EDA263 - Computer Security, Assignment 1

Robert Grzelka, 9110091c759

Introduction

Topics of Laboratory

In this laboratory assignment, we study users identification and authentication in terms of security. Also we investigate a UNIX application over the requirements to runs with permissions, especially higher privileges. In this lab we will work on linux system, so if not mentioned otherwise, all specifics will correspond to linux.

The assignment consists of two parts:

- one to be done at home with questions to answer
- another to be done in a laboratory that require to:
  - implement a login program for UNIX
  - demonstrate solution to TA.

Report of assignment is covering topics about:

- identification and authentication
- methods/types of authentication
- vulnerability understanding, recognition and avoidance
- requirements of a safe UNIX application that runs with higher privileges
- protection for attempts of unauthorized login with various methods

Code of assignment is about "UNIX login implementation" that containing protection methods like:

- buffer overflows protection during stdin
- protection from signal interruption from UNIX hotkeys
- brute force dictionary password guessing protection
- leak less login failures (in case of failed logins, no default stderr)
- users database records with refreshing values of password age and login failures
- password aging over successful logins
- account lock-down over failed login attempts
- password encryption with crypt lib
- password salt
- using sleep to delay login attempts

Preparations

Preparation for this laboratory required from us to review content of few readings:

- Stallings & Brown - Computer Security [1], Part One [1.1]:
  - Chapter 0 of Readers and Instructors Guide [1.1.0]
  - Chapter 1 of Overwiew [1.1.1]
  - Chapter 3 of User Authentication [1.1.3]
- EDA263 - Computer Security, 2018 Offprint [2], Chapter 1 [2.1] borrowed from "Computer Security, Stallings&Brown, 978-0-13-513711-6, Chapter 23.2 of Linux Security Model".

Generally from [1] we had to read only [1.1.3], but as said in [1] it is required to read all chapters in linked list, thought Chapter 2 was skipped.

Computer Security Introduction

To answer questions and write code of program, we will overview Computer Security Technology and Principles that are technical areas that must underpin any effective security strategy. We will look at [1.1] for specific technical areas of computer security:

- identification, authentication, and authentication technologies
- discretionary versus mandatory access control
- rule-based and role-based access control

This laboratory not only includes topic of User Authentication but also this of Access Control, which in reading of [1.1.4] has textbook coverage of 1 out of 10 CISSP certification Domains, domain of "Access Control" stated as:

- a collection of mechanisms that work together to create a security architecture to protect the assets of the information system

CISSP domain of Access Control contain topics like:

- Identification, authentication, and authorization technologies
- Discretionary versus mandatory access control models

Its worth noted that CISSP certification is required by many financial institutions according to [1]. Besides CISSP, there is also similar NSA/DHS certification fully covered in [1]. This laboratory contains partially one of its listed core Knowledge Units of "Cyber Defence" with topics:

- access control,
- cryptography,
- firewalls,
- intrusion detection systems,
- malicious activity detection and countermeasures,
- trust relationships,
- defense in depth.

Vulnerabilities, threats and protection mechanisms

In information and information systems security we need to ask tree basic questions: 1. What assets do we need to protect? 2. How are those assets thretened? 3. What can we do to counter those threats?

It may be seen that system having some assets or other usages may have its vulnerabilities used as threats to penetrate this system by an attacker. Thats why we introduce some means of data security which we set up according to importance/assets. We secure system with some objectives after definition of Computer Security:

- "The protection afforded to an automated information system in order to attain the applicable objectives of preserving the integrity, availability and confidentiality of information system resources (includes hardware, software, information/data, and telecomunications)."

System security is set in way that make it safe against intruders under objectives model ie. NIST standard FIPS 199 called CIA triad: Confidentiality, Integrity, Availability (with additional picturesque components of: Authenticity, and Accountability). Thus we have to deal with security problems of system, especially with network security attacks. For this purpose we set major defence lines, defend in depth by splitting security on layers of outside, boundary and inside of system.

Based on standards, guidelines and security certifications we may follow model of protection mechanism layers, and practical security protection mechanisms principles. Besides that we may get some hints from list of popular security tools. This list also let us tell what kind of security is less/more vulnerable and mature.

Proper usage of protection mechanisms to enhance security require from system administrators knowledge about basic information, methods and how-to about tools them-selfs. Knowledge about system objectives and available protection tools has to be matched with assets of the system what can be done by looking at security terminology flowchart.

When dealing with already implemented or planned system, admins will be thrown into challenges of computer security. Where they may struggle with breaches into mechanisms that should be impenetrable, ie. with threats for air-gapped networks. We have a lot of protection mechanisms examples with their usages, pros and cons, that proper config may be challenged itself.

Below we will shortly describe general information about system security, then we will smoothly pass to protection mechanisms focused on authentication methods.

Certification guidelines followed

CISSP

CISSP (Certified Information Systems Security Professional) certification:

- from International Information Systems Security Certification Consortium (ISC), that is non-profit org.
- referred as "gold standard" of information security certification
- only universally recognized certification in security industry
- rquired by many organizations, like US Dep. of Def., and many financial organizations
- in 2004 became first IT program with accreditation under ISO/IEC 17024 (General Requirements for Bodies Operating Certification of Person)
- CISSP examination is based on Common Body of Knowledge (CBK), a compedium of IS best practices developed and maintained by ISC
- CBK made up of 10 domains required for CISSP, where in this paper, we focus on first one that is "Access Control".

CISSP Access Control is defines as:

- A collection of mechanisms that work together to create a security architecture to protect the assets of the information system.

Key topics of CISSP "Access Control":

- identification, authentication and authorization technologies
- discretionary versus mandatory access control models
- rule based and role based access control

List of CISSP Domains:

- Access Control
- Application Development Security
- Bussiness Continuity and Disaster Recovery Planning
- Cryptography
- IS Governance and Risk Management
- Legal, Regulations, Investigations and Compliance
- Operations Security
- Physical (Environmental) Security
- Security Architecture And Design
- Telecomunications and Network Security

NSA/DHS

NSA/DHS comes from National Centers of Academic Excellence in Information Assurance/Cyber Defense (IA/CD) sponsored by US NSA and US Dep. of Homeland Security (DHS). Its goal is to:

- reduce vulnerability in national information infrastructure by promoting higher education and research in IA
- producing growing number of profesionals with IA expertise in various disciplines

Defined with set of Knowledge Units for 2/4 year institutions supported in curriculum. In area of CS the 2014 Knowledge Units lists 6 core items:

1. Cyber Defense:
    - access control, cryptography, firewalls, intrusion detection systems, malicious activity detection and countermeasures, trust relationships, defense in depth
2. Cyber Threats:
    - types of attacks, legal issues, attack surfaces, attack trees, insider problems, threat information sources
3. Fundamental Security Design Principles:
    - list of 12 principles in this paper found in section of "Security Functional Requirements"
4. Information Assurance Fundamentals:
    - threats and vulnerabilities, intrusion detecton and prevention system, cryptography, access control models, identification/authenitication, audit
5. Databases:
    - overwies of databases, database access controls, security issues of inference

Book [1] gives externsive information about this areas, but in this lab we will focus on conten of items like 1, 2, 4.

ACM/IEEE Computer Society Computer Science Curricual 2013

CS2013 (Computer Science Curricual 2013) of IAS (Information Assurance and Security) Knowledge Area:

- created by ACM (Association for Computing Machinery) and IEEE-CS (Computer Science Society of Institute of Electrical and Electronics Engineers)
- first revision from 2001 with many experts involwed
- from consensus that it is needed to add new Knowledge Area of Information Assurance and Security (IAS)
- have concept of assurance and security for complete perspective
- IAS is domain set of technical and policy controls and processes intended to:
    - protect information by keeping theri CIA triad, authentication and providing for non-repudation
    - give assurance carring attestation that current and past processes and data are valid
- divides course work on: Core Tier 1, Core Tier 2, Elective,
- IAS area includes CS2013 3 from Tier 1, 5 from Tier 2 and number of Elective topics with subtopics.

IAS Knowledge Units:

1. Tier 1:
    - Foundational Concepts in Security:
        - CIA triad
        - Risk, threats, vulnebralities, attack vectors
        - Authenitication and authorization, access control (mandatory vs discretionary)
        - Trust and trustworthiness
        - Ethics
    - Principles of Secure Design:
        - Least privilege and isolation
        - fail-safe defaults
        - open design
        - end-to-end security
        - defense in depth
        - security by design
        - tension between security and other design goals
    - Defensive Programming:
        - input validation and data sanization
        - choice of programming language and type-safe language
        - examples of input validation and data sanization errors (buffer, overflow,s, integer errors, SQL injection, XSS vulnerability)
        - race conditions
        - correct handling of exceptions and unexpected behaviours
        - correct usage of third party components
        - effective deploying security updates
2. Tier 2:
    - Principles of Secure Design:
        - complete mediation
        - use of vetted security components
        - economy of mechanisms (reducing trusted computing base, minimze attack surface)
        - usable security
        - security composability
        - prevention, detection and detterence
    - Threats and attacks:
        - attacker goals, capabilities, motivation
        - malware
        - DOS (denial of service) and distributed DOS
        - social engineering
    - Network security:
        - network specific threats and attack types

        - use of cryptography for data and network security
        - architectures for secure networks
        - defense mechanisms and countermeasures
        - security for wireless, cellular networks
    - Cryptography:
        - basic cryptography terminology
        - cipher types
        - overview of math. preliminaries
        - public key infrastructure

System penetration

Systems that are meant to be secure and private may get under intrusion or penetration. It will start with attack attempting to cause a breach in this system or to compromise it. Successful attack will end with breach in the system resulting in violation of the security policy of a system.

System threats

Threats as penetration attempts are usually performed using some vulnerability of system where it is open for attack (more or less). This usually hidden treats are giving undesired, negative consequences for the system. We protect system against attacks with some countermeasures, controls and restricted permissions.

Data Security

This brings us to security of data. We may define it with triad of Confidentiality, Integrity, Availability (CIA), that we see as key objectives of CS with additional Authenticity and Accountability (included by FIPS199).

Confidentality:

- Assures:
    - data confidentality: data privacy, not avaiable or disclosed to others
    - privacy: control or influence over individuals related information collection and storage, disclosure units
- Loss:
    - is unauthorized disclosure of information

Integrity:

- Assures:
    - data integrity: information and programs are changed with proper authorization
    - system integrity: unimpaired functionality of system without deliberate or inadvertend uauthorized manupulation
- Loss:
    - is unauthorized modification or descrution of information

Availability:

- Assures:
    - that system work promptly without service denials to authorized users
- Loss:
    - is disruption of access to or use of information or and information system

Authenticity:

- its property of being genuine and able to be verified and trusted
- confidence in the validity of a transmission, a message, or message orginator
- verification that users who they say they are and system input came from trusted source

Accountability:

- its goal is to trace actions of entity uniquely to that entity
- this supports nonrepudation, deterrence, fault isolation, intrusion detection and prevention, after-action recovery and legal action
- in case of insecurity

- ability to trace a security breach to responsible party
- keeping records of breach for later forensic analysis to trace breach or aid in transaction disputes

CIA triad breach

With low, moderate and high level of impacts on organizations or individuals, there are few examples worth to note of cases if security breach will happen in any of CIA triad objectives.

Breach of Confidentaility on example of students:

- high: student grade information
- moderate: student enrollment information
- low: directory information like list of students or faculty or departmental list (usually public on website)

Breach of Integrity on example of hospital database about patient allergy:

- doctor should be able to trust that database is correct and current
- in case of authorized employe messing with files that may cause harm to hospital, database need to be restored to trusted basis quickly, plus trace error to responsible
- high: patient allergy information exposed may cause death to patient and hospital under liability
- moderate: online forum with registered users to discuss specific topics
- low: anonymous online poll, bcos inaccuracy and unscientific nature of pools is understood

Breach of Availability:

- more critical component, higher level of avilability required
- high:
  - case of system with authentication services for critical systems, applications and devices
  - interruption will result in inability for custormers to access computing resources and staff to access resources they need to perform critical tasks
  - large finansial loss in employee productivity and potential customer loss
- moderate:
  - public website of university with information for current and prospective students and donors
  - not critical component, but inaviliability will cause some embarrassment
- low:
  - online telephone directory lookup application loss may be annoyance but there are other ways to acces this information like hardcopy or operator

System Intruders

In case of system security, we need to protect it from intruders that may be insiders and outsiders.

Outsiders compose of : hackers, terrorists, thieves, enemy states, spy organizations, anybody!

But insiders are units already having some kind of access to the system, they may be: ordinary users, former users, maintenance personnel (ie. admin), designers (that may leave back doors, Trojan horses, etc.).

Security Problems

We have many kinds of security problems that intruders may try to use to penetrate or compromise system, ie.:

- intrusions, attacks
- eavesdropping (local, transmission, radiation, tempest)
- hardware, hardware errors
- software errors (bugs), software design methods!
- malicious software (virus, Trojan Horses, COTS, etc...)
- inadequate management, deficient configurations
- failure propagation, ie. consequences of security problems in other systems
- naive or ignorant users
- mistakes

Network Security Attacks

We may classify this attacks as passive and active.

Passive attacks are eavesdropping:

- release of message contents
- traffic analysis
- are hard to detect so aim to prevent.

Active attacks do modify/fake data:

- masquerade
- replay
- modification
- denial of service
- hard to prevent so aim to detect.

Threat Consequences, Types of Threat Actions that cause each consequence

We may split threat consequences on types like (with actions):

- Unauthorized Dosclosure threat to confidentality:
  - exposure:
    - release sensitive info, ie. credit cards number
    - result of human, hardware or software error giving user unauthorized data
    - numberous cases, ie. uni posting student confidental information on web
  - interception:
    - common on communication
    - any device on LAN can receive copy of packet intended for another device
    - on internet hacker can gain access to email traffic or other data transfer
  - inference (indirectly acces data):
    - traffic analysis, ie. pattern of trafic or amount between pairs
    - of detailed information from a database by limited access user
    - eneabled by repeated queries that combined results
  - intrusion (circument security to gain data):
    - overcome acces control protection
- Deception threat to integirty of system or data:
  - masquerade (posing as authorized):
    - happend if learned about logon ID and password
    - malicious logic as Trojan horse that appears to do desirable function buy gains access or tricks user into executing other logic
  - falsification (deceive authorized user):
    - ie. student may alter grades on school database
  - repudation (deny responsibility):
    - user denies sending or receiving or possesing data
- Disruption threat to availability or system integrity:
  - incapacitation (attack on availability, disable component)
    - destruct or damage hardware
    - malicious software like Trojan horses, viruses, worms to disable a system or some services
  - corruption (attack on integrity, modify functions):
    - malicious software to make services run in unintended manner
    - gain acces and modify system functions
    - ie. user placing backdoor logic in the system to provide subsequent access by other than usual procedure
  - obstruction (hinder system):
    - interfere with communication by disabling links or altering comm. control information
    - overload system by placing excess burden on communication traffic or processing resources
- Usurpation:

- misappropriation (assume unauthorized control of resource):
  - theft of service
  - DDOS, when malicious software is installed on number oh hosts as platforms to launch traffic attack on target host
  - malicious software makes unauth. use of system resources
- misuse (perform function/service deterimental to security):
  - diabled or thwarted security functions of system by malicious logic or hacker with access

Threats and assets on examples

	Avaiability	Confidentality	Integrity
Hardware	major threat	unencrypted data stolen	
	accidental or deliberate damage or theft		
	most vulnerable to attack		
	least susceptible to automated controls		
	equipment is stolen or disabled, denying service		
	physical and admin measures deal with these threats		
Software	key threat, deletion denying acces to user	unauthorized copy, piracy	modifications causing fail during execution or unintended task
	software configur		viruses and related attacks
Data	files are deleted, denying acces to users	unauthorized read of data	existing files modified
	analysis of statistical data revelas underlying data	new files are fabricated	
Communication Lines and Networks	messages destroyed or deleted	messages are read	messages modified, delayed, reordered or duplicated
	lines or networks unavailable	traffic pattern observed	false messages fabricated

Computer Security - Major Defence Lines

We may see system security as graph composed of components like:

- threat -> system -> user.

To make system more secure, we intend to prevent or reduce the treat of system, also protect system on its boundary from outside interference, set means for system recovery in case of penetration. We need to make system secure under idea of CIA, so we can deliver services to user.

Application of Defence in Depth

Threat of system begins outside the system boundaries. Nearer we go to system and later, deeper we go inside the system, with next layers of defence, all threats should be minimized to zero until they arrive to system resources and system service can be shared to users.

Model of protection mechanisms layers

We use protection mechanisms in three types of layers:

- outside the system in form of prevention:
    - legal means protection
    - deterence
- on boundary of system:
    - shield cables
    - encryption
    - physical protection (ie. locks, cages, etc...)
    - access control
- inside boundary of system in form of internal control:
    - detection and countermeasures mechanisms
    - (anti-)virus programs
    - supervision mechanisms (with response capabilities)
    - intrusion detection (with response capability)
    - encryption of stored data
    - deflection to honeypot outside the system

Security Functional Requirements

Countermeasures to reduce vulnerabilities and deal with threats may be classified and characterized in multiways. We will view them in terms of functional requirements and follow classification defined in FIPS 200 (Minimum Security Requirements for Federal Information and Information Systems) - standard with 17 security areas of protecting CIA tria of information systems including their processing, storing and data transmission.

Protection of system may be seen from "technical measures" or/and "management controls and procedures" points of view. This principles of protection mechanisms can be categorized accordingly, based on FIPS 200.

Technical measures:

- access control
- identification & authentication
- system & communication protection
- system & information integrity

Management controls and procedures:

- awareness & training
- audit & accountability
- certification, accreditation & security assessments
- contingency planning
- maintenance
- physical & environmental protection
- planning
- personnel security
- risk assessment
- systems & services acquisition

Overlapping technical and management:

- configuration management
- incident response
- media protection

Here we focus mostly on Access Control and Identification & Authenitication.

Access control is stated as:

- Limit Information system access to authorized users, processes acting on behalf of authorized users, or devices (including other IS) and to the types of transactions and functions that authorized users are permitted to exercise.

Identification and Autheniticatin:

- Identify information system users, processes acting on behalf of users, or devices, and autheniicate (or verify) the identities of those users, processes, or devices, as a prerequisite to allowing aceces to organizational IS

Fundamental Security Design Principles

Flaws in security and unauthorized actions made it usefull to have set of widely agreed deisng principles helpfull in development of protection mechanisms. NCAE13 sponsored by NSA/DHS list following fundamental security design principles:

- Economy of mechanism
- fail-safe defaults
- complete mediation
- open design
- separation of privilidge
- least privilidge
- least common mechanism
- psychological acceptability
- isolation
- encapsulation
- modularity
- layering
- least astonishment

Of which first eight were proposed in SALT75 and withstood in time.

Economy of mechanism:

- try to eliminate unnecesary complexity,
- easier to verify,
- tend to have fewer exploitable flaws and require less maintanence
- configuration mechanisms are simple, so updating or replacing become less intensive process

Fail Safe default:

- access decisions based on permission rather than exclussion
- exclussions tend to fail by allowing acces, may go long unnoticed in normal use

Complete mediation:

- every access must be checked agains access control mechanism
- not rely on access retrieved from cache
- once user opened file, no check is made to see permissions change
- its resource-intensive and rarely used

Open Desing:

- ie. encryption keys must be secret, but encryption algorithms should be open to public

Separation of Privilidges:

- practice in which multiple privilidge attributes are required to achieve access to a restricted resource
- multifactor user authentication, ie. password and smart card to auth user
- program is divided into parts with limited specific privileges required in order to perform specific task
- ie. removing high privilidge operations to another processs and running it with higher privilidges required to perform its tasks, when day-to-day interfaces are executed in lower privilidges process

Least privilidge:

- every process and every user of the system should operate using the last set of privilidges necessary to perform the task
- ie. role based access control [1.4]
- system security policy can identify and define various roles of users or processes
- system programs or admins should have special privilidges only when necessary otherwise withdrawn for ordinary activities that close door to accidents

Least common mechanism:

- design should minimize functions shared by different users providing mutual security
- reduce unintended communication paths
- reduce amount of hardware and software that all users depend on
- easier to verify any undesirable security mechanisms

Psychological acceptablity:

- security should not interfere unduly with works of users
- but should meet needs of those who authorize access
- should make sense to user to avoid errors
- should be transparent and introduce minimal obstruction

Isolation:

- public access systems should be isloated from critical resources
- processes and files of individual users have isolate process space, memory space, and file space with protection for preventing unauth access
- security mechanisms should be isolated to prevent access to them, ie. isolation of cryptographic software and keys

Encapsulation:

- isolation bases on object oriented functionality

Typical security tools used

- Firewall: 97%
- Anti-Virus Software: 96%
- Anti-Spyware Software: 79%
- Server-Based access control list: 70%
- Intrusion Detection System: 69%
- Encryption for data in transit: 63%
- Encryption for data in storage: 46%
- Reusable account/login password: 44%
- Intrusion prevention systems: 42%
- Log management software: 41%
- Application-level firewall: 39%
- Smartcart/one-time password token: 38%
- Forensics tools: 38%
- Public-Key Infrastructure: 35%
- Specialized wireless security system: 31%
- Endpoint security client software: 30%
- Biometrics: 20%
- Other: 5%

Information, methods and tools to enhance security

General advices about system securing for administrator (and users):

- Know your system
- update it continuously
- supervise it
- make use of available security mechanisms
- alarm reports (CERT, OWASP, hacker-sites, etc...)
- information about "pathes"
- tools for analysis and intrusion detection
- educate the people (specially users)

Security terminology flow chart

System security may be described with graph containing elements like:

- Owners
- security policy or countermeasures
- vulnerabilities
- risk
- adversary or threat agents
- threats
- assets or system resources

Challenges of computer security

1. Security is not as simple as it may appear to the novice:

- possible to attack the security mechanism
- security is not done in isolation from the rest of the system
- mechanisms used to meet requirements of CIA triad, nonrepudation may be complex and need subtle reasoning

2. Development of security mechanism or algorithm we have to consider potential attack on those features. Successful attacks happens by looking in different way to exploit unexpected weakness in mechanism.
3. Bcos prev point, procedures providing services may be counterintuitive. Security mechanism is complex, not so obvious to see from particular requirement that elaborate measures are needed.
4. Its necessary to decide where to use designed security mechanisms, both in physical placement (what point of network require security) or in logical sense (at what layer or layers of ie. TCP/IP should security be placed).
5. Security involve typically more than some algorithm or protocol. Also require from participants to possess some secret info (ie. encryption key), which gives question how to create, distribute and protect this secret. Also reliance on communication protocols behaviour may complicate development of security mechanism, ie. random delays in network or protocols will render time limits in transit of message as useless.
6. Security is a "chess game" between the attacker and the system admin:

- attacker only needs to find single vulnerability to penetrate the system
- while the administrator needs to patch all holes to ensure system security

3. Natural tendency to disregard security problems until a security failure occurs
4. Security is a process of:

- constant monitoring
- long-term perspective

5. Security is often an afterthought:

- added after the system has been designed

6. Some users think security is restricting them in their job
7. Security is the lack of insecurity:

- in system chain weakest points define its security

Example of threats for air-gapped network meant to be secure

Equation Group:

- unit of NSA been around since 2001
- by CIA its name not of group by complex malware suite (tools for hacking),
- labeled as "most advanced that have been seen"
- infects firmware what is impossible to get rid off
- designed to counter air-gapped systems by transfer of information on USB
- its self terminating protocol make it hard to detect
- discovered by Kaspersky

Way of breaching air gapped system:

- attacker infects and air-gapped computer using a USB device
- infected computer displays a modified image containing hidden sensitive data
- graphics card emits FM radio signals containing hidden data
- this signal can be picked by an FM receiver within range of 7[m]
- insider can bring/have an infected device within range of FM signal to receive up to 60Bps of stolen data

Protection mechanisms examples

- hardware protection (computers, servers, CDs, backups, modems, printers)
- usage of authentication (passwords, smartcards, etc...)
- proper access control (read, write, execute, install)
- usage of anti-virus programs
- proper config of firewall
- supervision and intrusion detection mechanisms
- spam filtering (whitelisting, blacklisting, greylisting, etc...)
- isolation of real sensitive networks and computers

Identification and Authentication

Means of identification and authentication are required to keep user privacy in system and avoid unauthorized access to user/admin accounts. It is basic defensive mechanism against others trying to infiltrate systems, bring some damage or simply steal data.

In Linux systems, root is almighty and can manipulate with system and other users, thats why if attacker could get access to root account, then system is lost.

If only one user will be penetrated successfully then it may seem that only his account would be lost when he is not root. But penetrated user may have special permissions, group permissions, be in sudoers group, read only or execute permissions. In this way files of groups that breached user have access to may be lost and all files that he can read will be leaked or in case of execution permission, will be executed.

Ideally identification and authentication should be bullet-proof, but usually it carry bugs, vulnerabilities, etc. that are easily to overlook. There are many methods of identification and authentication of user, that may be split on types: user knows, users has, user is(/do). Different types will have its own pros and cons and may be considered as different layers of protection with specific types of vulnerabilities giving potential risk of breach. If used means of identification and authentication contains bugs then they become threat to security.

In this laboratory we will implement simple program of user identification and authentication by password means, then look for and repair bugs and typical issues in this program that attacker may use to infiltrate user account.

User Auth(entication)

In most CS designs, we typically have user auth as:

- fundamental building block
- primary protection mechanism
- basis for most types of access control and user accountability

RFC4949 defines user auth with:

- The process of veryfying and identity claimed by or for a system entity, consisting of two steps:
  - identification: Presenting and UID to the security system. Where id should be assigned carefully, bcos auth identities are the basis for other security services, such as access control service.
  - verification: Presenting/generating auth information that corroborates the binding between the entity and the identifier.

Each account has its UID (user identificator) that usually comes with password known only to owner of account and to system for verification purpose. Password should come with policy that make it hard to guess or steal from user, by itself its saved in system in hashed (to make it secret) form, where hash itself may not be secret. Admin is managing (ena/disa-bling) user permissions and is auditing activity of user.

Model for Auth

NIST SP 800-63-2 (Electronic Auth Guideline, 09.2013) defines general model for user auth that involves number of entities and procedures set as sequence of user registration:

- applicant apply to registration authority (RA) to become subscriber of a credential service provider (CSP);
- RA is trusted, set and vouches for identity of applicant to CSP
- CSP engages in exchange with subscriber
- depending on details of overall auth system, CSP issues some sort of electronic cedential to subscriber;
- credential is datastructure with authority to binds an identity and additional attrs to a token of subscriber, and can be verified at auth transaction;
- token can be encryption key or encrypted passwd that identifies subscriber, issued by CSP, generated directly by subscriber or given by 3-party.
- token and credential may be used in subsequent auth events.

Means of Auth

We have 3-4 general means of auth of UID used alone or combined:

- User Knows
- User Has
- User Is/Does

All of this methods if come with proper implementation can provide secure auth, but they may have problems like adversary may: guess or steal what User Knows, forge or steal what User Has. User Knows and Has come with administrative overhead for managing passwds and token infos on systems and securing this info. User Is/Does may end with false positives/negatives, deal with user acceptance, cost and convenience.

Risk Assessment for User Auth

There are 3 concepts of security risk assesments related to auth that we wish to relate one with another:

- assurance level
- potential impact
- areas of risk

Assurance Level

It describes degree of certainity if who refers himself as certain user, truly has this identity. We define it as degree of confidence in the:

- vetting process that identify individual to whom has credential issued
- who uses credential is whom it was issued

SP 800-63-2 issues 4 levels of assurance: Level 1, Level 2, Level 3, Level 4

Level 1:

- little or no confidence in asserted ID validity
- typically user supplied ID and password at same time,
- ie. consumer registering to participate in a discussion at a company web site discussion board

Level 2:

- some confidence in ID
- good for wide range of bussiness with public where required is initial ID assertation (verified independently prior any action)
- used secure auth protocols together with one means of auth from (User Knows/Has/Is)

Level 3:

- high confidence in ID
- good to enable clients and employees to access restricted services of high value but not highest
- ie. patent attorner eloctronically submits confidential information to US Patent and Trademark Office
- improper disclosue would give competitors advantage
- require more than one factor of auth (at least two idependent)

Level 4:

- very high confidence
- access restricted services of very high value or for which improper access is very harmfull
- ie. in case of law enforcement database with criminal records, unauth access could rise privacy issues and compromise investigations
- require multiple auth factors and in-person registration

Potential Impact

Concept closely related to assurance level. Defined with FIPS 199 with 3 levels for organizations and individuals in case of breach of security (here, failure in user auth):

- Low: Auth error is expected to have limited adverse effect on organization operations, assets, or individuals, minor damage, loses and harm
- Moderate: serious adverse effect, able to perform primary functions but with reduced effectiveness, big damage and loses, significial harm without death
- High: severe or catastrophic adverse effect, not able to perform one or more primary functions, major damage and loses, loss of life or life threateting injures

Areas of Risk

Mapping between potential impact and level assurance that is satisfactory to deal with potential impact depend on context. Considering case of potential financial loss in case of auth error that ends with unauth access to database. Depending from type of database, impact could be: low, moderate, high.

Low:

- insignificiant or inconsequential unrecoverable financial loss to any party, or at wors to organization liablity.

Moderate:

- at worsk a serious unrecoverable financial loss to any party or a serious organization liability

High:

- severe or catastrophic unrecoverable financial loss to any party or to organization liability.

If potential impact is low, then adequate is assurance level 1. If moderate, then assurance level 2/3. If high, then level 4. Ie. if any impact categories has a potential impact of high, or if personal safety category has a potential impact of moderate high, then level 4 assurance should be made. More examples are described in Table 1.

Table 1 : Maximum Potential Impact for Each Assurance Level

Potential Impact Categories for Auth Errors	**Assurance Level Impact Profiles (1,2,3,4)**
Inconviecence, distress, or damage to standing or reputation	Low, Mod, Mod, High
Financial loss or rganization liability	Low, Mod, Mod, High
Harm to organization programs or intersests	None, Low, Mod, High
Unauth release of sensitive information	None, Low, Mod, High
Personal Safety	None, None, Low, Mod/High
Civil or personal violations	None, Low, Mod, High

Answers to the questions

1. Password Aging

What is password aging?

It is maximum and minimum lifetime for user passwords, a method used to protect user account by means of recording number of user logins with correct password. By this policy user is prompted or forced to change password on regular basis, ie. after 10 successfully logins.

And what methods exist to implement it?

Aging is set globally in the files /etc/login.defs and /etc/default/useradd, but these settings are only applied when new user accounts are created, so to modify lifetime of password for some existing account, one may use change command.

Considering minium and maximum password ages, passwords should have some minimum age to prevent users from rapidly "cycling throught" password changes in attempts to resuse old passwords. Few days is optimal minimum password lifetime. As for maximum password age, reasonable seems to be two or few months to dont let user be frustrated due to many password changes that may lower password quality.

Its better to eduacate users about proetecting passwords that to rely on password aging.

2. If you want to increase the security of a system, you can use one-time passwords. What are the advantages? What are the disadvantages?

One-time password is valid only once. This kind of password system may be implemented using special password generators (time dependant passwords, dynamic password generation) or simply as a list of passwords. Special type of one-time passwords are those generated by a challenge-response system for which user calculates response (as password) using the challenge, so password will change every time and can not be reused; secret lays in function translating challenge to response.

Main advantage and distadvantage are:

- resistant to evasdropping and wire-taping;

- being non user friendly;

- anti brute force guessing;
- leaks of databases will not give valid password;

- requires some way of creating new password, like mobile phone, already generated list, smart card, email;
- attacker with fresh one-time password is free to go;
- replaces know with have;

Biggest disadvantage is being non-user friendly, also hard to implement and forces user to have external source of one time passwords. Advantage is protection against brute force login attempts. In this case salt may strengthen it even more, but at the same time, salt seems to be not necessary if hacker has only one time guess for each password.

3. Authentication systems are often based on some knowledge shared by the computing system and the user. This knowledge can be of three types: something the user knows, has or is. For each of these types, answer the following questions in subsections.

Role of authentication is to verify a users identity. Each user will have one identifier and one authenticator, where identifier tells who you are, and authenticator verifies that this is true that you are specific user. Authentication procedure has 4 steps of : identification, provision of authentication information, transmission of authentication information, validation of information vs reference. Problems or attacks may occur in all 4 steps. It may be assumed that validation is similar among tree types : knows, has, is. Weakest point is often at transmission channel, specially at long distances (short may be vulnerable too). Transmission channel usual threats are: eavesdropping; manipulation of routers or gateways; replay attacks. Usual remedies also apply for it.

3.1. How can the authentication mechanism be implemented?

User Knows:

- Generally passwords kept in user brain memory.
- Examples include a password, PIN, answers to set of questions.
- System may offer user accounts with passwords/pin-code that only user should know, then restrict access to this user files only for this user (and in linux to root).

User Has:

Generally:

- tokens, ie. electronic keycards, smar cards, physical keys, emails, ID badges, two step authentication with phone or email, NFC or RFID chips
- external tools that are hard to get access to beside user and hard to emulate
- For example to access by ssh it is more safe to use encrypted key than password, which user may store in his account

User can have virtual (email, app) or physical (card, keytag) tokens that may differ in means of advancement or complexity. Cards may be of type: - embossed (old credit card), - magnetic (bank card), - memory (prepaid phone card), smart contact/contactless (biometric ID card)

As tokens we call objects that user possesse for purpose of auth. Widely used are two types of tokens both with appearance of and size of bank cards: memory cards and smart cards. Memory cards can store but not process data, commonly they have magnetic stripe on the back that can strony only simple security code. This code can be read and reprogrammed by attacker that uses inexpensive card reader. Some memory cards additionaly include internal electronic memory. Besides, a wide variety of devices qualify as smart tokens. They can be categorized along four dimensions that are not mutually exclusive: physical characteristics, user interface, electronic interface (contact or contactless), authentication protocol (static, dynamic password generator, challenge response).

In terms of user auth, most important of smart tokens are smart cards that contains entire microprocessor, including processor, memory and I/O ports. Where memory is of 3 types: read-only memory (ROM) that stores card number and owner name; erresable ROM that hold app data and programs ie. protocols, and data that may vary with time; random access memory (RAM) holds temp data generated when apps are executed.

Increasing importance of smart cards lays in usage them as national ID cards for citizens.

User authentication may be shown on example of online use of the eID by web-based application:

- (user, website): visit website via web browser
- (website, app-server): request for auth service via website
- (app-server, eID-server): website forwards an auth request to an eID server by user
- (eID-server, user): eID server requests PIN for eID card
- (user): user gives correct PIN
- (eID-card, eID-reader): encrypted data exchange between eID card and terminal reader
- (eID-card, eID-server): server do auth protocol exchange with microprocessor on the eID card
- (eID-server): user is authenticated
- (eID-server, user): results are sent back to user system
- (user, app-server): results are redirected to the web server app
- (app-server, user): service is granted to user

For this scenario needed is software and hardware on user system. Software need functions like: request and accept PIN number; message redirection. Hardware requires external card reader, that can be external contact or contactless reader or contactless reader internal to user system.

User Is:

Generally:

- Unique characteristics divided on static and dynamic (is/does).
- Biometric authentication based on pattern recognition of DNA, iris, face, fingerprint scan or voice, hand geometry, signature, gesture;
- It can be implemented by scanners that at first record user characteristics and then use it authenticate at each login attempt by means of comparision.

Biometric security usually goes with steps like:

- enrollment to system:
  - system senses some biometric characteristics of user,
  - digitizes it,
  - extracts set of features
  - store them with set of numbers as user template;
- auth may be performed as either:
  - verification:
    - analogous to login by smart card with PIN/password,
    - user enters PIN and uses biometric sensor,

        - system extracts corresponding feature
        - system compares it to template stored for that user
    - identification:
        - use biometric sensor without other information,
        - system compares it to existing templates
        - some existing user can be identified

3.2. What advantages and disadvantages does the authentication method have with respect to e.g. user friendliness, cost of introduction, and accuracy?

All of these methods, properly implemented and used, can provide secure user authentication. However, each method has pros, cons and problems. Passwords comes with no cost and if easy to remember, is user friendly, but an adversary may be able to guess or steal a password. Similarly, an adversary may be able to forge or steal a token. So they may be seem as not as accurate security tool as they are. A user may forget a password or lose a token. Further, there is a significant administrative overhead for managing password and token information on systems and securing such information on systems. With respect to biometric authenticators, there are a variety of problems, including dealing with false positives and false negatives, user acceptance, cost, and convenience.

User auth of type like knows, has, is/do, specially remote ones, is subject to variety of attacks like:

- client attack:
    - user auth without acces to remote host or to interviewing communications path
    - attempt to masquerade as legitimate user
- host attack:
    - directed at user file at host where passwords, passcodes or templates are stored;
- eavesdropping, theft, copying:
    - human is simply spying user and try to know password by nasty means
    - malicious software is keyloggin user activity for later analysis
- reply:
    - involve repeating a previously captured user response
- trojan horse:
    - app or device masquerades as authentic app or device for purpose of capturing password, passcode, template;
    - with this adversary may masquerade as legitimate user
    - ie. rogue bank machine used to capture user ID/passwd combinations
- DOS:
    - attempts to disable a user auth service by flooding the service numerous auth attempts
    - selective attack denies service to specific user by causing lockdown

Among which, trojan horse and DOS attacks are very similar and base on respectively: installation of rogue client or capture device; lockout by multiple failed auths.

User Knows:

- Easy to implement, low cost;
- familiar to user and easy to get used to;

- less user friendly in case of one-time passwords;
- Can be steal or guessed by adversary;

- high accuracy if salt included;
- Password authentication is portable;

- Cant be lost, but may forgotten;

- totally accurate;

User Knows is generally threated by:

- offline dictionary attack (brute force)
- specific account attack
- popular password attack

- password guessing against single user
- workstation hijacking
- exploiting user mistakes
- exploiting multiple password use
- electronic monitoring

Examples of potential attacks on passwords are:

- client attacks: guessing, exhaustive search
- host attack: plaintext theft, dictionary/exhaustive search
- eavesdropping, theft, copying: shoulder surfing
- replay: replay stolen password response
- trojan horse: installation of rougue client or capture device
- DOS: lockout by multiple failed attempts

User Has:

Generally:

- require special reader: additional hardware to match tokens, rise cost;
- user dissatisfaction: ie. for memory cards fine with ATM but for computer access seems inconvienant
- its not portable in terms of needed readers;

- requires direct access;

- direct access to token-reader may allow attacker to break it by special tools;

- more user friendly, when it only requires some connection of token to reader;

- if system is simple, ie. rfid token may be forgeable or system may be breached with brute force;
- can get lost or stolen;
- token loss: temp prevent its owner from system access

- can be shared between users;

- can be borrowed by attacked without notice of user;

- accurate as password (its actually kind of movable password);
- may have multiple forms, virtual form like email or authentication application, or be physical tokens
- may have multiple functions, ie. smart card may carry online/offline auth functions of ePass, eID, eSign

Memory cards can be used alone for physical access, ie. to holet room, but for auth additional pin is required, ie. for ATM. Memory card combined with passwd/PIN rises security over password alone in meaning that adversary need to have or clone card and know password. Drawbacks of token cards like memory or smart are:

- requires special reader: rises cost and requires maintanence of security over readers hard/soft-ware.
- token loss: disable access for legitimate user, requires cost of replacement, can be found, stolen, forged by adversary (then only determination of PIN is required to gain access)
- user dissatisfaction: its OK with ATM but to access computer it may be inconveniant

Examples of potential attacks on tokens are:

- client attack: exhaustive search
- host attack: passcode theft
- eavesdropping, theft, copying: theft, counterfeit hardware
- replay: replay stolen passcode response
- trojan horse: installation of rougue client or capture device
- DOS: lockout by multiple failed auths

User Is:

Pros/Cons:

- require additional hardware that accuracy will rise with price;
- can be breached with brute force in some cases, ie. multiple pictures of iris;

- faster and more user friendly than password and even pin, like phone finger print scan;
- generally more safe that token and password, require spy techniquest, laboratory access;

- with funds to hire a spy it may be easier to steal than password ie. user may watch out his password, but not where he left fingerprints, if someone film user, or user may be recorded.

- requires direct access to user, so it may be hard to achieve

- in case of very high standard of user tools, user life may be in danger;

- mass produced may rise general effectiveness

- mass produced may make it more crackable
- may give acces to imposer with some probability and block genuine user
- compared to password and tokens more complex and expensive;
- while used in number of specific apps, biometrics has yet to mature as a standard tool for user auth to computer systems;

- cant be stolen like pin/password by look or lost like token;

- not yet mature for special cases

- can give extra security when included in multi auth
- may be used for assurance level 1/2

Properly implemented system of biometric auth on example of iris scan used in UAE borders checkpoints will pass requirements like:

- identity single person from a large population of people
- rely on a biometric feature that does not change over time
- use biometric features that can be acquired quickly
- be easy to use
- respond in real-time for mass transit apps
- be safe and non-invasive
- scale into the billions of comparisons and maintain top performance
- be affordable

Here concept of accuracy does not apply to user auth schemes like with passwd and tokens. When user enters passwd it matches exactly or not with integer precision. In biometric pattern recognition we have floating precision of probability between 0 and 1, so system must determine how closely a presented biometric characteristics matches that what are stored. From most to least costly and grouped by accuracy we may consider biometric authentication of: iris; retina, finger; hand, signature, face, voice. That shows that easies to forge is last group, and hardest to forge is iris patterns.

Biometric accuracy:

- in any biometric scheme, some physical characteristics of individual are mapped into digital representation
- each individual single template is stored in computer
- during auth system compares stored template to presented one
- given complexities of physicial characterisitcs its hard to expect exact match between two templates
- system uses algorithm to generate matching score that has to be bigger than threshold for positive auth
- error of assesment during auth may have two forms in terms of probability: false match, false nonmatch;
- if single user is tested numerous times, the matching score *s* will vary with bell-curved PDF (probability density function)
- ie. for fingerprint result may vary due to sensor noise, print changes due to swelling or dryness, finger placement, etc...
- on average any non-valid humand should have a much lower matching score but even so will exhibit a bell shaped PDF
- dificulty lays in overlap of matchin scores of imposer and genuine user
- by moving treshold the probabilities can be altered, but: decrease in false match rate results in increase in false non match rate and vice versa

Biometric scheme:

- for given biometric scheme, false match and false nonmatch sets give operating characteristic curve

- schifting threshold point along the curve:
    - up to bigger false non-match rate will increase security, decrease convenience, ie. for high security apps
    - down to bigger false match rate, will decrease security, increase convienience, ie. for forensic app for many possible matches
    - to equal error rate line will result in tradeoff between security and convienience

Examples of typical attacks on biometric security:

- client attack: false match
- host attack:
    - template theft
    - biometric features are hard to secure, bcos they are physicial features of the user
- eavesdropping, theft, copying: copying (spoofing) biometric
- replay: replay stolen biometric template response
- trojan horse: installation of rogue client or capture device
- DOS: lockout by multiple failed auths

3.3. How can the disadvantages be overcome?

User Knows:

- use of password manager, ie. in browsers there is last-pass;
- usage of long passwords with fake words that are easy to remember for user;
- strong password may be better than password ageing;
- usage of one-time passwords;
- use within multiauth with has or/and is

User Knows - countermeasures to types of attacks:

- offline dictionary attack:
    - include controls to prevent unauthorized access to password file
    - intrusion detection measures to identify a compromise
    - rapid reissuance of passwords should one be compromissed
- specific account attack:
    - account lockout mechanism against brute force guessing after number of failed logins, typical limit is 5 attemps
- popular password attack over multiple users:
    - include policies to inhibit selection by users of common password and scanning the IP addresses of auth requests and client cookies for submission patterns
- password guessing against against single user:
    - training and enforcment of password policies of secrecy, minimum length and character set of passwords, prohibition to use well-known ID, password aging
- workstation hijacking:
    - auto logout after period of inactivity
    - detection schemes used on user behaviour
- exploiting user mistakes:
    - user training, intrustion detection, simpler passwords within multiauth
- exploiting multiple password use:
    - policy that forbids same or similar password on particular network devices
- electronic monitoring:
    - hard to deal with, simple encryption will not fix problem, so password can be observed and reused by adversary

Typical defenses agains potential attack examples (in form -> type:example:defense):

- client attack: guessing, exhaustive search: large entropy (many bits required to represent password, limited attempts in given time period from given source
- host attack: plaintext theft, dictionary/exhaustive search: hashing, large entropy, protection of password database
- eavesdropping, theft, copying: shoulder surfing: diligence to keep secret, admin diligence to quickly revoke compromised passwds, multifactor auth
- replay: replay stolen passwd response: challenge-response protocol
- trojan horse: installation of rogue client or capture device: auth of client or capture device within trusted security perimeter

- DOS: lockout by multiple failed auths: multifactor with token

Despite security vulnerabilities of passwords they are still most commonly used for auth with reasons like:

- problem with software for utilization of client-size auth hardware
- physicial tokens are expensive and inconvenient to carry (if multiple of them needed)
- schemes that rely on single sign-on to multi services using non-passwd services create single point of security risk
- automated password managers that relieve users have poor support for roaming and sync in cross platform.

User Has:

Solutions for general problems:

- Make tokens user friendly, that can be keep with keys, in wallet, phone app, etc
- autolock token readers in case of brute force attacks.

Typical defenses against potential attacks examples (in form -> type:example:defense):

- client attack: exhaustive search: large entropy, limited attempts, theft of object requires presence;
- host attack: passcode theft: same as passwd, 1-tim passcode
- eavesdropping, theft, copying: theft, counterfeiting hardware: multifactor auth, tamper resistant/evident token
- replay: replay stolen passcode response: challenge-response protocol, 1-time passcode
- trojan horse: installation of rogue client or capture device: auth of client or capture device within trusted security perimeter
- DOS: lockout by multiple attempts: multifactor with token

User Is:

Solutions for general problems:

- Mass produced may rise general security
- Protection and life monitoring of important users.
- Connect with authentication by knows and/or has

Typical defenses against potential attacks examples (in form -> type:example:defense):

- client attack: false match: large entropy, limited attempts
- host attack: template theft: capture device auth, challenge response
- eavesdropping, theft, copying: cpying (spoofing) biometric: copy detection at capture device and capture device auth
- replay: replay stolen biometric response: copy detection at capture device and capture device auth via challenge response protocol
- trojan horse: installation of rogue client or capture device: auth of client or capture device within trusted security perimeter
- DOS: lockout by multiple failed auths: multifactor with token

4. What authentication method would you recommend for use in computer systems in the following environments? (Motivate your answer, and state the assumptions you have made concerning the security needs in each environment.)

4.1. A university

Univeristy in means of security could be divided accoringd to purpose on:

- students
- administrative staff
- researchers

Also it may be divided on importance level of information:

- public: news, general informations, schedules, lists of students/staff in department, teachers contact data, study materials (ie. lectures)
- private: student accounts (ie. student portal, student accounts on computers for work in class), student grades, lists of courses student have, private personal informations of students and staff
- organizational: high costly data of research performed by university for external customers like companies or private organizations, technological data

- national: high security and costly data of research connected to national security, similar to above but for ie. national organizations, military, government

Even if data importance is important in uni, it will not directly affect human life. Attacks with succesfull penetration may cause damage to uni liability, financial loses, technological spying, students mental life. But indirect effects may include effects on third-parties related to university in some kind of cooperation. If ie. uni is working on OS for autonomous driving car, stealing source of this OS may help attackers to crack system, or even plant some backdoor without notice. Though, attackers need first to get knowledge who is researching what in uni (maybe by public news or researchers portfolio), then spy and outfox this person.

Now lets see what may happend as result of hacker attacks for each of this sectors of importance:

- public (from best to worst cases):
  - removal or falsification of schedule: ie. for system like timeedit will create chaos in students schedule
  - study materials removal: will make study harder
  - lists of students/staff change/removal: important during exam sessions
  - simple prank in news or general information: will be like slap to university face
  - prank on university webpage: similar as above
  - general information masquerade: may lead to contacting units faking to be university staff, low to medium loses of contacting person, lose of uni liability
  - help information masquerade: may lead to helping with malicious software or websites, ie. change password on keyloggin website
  - teachers contact data masquerade: may lead to phone, email exchange with attacker that will forward it and steal data on the way
- private:
  - by private computers of staff/student researcher my get access to his uni account
  - students grades may be altered
  - students/staff identity may be removed, falsified
  - staff/scholarships bank accounts may be altered
- organizational/national:
  - theft, damage, corruption (or if software related maliciousanization) of research data of third-party product
  - eavesdropping of communication between researchers and third-party clients
  - indirect infiltration of third party products using uni as bridge

It means that university computer systems has to be classified depending on terms of: levels of assurance, potential impact, mapping between two by means of areas of risk. Then this kind of systems has to be isolated/restricted to not affect one from another. According to this authentication access to students, staff, researchers computer systems may be implemented like:

- students:
  - password based authentication with assurance level 2
  - student account is created by uni staff
  - this accounts have default permissions
  - with access restricted to only their account
  - without permissions to execute, write or read other users or system files (including directories)
  - password solution is costly affordable for big amount of students,
  - additionally for students convenience to let them access study placess at will
  - to secure this places from thiefs they may be equipped with smart cards readers for students eID,
  - then others and thiefs will not have possibility to masquerade as students
  - additionaly there should be put physical restrictions from stealing hardware, ie. metal boxes for computers and chains for screens
  - password cracking of students, including local and remote access, may be protected by account lockdown for some time, delays between each login attemps, large entropy of passwords, and protection of passwords database
  - this solution will outweight potential gains from attacking students accounts
  - passwords may be still eavesdropped by intruders
  - by user account intruder may get "offline" basic access to system and do further attempts of penetration
  - targeted may be account of administrator to gain access to system as whole
  - backup servers are not required for computer disks due to low value of students work, but its good idea to keep backup of data in student portal
  - keeping same uid and passwd for multiple students services comes with convenience for students probably outweighting

    caused risks,
    - thought it rises potential risks of swooping over all this services in case of intruders sucessfull penetration, ie. onedrive, email, lib account, study account, registered courses
- researchers of low/medium values projects, staff/administration of studies:
    - secure rooms with only required staff access by ie. eID
    - password and token (ie. smart card) based authentication with assurance level 3,
    - required to avoid hijacking of logged in staff workstations (where students may alter grades)
    - multiauth is good solution for multiple types of attacks: trojans, DOS, password cracking
    - hasing, large entropy will make password more secure
    - backup servers are required and watching over logs of sudoers accounts
- researchers of high values projects, roots/admins of uni computer networks:
    - multiauth based authentication with assurance level 3
    - each faculty and department should have isolated networks and computers
    - one-time passwords or passwords with large entropy
    - tokens like eID or phone smsm codes
    - biometric authentication like iris scanner
    - backup servers and logging

4.2. A military facility

Military facility may mean:

- military border stations
- military research centers (ie. chemical, biological weapons)
- military bases with troops
- military bases with planes
- military bases with rockets, bombs

But generally we classify military facility as important to national security, so we tend to give it assurance of level 3/4 due to high level of potential risks that may lead to catastophic events, death of humans life, etc... Thus military facility should have strict user authentication beginning from policy to used tools. Access to computer systems, including hardware and rooms should be only granted to genuine users with multifactor authentication. Facility should be accessed by eID, computer/server rooms with eID and iris scanner, both watched by human guardians to avoid outfoxing reader devices. Then computers should be accessed with eID, and passwords or one-time passwords given by superior. Only responsible for this computer systems should be capable of reaching computers physicially and attempt to login. Cleaning service should be also performed by personel responsible for computers to avoid internal intruders. Problem may rise with plugin devices, ie. usb pendrive for update purposes that may contain malicious software. Gaining highest authority should be only accesed by offline means to isolated computers, split on two/three people with long passwords and token authentication.

4.3. A middle-sized company

We define mid-sized company as that having 50-250 workers and annual revenue of less than 50 [M Euro]. Depending of what type of market is this company area, ie. shopping, transportation, automotive, chemistry, biochemistry, services, medicinal, metalurgy, software, hardware, hospitals, etc... we may set required assurance level from 2 to 4. Some companies may influence human life more than others and some may have assets more sensitive on cybercrime than others. But generally we may say that mid-sized company will have assets more or less sensitive for its proper functioning, thats why standard token multiauth authentication with token and password is proper choice.

4.4. A personal home computer

Personal home computer needs assurance of level 1 or 2 of authentication. User will create account for himself and manage it how he see it. If this computer is related to work, ie. perform remote connection then it require similar authentication as that in work. If this computer is only needed for browsing network, watching movies, and typical fun, then even password is not required. But if user logins to bank, email from this computer them password/pin is required with high entropy. Also home wifi routers require strong password. Naive humans or kids may also need to install antivirus software.

5. In some systems you do not only implement authentication of the user against the system, but also of the system against the user. What is the purpose of doing this?

This implementation of two sides authentication is to make shure that both sides are genuine. For example hashed key authentication within SSH, website with SSL authentication.

It is mutual authentication used to avoid problems like man in the middle attacks.

Lets consider what happens with a Web server with SSL authentication:

- browser first connects to the server,
- server sends its certificate.
- certificate and the encryption informs that client connected real server: browser draws that information to the user by displaying green padlock icon, and dont show popub about new certificate
- user types his login and password, which are sent to the server (within the SSL tunnel).

In SSL the server certificate authenticates the server, so the user knows he's talking to the genuine server.

Authentication of the system against the user is done with: server certificate. Authentication of the user against the system is done with: password.

Both ways of authentication are important, bcos in case if there was no server certificate:

- the user believe that it contacted the genuine "www.bank.com" server,
- in reality, user data in under unauthorized disclosure threat to confidentality by interception,
- ie. user data packets are hijacked by an attacker that operates a free hotspot:
- so user is talking to the attacker's server that may masquerade as genuine "www.bank.com",
- attacker server be the man in the middle and simply forward intercepted packets to genuine website, then send back response.

6. A user is running a program containing the system call setuid(). Depending on who is running the program and on the value of the argument to setuid(), different things can happen.

If executable has speciall permission of setuid (>= 4xxx) then effective UID of process will be that of owner of its file, but real UID will be of user that executed it.

So if file is:

- run by x and owned by root:root it will have: euid = 0, ruid = x. Can change uid to anybody.
- run by x and owned by x:x it will have: euid = x, ruid = x. Can change uid to x only.
- run by x and owned by y:y it will have: euid = y, ruid = x. Can change uid to x and y only.

File with permissions 4755 and owned by root:root may be run by anybody (users, world) and can setuid to anybody, so it is dangerous.

If executable is run by x, owned by y, and have permissions to change uid, then calling setuid(y) will change ruid and euid to y, but calling seteuid(y) will change only euid to y.

6.1. Study Figure 1. What are the values of the real user ID (ruid) and the effective user ID (euid) at i) and ii)?

csec28 is running program /bin/prog with file access rights: -rwxr-xr-x /bin/prog root root

Setuid special permission is not set, and executable is not run by root or sudo, so it is run by general user and then process will be created with efective and real UID of user csec28.

- i) Before setuid(csec028): euid = csec028, ruid = csec028
- setuid(csec028)
- ii) After setuid(csec028): euid = csec028, ruid = csec028

6.2. Table 1 illustrates six cases where root or a normal user account starts a normal program which uses setuid() with different arguments (user IDs). Fill in the third and fourth column in the table. The third column should state whether the system call will succeed or fail. The fourth column should state the user ID (ruid) of the program after the setuid() call.

	UID before setuid()	setuid(UID)	success/failure	UID after setuid()
1	root	0 (root)	s	0
2	root	20757 (csec069)	s	20757

3	root	20716 (csec028)	s	20716
4	csec028	0 (root)	f	20716
5	csec028	20757 (csec069)	f	20716
6	csec028	20716 (csec028)	s	20716

7. On Unix systems, file-access permissions on programs may be set to set-user-ID (these are often called SUID programs) as in the passwd program:

csec@legolas~ > ls -l /usr/bin/passwd -rwsr-xr-x 1 root root 27888 Jul 26 09:22 /usr/bin/passwd

Study Figure 2 while answering the following questions:

7.a. What are the values of the real UID (ruid) and the effective UID (euid) at i), when the user csec028 runs the SUID program /bin/prog?

csec028 is running the SUID program /bin/prog with the file access rights: rwsr−xr−x /bin/prog root root

Then:

RUID = csec028 EUID = root (0)

7.b. What is the purpose of using SUID on programs?

This program is executed with RUID of user who executed it csec028, but EUID is that of owner of file root:root. Due to special permission bit setuid 4xxx is set, then no matta who executed this program, its process will have permissions that of root. This kind of programs if owned by root are executed to let standard users run with delegated root privilidges, but for this better is usage of sudo. Note that linux kernel ignores setuid bits on files other than executable.

This kinds of programs may be usefull if one user wish to let other user run owned by him program with his permissions, then this other user do not need to be in sudoers group.

8. Sometimes the setuid() (or perhaps more likely, seteuid(2)) function is used in programs where the set-user-ID bit is activated on the program's file-access permissions. This is +done, for example, in the /usr/bin/passwd program.

8.a. What are the values of ruid and euid at i) and ii) in Figure 3?

csec028 is running the SUID program /bin/prog with the file access rights: −rwsr−xr−x /bin/prog root root

- i) before setuid(csec028): euid = 0, ruid = csec028
- setuid(csec028)
- ii) after setuid(csec028): euid = csec028, ruid = csec028

8.b. What is the purpose of doing this?

After executing necessary calls in SUID program that require root privilidges, we lower this rights to that of standard user csec028 by call setuid(csec028). But problem lays with SUID, so its possible to revert back to root privilidges by call setuid(0). To properly drop root uid, we may follow guidelines from "POS36-C" and "POS37-C". That is, dropping root uid should follow schema like in code of Appendix B.

```c
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>

#define NEED_EUID 1
#define EUID geteuid()
#define RUID getuid()

int root2user(){
    /* Store the privileged ID for later verification */
    uid_t myRUID = RUID;
    uid_t myEUID = EUID;

    /* Code intended to run with elevated privileges   */

    printf("1(ruid=%4d, euid=%4d) ::: ", myRUID, myEUID);

    /* Temporarily drop privileges */
    if (seteuid(myRUID) != 0) {
        printf("error: temporary drop failed: %s\n", strerror(errno));
    }

    printf("2(ruid=%4d, euid=%4d) ::: ", RUID, EUID);

    /* Code intended to run with lower privileges  */

    if (NEED_EUID) {
        /* Restore Privileges */
        if (seteuid(0) != 0) {
            printf("error: restore privileges failed: %s\n", strerror(errno));
        }

        /* Code intended to run with elevated privileges   */
        printf("3(ruid=%4d, euid=%4d) ::: ", RUID, EUID);

    }
    /* ... */
    printf("\n4(ruid=%4d, euid=%4d) ::: ", RUID, EUID);

    /* Restore privileges if needed */
    if (EUID != 0) {
        if (seteuid(0) != 0) {
            printf("error: restore privileges failed: %s\n", strerror(errno));
        }
        printf("5(ruid=%4d, euid=%4d) ::: ", RUID, EUID);
    }

    /* Permanently drop privileges */
    if (setuid(myRUID) != 0) {
        printf("error: permanent drop failed: %s\n", strerror(errno));
    }
    printf("7(ruid=%4d, euid=%4d) ::: ", RUID, EUID);

    if (setuid(0) != -1) {
        printf("error: privileges can be restored: %s\n", strerror(errno));
    }

    printf("8(ruid=%4d, euid=%4d)\n\n", RUID, EUID);

    /*
     * Code intended to run with lower privileges;
     * attacker cannot regain elevated privileges
     */

    return 0;
}


int main(int argc, char **argv)
{

    printf("DROP EUID TEST \t");

    root2user();
    return 0;
}
```

Conclusions

Authentication problem is basic tool of CIA triad that:

- allows for genuine user to access his account data in means of avaliability,
- keep this account private from intruders with means of confidentality
- more or less made it work properly without attackers inteference in account functions by means of integrity

Used means to secure this login program made it have:

- more confidentality and integrity, by unauth login prevention means
- less avaliability by cost of previous two in case of attacks like password guessing

User is gaining confidence that his account is more confident and integrated, but this comes with cost of possible drop of availability in case of attacker attempts of attack on this account.

This implementation include:

- password ageing after sucessfull logins, then forcing user to change password after limited age to new encrypted password by contacting admin
- signal handler that is blocking: Ctrl+z, Ctrl+Del, Ctrl+Backspace
- use of buffer overflow protected functions and protection means
- account lockdown after too many failed logins
- infinite executing service that cycle in while loop until sucessfull login to user shell
- no information spreading if given login exsist or not
- delay in form of sleep for 1[s] before stdin of password, after failed login, and in case of account lockdown there is 5[s] delay: this means that each login attempt has delay of 2[s]
- password encrypted with added salt

This kind of login program is secured from password guessing, brute force, DOS, buffer overflow attacks, by means of delays that make slower for password crackers and finally by account lockdown. Also this implementation do not give any hints about existing accounts to attacker. Password aging in this implementation forces user to get new one-time password from administrator, and that method should secure it from theft of old passwords.

We may see now, that if login program is properly designed and coded, then it stands at bulletproof tool against unauthorized logins to users account, but its still vulnerable on DOS attacks. Thats why next step would be introducrion of tools to rise avaialibity of this account to make it safe from DOS attacks, or to make multi auth, that is passwd with token like eID.

Pity we do not go deep in buffer overflow attacks and only get to know how to protect from them, by using standard functions and coded handlers. This implementation let to turnoff mechanisms of bufferoverflow protection and in case of its happening, labs system will simply echo overflowed chars. Real bufferoverflows attacks would inject shellcode in stack memory of program and may result in executing shell without need to login by intruder, so phenomena of bufferoverflows are dangerous for users security.

Trthfully, we would rather spend time on bufferoverflows coding than writing this report 😦 but this knowledge is probably usefull too, ie. for exam.

Appendix A - Code: Linux Login

```
/* $Header:
https://svn.ita.chalmers.se/repos/security/edu/course/computer_security/trunk/lab/login_linux/login_l
inux.c 585 2013-01-19 10:31:04Z pk@CHALMERS.SE $ */

/* gcc -Wall -g -o mylogin login.linux.c -lcrypt */
```

```c
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <stdio_ext.h>
#include <string.h>
#include <signal.h>
#include <pwd.h>
#include <sys/types.h>
#include <crypt.h>
#include <linux/capability.h>
#include <errno.h>

/* Uncomment next line in step 2 */
#include "pwent.h"

#define TRUE 1
#define FALSE 0
#define LENGTH 16
#define MAX_PASSWD_AGE 10
#define MAX_LOGIN_ATTEMPTS 5
/*
 in step 2:
 create file passdb and add records of users, salt need two numbers

 in step 4:
 ./makepass 'sa'

 */


//flags for toggling protections, hange 0 to 1 to enable
#define DOSIGNAL_CATCHING 1
#define BUFFER_OVERFLOW_PROTECTION 1

//get line flags
#define OK       0
#define NO_INPUT 1
#define TOO_LONG 2
#define FAIL     3

static int get_line(char *prmpt, char *buff, int sz) {

        if(BUFFER_OVERFLOW_PROTECTION) {

                int ch, extra;

                // Get line with buffer overrun protection.
                if (prmpt != NULL) {
                        printf("%s", prmpt);
                        fflush(NULL); /* Flush all  output buffers */
                        __fpurge(stdin); /* Purge any data in stdin buffer */
                }


                /* step 3, gets() is vulnerable to buffer,
                 * change to fgets(char * __restrict__ _Buf,int _MaxCount,FILE * __restrict__ _File)
*/
                if (fgets(buff, sz, stdin) == NULL)
                        return NO_INPUT;

                // If it was too long, there'll be no newline. In that case, we flush
                // to end of line so that excess doesn't affect the next call.
                if (buff[strlen(buff) - 1] != '\n') {
                        extra = 0;
                        while (((ch = getchar()) != '\n') && (ch != EOF))
                                extra = 1;
                        return (extra == 1) ? TOO_LONG : OK;
                }

                // Otherwise remove newline and give string back to caller.
                //step 3, fgets is used instead of gets you will get a "\n"
                // at the end of the string that need to be replaced by "\0".
                buff[strlen(buff) - 1] = '\0';

        } else {
                if (prmpt != NULL) {
                        printf("%s", prmpt);
                }

                gets(buff);

                printf("...buffer overflow protection turned off for stdin getter!\n");
        }

        return OK;
}

void passwd_age_handler(mypwent *passwddata) {
        char doNewPass[2]; //has to be size=2, bocs y/n + \n is 2 chars

        passwddata->pwage++;

        if (passwddata->pwage > MAX_PASSWD_AGE) {

        //BETTER WOULD BE TO NOT CHANGE PASSWORD DURING LOGIN, BUT ASK FOR ONETIME PASSWORD FROM
ADMIN
        printf("...Ask for new one-time password from admin!\n");
        exit(0);

        //DURING LABS IT WAS LIKE THIS THAT IF PASSWORD IS OLD, USER WILL CHANGE IT HIMSELF:
                //int passwdChangeStatus = get_line("...Password to old! Change password (y/n)? ",
doNewPass, 2);

                // if (passwdChangeStatus != 0) {
                //      printf("...Change password wrong option. Terminating...\n");
                //      exit(0);
                // }

                // printf(doNewPass);
                // printf("\n");

                // if (doNewPass[0] == 'y') {
                //      printf("...Password changed (fakely, not implemented), age reset ...\n");
                //      passwddata->pwage = 0;
                // } else {
                //      printf("....Password not changed. Terminating...\n");
                //      exit(0);
                // }
        }
}

void uid_setter(mypwent *passwddata) {
        /*  check UID, see setuid(2) */

        printf("...set uid\n");

        int setuidStatus = setuid((__uid_t) passwddata->uid);

        if (setuidStatus != 0) {
```

```c
                printf("...Setuid failure, error : %s! Terminating...\n", strerror(errno));
                exit(errno);
        } else {
                printf("...setuid(%d) status : %d\n", passwddata->uid, setuidStatus);
                printf("...real user id -> getuid(%d)\n", getuid());
                printf("...effective user id of process -> geteuid(%d)\n", geteuid());
        }
}

void shell_runner(mypwent *passwddata) {
        /*  start a shell, use execve(2) */

        printf("...run /bin/sh\n");
        //Init of array used for execve()

        char *execveEnvp[16];
        char envc[16][64];

        //here even if we set user name: hpc, we will still gen shell run by user that executed this
program
        sprintf(execveEnvp[0] = envc[0], "TERM=xterm");
        sprintf(execveEnvp[1] = envc[1], "USER=%s", passwddata->pwname);
        sprintf(execveEnvp[2] = envc[2], "HOME=/");
        sprintf(execveEnvp[3] = envc[3], "SHELL=/bin/sh");
        sprintf(execveEnvp[4] = envc[4], "LOGNAME=%s", passwddata->pwname);
        sprintf(execveEnvp[5] = envc[5], "PATH=/usr/bin:/bin:/opt/bin");

        int execveStatus = execve("/bin/sh", NULL, execveEnvp);

        if (execveStatus != 0) {
                printf("...Execve(bin/sh) error : %s! Terminating... %d\n", strerror(errno),
execveStatus);
                exit(errno);
        }
}

void bad_pass_handler(mypwent *passwddata) {
        printf("...login incorrect \n");
        passwddata->pwfailed++;//step 5, failed logins
        sleep((unsigned int) passwddata->pwfailed); //sleep for n[s] where n is # bad passwords
        if (passwddata->pwfailed > MAX_LOGIN_ATTEMPTS) passwddata->pwage = MAX_PASSWD_AGE + 1;
}

int good_pass_handler(mypwent *passwddata) {

        int loginStatus = OK;

        if (passwddata->pwfailed > MAX_LOGIN_ATTEMPTS) {
                printf("To many login attempts. Account locked!\n");
                sleep(5);
                loginStatus = FAIL;
        } else {
                printf(" You're in !\n");
                passwddata->pwfailed = 0;//step 5, failed logins reset
                passwd_age_handler(passwddata);//even if we login successfully, raise the age
        }
        return loginStatus;
}

void signal_catcher(int sign, void(*function)(int)) {
        signal(sign, function);
}

void signal_handler() {
        /* add signalhandling routines here */
        /* see 'man 2 signal' */
        if(DOSIGNAL_CATCHING) {
                signal_catcher(SIGINT, (void (*)(int)) &signal_handler);  //int
                signal_catcher(SIGTSTP, (void (*)(int)) &signal_handler); //suspend
                signal_catcher(SIGQUIT, (void (*)(int)) &signal_handler); //quit
                signal_catcher(SIGABRT, (void (*)(int)) &signal_handler); //abort
        } else {
                printf("...signal catching is turned off!\n");
        }
}

int main(int argc, char *argv[]) {

        //struct passwd *passwddata; /* this has to be redefined in step 2 */
        /* see pwent.h */
        int nEmptyLogins = 0;

        char *userPass;
        char *encrPass;

        char prompt[] = "password: ";

        char user[LENGTH];
        char important[LENGTH] = "***IMPORTANT***";

        mypwent *passwddata;

        signal_handler();

        while (TRUE) {
                /* check what important variable contains - do not remove, part of buffer overflow
test */
                printf("...value of variable 'important' before input of login name: %s\n",
                       important);

                int loginStatus = get_line("login: ", user, LENGTH);

                if (loginStatus == TOO_LONG) {
                        printf("...buffer overflow attack detected, terminating!\n");
                        //exit(0); should not exit, fgets
                }

                /* check to see if important variable is intact after input of login name - do not
remove */
                printf("...value of variable 'important' after input of login name: %*.*s\n",
                       LENGTH - 1, LENGTH - 1, important);

                userPass = getpass(prompt);
                passwddata = mygetpwnam(user); //step 2

                sleep(1);//against brute force attack

                if (passwddata != NULL) {
                        /* You have to encrypt user_pass for this to work */
                        /* Don't forget to include the salt */

                        //step 2, change to ->passwd
                        //step 4, has to hash given password with salt
                        encrPass = crypt(userPass, passwddata->passwdSalt);

                        loginStatus = FAIL;
                        if (!strcmp(encrPass, passwddata->passwd)) {
                                loginStatus = good_pass_handler(passwddata);
                        } else bad_pass_handler(passwddata);
                        //step 5, failed logins
                        mysetpwent(passwddata->pwname, passwddata);

                        if (loginStatus == OK) {
                                uid_setter(passwddata);
                                shell_runner(passwddata);
                        }

                } else {

                        //NOTHING SHOULD BE HERE, DONT INFORM IF LOGIN BAD OR WRONG PASSWORD
                }


        }

        return 0;
}

//cd ~/projects/EDA263_CODE/Lab1 && make && ./login_linux
```

Appendix B - Code: Drop root privileges

```
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>

#define NEED_EUID 1
#define EUID geteuid()
#define RUID getuid()

int root2user(){
    /* Store the privileged ID for later verification */

    uid_t myRUID = RUID;
    uid_t myEUID = EUID;

    /* Code intended to run with elevated privileges   */

    printf("1(ruid=%4d, euid=%4d) ::: ", myRUID, myEUID);

    /* Temporarily drop privileges */
    if (seteuid(myRUID) != 0) {
        printf("error: temporary drop failed: %s\n", strerror(errno));
    }

    printf("2(ruid=%4d, euid=%4d) ::: ", RUID, EUID);

    /* Code intended to run with lower privileges  */

    if (NEED_EUID) {
        /* Restore Privileges */
        if (seteuid(0) != 0) {
            printf("error: restore privileges failed: %s\n", strerror(errno));
        }

        /* Code intended to run with elevated privileges   */
        printf("3(ruid=%4d, euid=%4d) ::: ", RUID, EUID);
```

```c
    }

    /* ... */
    printf("\n4(ruid=%4d, euid=%4d) ::: ", RUID, EUID);

    /* Restore privileges if needed */
    if (EUID != 0) {
        if (seteuid(0) != 0) {
            printf("error: restore privileges failed: %s\n", strerror(errno));
        }
        printf("5(ruid=%4d, euid=%4d) ::: ", RUID, EUID);
    }

    /* Permanently drop privileges */
    if (setuid(myRUID) != 0) {
        printf("error: permanent drop failed: %s\n", strerror(errno));
    }
    printf("7(ruid=%4d, euid=%4d) ::: ", RUID, EUID);

    if (setuid(0) != -1) {
        printf("error: privileges can be restored: %s\n", strerror(errno));
    }

    printf("8(ruid=%4d, euid=%4d)\n\n", RUID, EUID);

    /*
     * Code intended to run with lower privileges;
     * attacker cannot regain elevated privileges
     */

    return 0;
}


int main(int argc, char **argv)
{

    printf("DROP EUID TEST \t");

    root2user();

    return 0;
}
```