: Your google android 7+ phone is now a fido2 security key. `https://fidoalliance.org/news-your-google-android-7-phone-is-now-a-fido2-security-key/`, Online; accessed 5-November-2020.

[53] Identity-aware proxy overview. `https://cloud.google.com/iap/docs/concepts-overview`, Online; accessed 5-November-2020.

[54] Security assertion markup language (saml). `http://xml.coverpages.org/saml.html`, Online; accessed 5-November-2020.

[55] Oauth 2.0. `https://oauth.net/2/`, Online; accessed 5-November-2020.

[56] Openid. `https://openid.net/connect/`, Online; accessed 5-November-2020.

[57] Identity-aware proxy. `https://cloud.google.com/iap`, Online; accessed 5-November-2020.

[58] John Kindervag. Build security into your network's dna: The zero trust network architecture. *Forrester Research Inc*, pages 1–26, 2010.

[59] Introducing ad fs 2.0. https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/ff641697(v=ws.10), Online; accessed 5-November-2020.

[60] Keycloak about. https://www.keycloak.org/about, Online; accessed 5-November-2020.

[61] What is pomerium? https://www.pomerium.com/docs/, Online; accessed 5-November-2020.

[62] Openssh 8.2 released. https://lists.mindrot.org/pipermail/openssh-unix-announce/2020-February/000138.html, Online; accessed 5-November-2020.

[63] Openssh proxies and jump hosts. https://en.wikibooks.org/wiki/OpenSSH/Cookbook/Proxies_and_Jump_Hosts, Online; accessed 5-November-2020.

[64] Günter Schäfer. *IPSec Security Architecture*, pages 195–242. John Wiley and Sons, Ltd, 03 2006.

[65] Jason Donenfeld. Wireguard: Next generation kernel network tunnel. In *NDSS*, San Diego, California, USA, 01 2017. Proceedings of the Network and Distributed System Security Symposium.

[66] Red hat single sign-on. https://access.redhat.com/products/red-hat-single-sign-on, Online; accessed 5-November-2020.

[67] Jörg Schwenk. *DNS Security*, pages 195–215. Springer, 2010.

[68] pfsense. https://www.pfsense.org/, Online; accessed 5-November-2020.

[69] Securing a cluster. https://kubernetes.io/docs/tasks/administer-cluster/securing-a-cluster/, Online; accessed 5-November-2020.

[70] Ansible. https://www.ansible.com/, Online; accessed 5-November-2020.

[71] Lynis security auditing tool. https://cisofy.com/lynis/, Online; accessed 5-November-2020.

[72] Privileged access workstations. https://docs.microsoft.com/en-us/windows-server/identity/securing-privileged-access/privileged-access-workstations, Online; accessed 5-November-2020.

[73] Antivirus tools are a useless box-ticking exercise says google security chap. https://www.theregister.com/2016/11/17/google_hacker_pleads_try_whitelists_not_just_bunk_antivirus_ids/, Online; accessed 5-November-2020.

[74] Artificial ignorance. https://searchenterpriseai.techtarget.com/definition/artificial-ignorance, Online; accessed 5-November-2020.

[75] duplicity. http://duplicity.nongnu.org/, Online; accessed 5-November-2020.

[76] Tarsnap. https://www.tarsnap.com/, Online; accessed 5-November-2020.

[77] Boot hole vulnerability - grub 2 boot loader - cve-2020-10713. https://access.redhat.com/security/vulnerabilities/grub2bootloader, Online; accessed 5-November-2020.