

# Content

1. Termin 1 .....	2
1. Zakladne vlastnosti bezpecnosti .....	2
2. Elektronicky podpis .....	3
1.2.1. Digitalny podpis.....	3
1.2.2. Digitalna obalka.....	4
1.2.3. SSL .....	5
3. Email – Elektronicka posta .....	6
2. Termin 2 .....	8
1. DNS and DNSSEC .....	8
2.1.1. DNS.....	8
2.1.2. DNS SEC .....	8
2. XSS.....	8
3. Symetricke a Asymetricke Sifrovanie .....	8
2.3.1. Symetricke Sifrovanie.....	8
2.3.2. Hashovacie funkcie.....	8
4. Pretecenie Zasobnika – Buffer Overflow.....	9
5. Security Modules.....	9
2.5.1. DAC vs MAC.....	9
2.5.2. Bella-La Padula .....	10
2.5.3. Biba.....	11
3. Bonus.....	13
1. URL shortener (goo.gl, tinyurl...) .....	13
2. Salting.....	13
3. Phishing .....	15

# 1. Termin 1

## 1. Zakladne vlastnosti bezpecnosti

- **Utajenosť (Confidentiality)**

**Data confidentiality** : Assures that private or confidential information is not made available or disclosed to unauthorized individuals. **Privacy** : Assures that individuals control or influence what Information related to them may be collected and stored and by whom and to whom that information may be disclosed. **Mechanizmus na zabránenie prístupu k informáciám neautorizovaným osobám, ktoré nemajú na to dostatočne pravá.**

Nastroje pre zabezpečenia utajenia:

- Sifrovanie
- Kontrola Prístupu
- Autentifikácia
- Autorizácia
- Fyzická kontrola prístupu

- **Integrita (Integrity)**

**Data integrity** : Assures that information and programs are changed only in a specified and authorized manner. **System integrity** : Assures that a system performs its intended function in an unimpaired manner, free from deliberate or inadvertent unauthorized manipulation of the system. **Mechanizmus na to, aby v komunikácií nedošlo k nejakej modifikácii informácií, či už nejakými útočnými alebo nejakým iným spôsobom**

Nastroje pre zabezpečenia integrity

- Zalhovanie
- Kontrolne Sumy

- **Dostupnosť (Availability)**

Assures that systems work promptly and service is not denied to authorized users.

Nastroje:

- Redundancie
- Fyzická ochrana
- **Istota, Dovernosť (Accountability ?!)**

Manažment dôveryhodnosti (dôvernosc medzi systémom a používateľom) je na takom stupni, že systém a používateľ si dôverujú navzájom.

- **Autentickosť (Authenticity)**

Stanovenie, že údaje, postupy a práva vydané osobami alebo systémom sú pravé

Under Integrity. Je to stanovenie, že údaje, postupy a práva **vydané osobami alebo systémom sú pravé**. Hlavným nástrojom na zaistenie autentickosti je **digitálny podpis**.

- **Anonymita:**

Je to vlastnosť, že určité **záznamy alebo transakcie nepripadnú ku žiadnemu jednotlivcovi**.

Nástroje pre zabezpečenie anonymity sú: **Agregácia** – kombinácia dát od viacerých používateľov.

**Mixácia** – agregovanie informácií z viacerých strán a spájanie ich do zložiek, ktoré sa nedajú

rozložiť. **Proxy** – dôveryhodní agenti, ktorí nahrádzajú skutočnú identitu používateľa. **Pseudonym**

– fiktívna identita používateľa, ktorý predstiera identitu.

## 2. Elektronický podpis

An eSignature refers to the electronic signing process that creates a legally binding agreement, while a digital signature tends to mean the cryptographic signature produced by public/private key signing.

A public/private key digital signature could be one of the signing methods used for eSignature.

The reason a digital signature is not an eSignature is that applying it alone won't create a legally binding agreement, for example, you could apply a digital signature to secure some data for transmission through an untrusted network (in fact this was the original purpose of the public/private key).

An electronic signature is any author identification and verification mechanism used in an electronic system. This could be a scan of your real hand-written signature, or any kind of electronic authenticity stamp. It's a generic term that covers a lot of authenticity measures.

A digital signature is a type of electronic signature.

### 1.2.1. Digitálny podpis

1. Hlavný nástroj na zaistenie autentičnosti
2. analogický ručnému podpisu, ktorý slúži ako dôkaz autorstva, resp. súhlasu s obsahom dokumentu

3. je to určitá dátová štruktúra, ktorá je závislá na dokumente, vzniká hashovaním tohto dokumentu a tento kód je zašifrovaný súkromným kľúčom, ktorý je jednoznačným vlastníctvom vlastníka dokumentu
4. digitalnu signaturu
5. na vytvorenie digitalnej signatury sa využije asymetrické šifrovacie algoritmi s verejným kľúčom (RSA, DSA) a bezpečne jednocestné kryptovacie algoritmy (kryptografické hashovacie funkcie MD5)
6. nie je možné dosiahnuť 100% autentičnosť údajov

How it works:

1. No one cares if somebody reads the document, we just want to be sure that it comes unmodified from given person.
2. Create hash of the document (SHA-256). Even the tiniest change in the document would result in completely different hash value.
3. Now encrypt the hash with the private key and embed this hash into the document we want to send.
4. Make the public key available (alternatively a website where ones can fetch)
5. To verify: decrypt the hash – if the public key decrypts it it means it comes from the person who has the public key. If the SHA-256 hash of the docs is the same as the previously decrypted hash, then no one modified the document.

Certification:

1. Digital certificate issued by a certification authority guarantees sender's identity.
2. The digital certificate contains a public key along with other information about the sender+ expiration date.

### 1.2.2. Digitalna obalka

Digitálna obálka: chráni symetrický kľúč pomocou public key

1. Bob vytvorí náhodný symetrický kľúč na jedno použitie
2. Zašifruje správu symetrickým kľúčom
3. Zašifruje jednorazový kľúč Aliciným public key
4. Pripojí šifrovaný symetrický kľúč k šifrovanej správe a odošle ju Alici
5. Iba Alica je schopná dešifrovať jednorazový kľúč a správu

### 1.2.3. SSL

„SSL" is the name that is most often used to refer to this protocol, but SSL specifically refers to the proprietary protocol designed by Netscape in the mid 90's. "TLS" is an IETF standard that is based on SSL, so I will use TLS in my answer. These days, the odds are that nearly all of your secure connections on the web are really using TLS, not SSL.s

TLS has several capabilities:

- Encrypt your application layer data. (In your case, the application layer protocol is HTTP.)
- Authenticate the server to the client.

Note: I wrote my original answer very hastily, but since then, this has turned into a fairly popular question/answer, so I have expanded it a bit and made it more precise.

#### **TLS Capabilities**

"SSL" is the name that is most often used to refer to this protocol, but SSL specifically refers to the proprietary protocol designed by Netscape in the mid 90's. "TLS" is an IETF standard that is based on SSL, so I will use TLS in my answer. These days, the odds are that nearly all of your secure connections on the web are really using TLS, not SSL.

TLS has several capabilities:

- Encrypt your application layer data. (In your case, the application layer protocol is HTTP.)
- Authenticate the server to the client.
- Authenticate the client to the server.

#1 and #2 are very common. #3 is less common. You seem to be focusing on #2, so I'll explain that part.

A server authenticates itself to a client using a certificate. A certificate is a blob of data[1] that contains information about a website:

- Domain name
- Public key
- The company that owns it
- When it was issued
- When it expires
- Who issued it
- Etc.

1. You can achieve confidentiality (#1 above) by using the public key included in the certificate to encrypt messages that can only be decrypted by the corresponding private key, which should be stored safely on that server.[2] Let's call this key pair KP1, so that we won't get confused later on. You can also verify that the domain name on the certificate matches the site you're visiting (#2 above).
2. But what if an adversary could modify packets sent to and from the server, and what if that adversary modified the certificate you were presented with and inserted their own public key or changed any other important details? If that happened, the adversary could intercept and modify any messages that you thought were securely encrypted.
3. To prevent this very attack, the certificate is cryptographically signed by somebody else's private key in such a way that the signature can be verified by anybody who has the corresponding public key. Let's call this key pair KP2, to make it clear that these are not the same keys that the server is using
4. Oversimplifying a bit, a certificate authority creates KP2, and they sell the service of using their private key to sign certificates for other organizations..

### 3. Email – Elektronicka posta

Heathers;MUA;POP/IMAP;SMTPMTA;SPAM;DNS;MX-record

Every mta adds a line to the heather.

Spoofing – changing the heathers

Private (Outlook, Thunderbird) and Public (gmail) email system

1. pouzivatel A napise spravu pomocou programu MUA (mail user agent)
2. zvolí e-mailovu adresu pouzivatela B,
3. odosle a MUA pouzije SMTP (simple mail transfer protocol) na to, aby poslal spravu MTA (mail transfer agentovi), kt prevadzkuje ISP (internet service provider) pouzivatela A
4. MTA sa pozre na cielovu adresu, vyhlada nazov domeny v DNS aby zisti mail exchange servery pre danu domenu
5. DNS server odpovie MX zaznamom, kde uvedie mail exchange servre pre danu domenu, v tomto pripade ISP server pouzivatela B
6. MTA odosle spravu pouzitim SMTP protokolu, ktory ho doruci do webschranky pouzivatela B

7. pouzivateľ B skontroluje novú poštu v jeho MUA, ktorý vyzdvihne poštu pomocou protokolu POP/IMAP

## 2. Termin 2

### 1. DNS and DNSSEC

#### 2.1.1. DNS

- domain name system
- DNS je system na spravu domenovych mien a ip adries
- protokol aplikacnej vrstvy, ktory preklada domenoveho mena na ip adresu
- uklada: Address (A), Mail Exchange (MX), Name Server (NS)
- prostrednik medzi potrebami cloveka a softveru
- poskytuje mechanizmus získania IP adresy pre každé meno stroja (lookup) a naopak (reverse)
- uvádza poštové servery (MX záznam) akceptujúce poštu pre danú doménu

#### 2.1.2. DNS SEC

- umožňujú zabezpečiť informácie poskytované DNS systémom v IP sieťach
- rozšírenie, ktoré klientom DNS umožňuje overenie pôvodu dát a ich integrity
- nezaistuje však zašifrovanie prenášaných dát a nezaručuje ich dostupnosť
- používa asymetrické šifrovanie (1 kľúč pre zašifrovanie a 1 kľúč pre dešifrovanie)

### 2. XSS

### 3. Symetricke a Asymetricke Sifrovanie

#### 2.3.1. Symetricke Sifrovanie

- sifrovanie a desifrovanie sa deje len pomocou jedneho kluca
- sila algoritmu zavisi od dlzky kluca a schopnosti uchovat kluc na oboch stranach vbezpeci
- vyhodou je nizka vypoctova narocnost
- priklad DES, AES

#### 2.3.2. Hashovacie funkcie

- hashovacia funkcia sluzi na kontrolu integrity dat
- napríklad či zasifrovaná sprava a odsifrovaná sprava su totzne
- priklad MD5
- hashovacia funkcia vracia sekvenciu znakov ktore sa navzajom porovnavaju



#### 4. Pretecenie Zasobníka – Buffer Overflow

### Example of Exploiting a Buffer Overflow

- Application login page

**Username must be between 6 and 24 characters**

- You type in any username followed by attack code

**Mikedansegliaoisacoolduderund1132%20shell.dll,c  
mdprompt**

- The application fails to check the input, puts it all on the stack, reads the first 24 characters
- A compiler or shell error can allow execution of code beginning at character 25

A situation where we're using some, probably low-level C function or something to write a string or some other variable - into a piece of memory that is only a certain length. But we're trying to write something in that's longer than that and it then overwrites the later memory addresses, and that can cause all kinds of problems.

#### 5. Security Modules

##### 2.5.1. DAC vs MAC

In this regard, Mandatory Access Control (MAC) and Discretionary Access Control (DAC) are two of the popular access control models in use. The main difference between them is in how they provide access to users. With MAC, admins creates a set of levels and each user is linked with a specific access level. He can access all the resources that are not greater than his access level. In contrast, each resource in DAC has a list of users who can access it. DAC provides access by identity of the user and not by permission level.

## 2.5.2. Bella-La Padula

### Bell-LaPadula model:

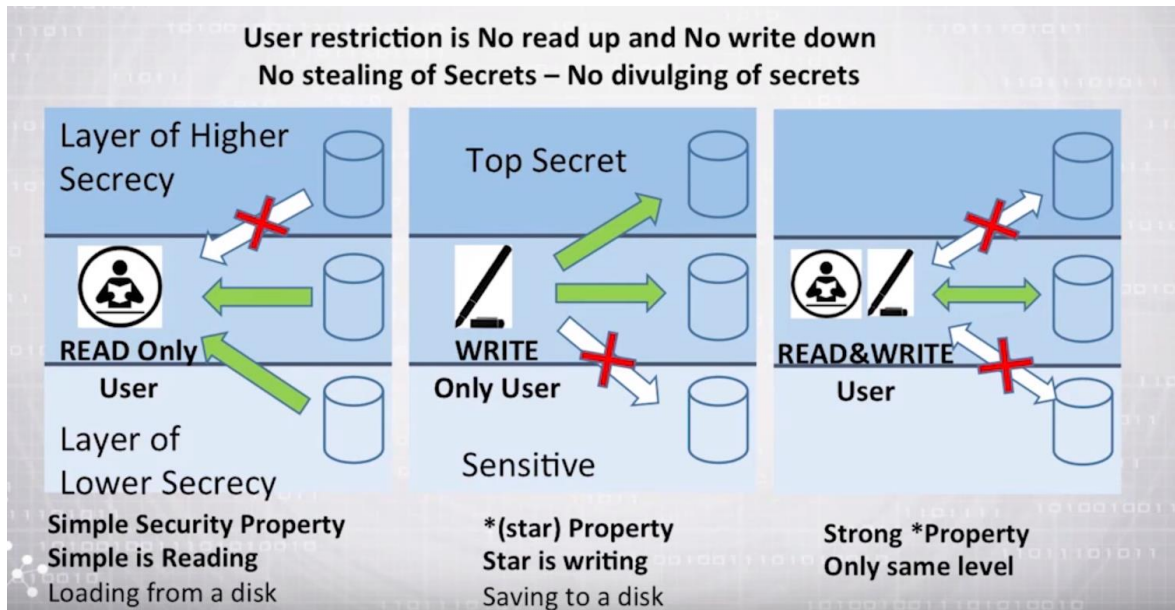
- **formálny** model pre riadenie prístupu
- **subjekty** a **objekty** majú priradené **bezpečnostné triedy**, ktoré riadia spôsob prístupu **subjektu** ku **objektom**:
  - top secret > secret > confidential > restricted > unclassified
  - subjekt má security clearance
  - objekt má security classification
- **prístupové módy**:
  - read - subjekt môže iba čítať z objektu
  - append - subjekt môže iba písať do objektu
  - write - subjekt môže čítať aj písať do objektu
  - execute - subjekt **nemôže** čítať ani písať do objektu, ale môže ho spustiť
- ak definované **viaceré úrovne dát**, tak **podmienky** sú definované **multilevel security (MLS)**:
  - subjekt **vysokej** úrovne **nemôže** dodať **informácie** subjektu **nižšej** úrovne, ak prúd informácií **nie je deklasifikovaný autorizovaným používateľom**
  - **MLS systém** musí spĺňať:
    - **no read up**:
      - subjekt môže čítať objekt s **menšou** alebo **rovnakou** bezpečnostnou triedou
      - ide o **simple security property**, tj. **ss-property**
    - **no write down**:
      - subjekt môže písať iba do objektu s **väčšou** alebo **rovnakou** bezpečnostnou triedou
      - ide o **star property**, tj. **\*-property**
- **Mandatory Access Control (MAC)** - prístup je zamietnutý, ak **ss-property** a **\*-property** nie sú splnené
- **Discretionary Access Control (DAC)** - **ds-property**, ktorá umožňuje **používateľovi** alebo **roly** dať prístup do súboru inému **používateľovi**, ale pod **MAC** obmedzeniami.



Confidentiality model, that's concerning with keeping confidentiality data secure. Inflexible.

Security Labels.

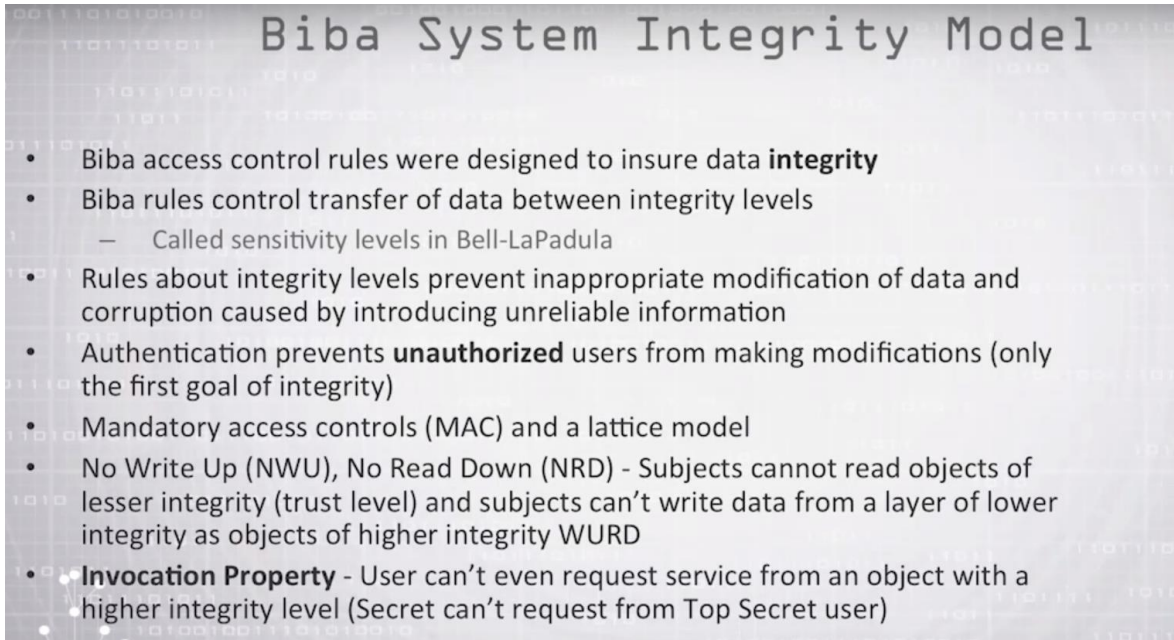
- The model is an inflexible, formal, state transition model of computer security policy that describes mandatory access control (MAC) rules
- Uses security/sensitivity labels on objects and clearances for subjects
- It was originally defined by DoD in TCSEC to support multilevel security and Mandatory Access Control (MAC) – Tranquil – No state changes
- Every object must be labeled from the most sensitive, "Top Secret", down to the least sensitive "Public."
- Systems are divided into subjects (users) and **labeled** objects
- State machine with a set of allowable system states
- Preserves security of information even as the system moves from one state to another state – Information flow model
- Only applies to secrecy (confidentiality) of information
  - \* (star) property - no write-down (NWD), by preventing subjects with access to high level data from writing the information to objects of lower access (no copy & paste into lower class) to prevent leakage
  - Simple security property – No read up (NRU) – Confidentiality
  - Discretionary: Uses an access matrix of subjects and labeled objects



### 2.5.3. Biba

#### BIBA Integrity model:

- rieši prípad keď **dáta** musia byť **dostupné** používateľom vo **viacerých** alebo **všetkých** bezpečnostných triedach, ale môžu byť **upravené** iba **autorizovanými** agentami
- základné prvky sú **podobné** ako v BLP (Bell-LaPadula), keďže obe pracujú so **subjektami** a **objektami**, kde **každému** je priradený **level integrity**
- In general the model was developed to address integrity as the core principle, which is the direct inverse of the Bell–LaPadula model.
- In general, preservation of data integrity has three goals:
  - Prevent data modification by unauthorized parties
  - Prevent unauthorized data modification by authorized parties
  - Maintain internal and external consistency (i.e. data reflects the real world)
- This security model is directed toward data integrity (rather than confidentiality) and is characterized by the phrase: "read up, write down". This is in contrast to the Bell–LaPadula model which is characterized by the phrase "read down, write up".
- The Biba model defines a set of security rules, the first two of which are similar to the Bell–LaPadula model. These first two rules are the reverse of the Bell–LaPadula rules:
  - The Simple Integrity Property states that a subject at a given level of integrity must not read data at a lower integrity level (no read down).
  - The \* (star) Integrity Property states that a subject at a given level of integrity must not write to data at a higher level of integrity (no write up)[2].
  - Invocation Property states that a process from below cannot request higher access; only with subjects at an equal or lower level.

The slide features a background of binary code (0s and 1s) and a grid pattern. The title 'Biba System Integrity Model' is centered at the top in a large, white, monospace-style font. Below the title, a bulleted list of eight points is presented in white text. The points describe the design goals and rules of the Biba model, including its focus on data integrity, control of data transfer between levels, prevention of inappropriate modifications, authentication to prevent unauthorized users, mandatory access controls (MAC), and a lattice model with No Write Up (NWU) and No Read Down (NRD) rules. The final point, 'Invocation Property', is highlighted with a small white circle next to the bullet point.

## Biba System Integrity Model

- Biba access control rules were designed to insure data **integrity**
- Biba rules control transfer of data between integrity levels
  - Called sensitivity levels in Bell-LaPadula
- Rules about integrity levels prevent inappropriate modification of data and corruption caused by introducing unreliable information
- Authentication prevents **unauthorized** users from making modifications (only the first goal of integrity)
- Mandatory access controls (MAC) and a lattice model
- No Write Up (NWU), No Read Down (NRD) - Subjects cannot read objects of lesser integrity (trust level) and subjects can't write data from a layer of lower integrity as objects of higher integrity WURD
- **Invocation Property** - User can't even request service from an object with a higher integrity level (Secret can't request from Top Secret user)

### 3. Bonus

1. URL shortener (goo.gl, tinyurl...)
2. Salting

Hashing algo : transforms input data into a simingly random fixed length string.

Rainblow table (Dictionary Attack) – precalculated hashes of most common passwords

Bruteforce attack – trying every single combination. The comlexity increases exponentially by increasing the length.

```
Stored Hash = Hash(Pass + Salt)
```

eg:

```
Pass: catsRcool
```

```
salt: P#)!z
```

```
So hash: catsRcoolP#)!z
```

↓

```
51bf5aec32fc50df19055a316f41805b
```

```
User Password Input: catsRcool
```

```
Website's backend adds the salt
```

```
Giving: catsRcoolP#)!z
```

Hash stored: Hash(Pass + pepper)

If:

Pepper = M

Password = catsRcool

Then stored hash...

Hash(catsRcoolM)

↓

b8b8bc09b11249cd21c9dd3c5f1e8847

Website cycles through all possible peppers...

catsRcoola

catsRcoolb

catsRcoolc

...

until... catsRcoolM

which matches the hash!

Since every user will have a completely different salt, this also avoids the problem with simple hashes, where we could easily tell if 2 or more users are using the same password; now the hashes will be different. We can also no longer take the password hash directly and try to google it. Also, with a long salt, a brute-force attack is improbable. But, if an attacker gets access to this salt either by an sql injection attack or direct access to the database, a brute-force or dictionary attack becomes probable, especially if your users use common passwords (again, like '123456'):



## The randomness issue

In order to generate a good salt, we should have a good random number generator. If php's `rand()` function automatically popped up in your mind, forget it immediately.

There is an excellent article about randomness in [random.org](https://random.org). Simply put, a computer can't *think* of random data by itself. Computers are said to be **deterministic machines**, meaning that every single algorithm a computer is able to run, given the exact same input, will always produce the same output.

Pass – the standard unix password manager.

### 3. Phishing

#### Link manipulation [\[edit\]](#)

Most methods of phishing use some form of technical deception designed to make a [link](#) in an email (and the [spoofed website](#) it leads to) appear to belong to the [spoofed organization](#).<sup>[19]</sup> Misspelled URLs or the use of [subdomains](#) are common tricks used by phishers. In the following example URL, `http://www.yourbank.example.com/`, it appears as though the URL will take you to the *example* section of the *yourbank* website; actually this URL points to the "yourbank" (i.e. phishing) section of the *example* website. Another common trick is to make the displayed text for a link (the text between the `<A>` tags) suggest a reliable destination, when the link actually goes to the phishers' site. Many desktop email clients and web browsers will show a link's target URL in the status bar while hovering the [mouse](#) over it. This behavior, however, may in some circumstances be overridden by the phisher.<sup>[20]</sup> Equivalent mobile apps generally do not have this preview feature.<sup>[21]</sup>

[Internationalized domain names](#) (IDN) can be exploited via [IDN spoofing](#)<sup>[22]</sup> or [homograph attacks](#),<sup>[23]</sup> to create web addresses visually identical to a legitimate site, that lead instead to malicious version.

Phishers have taken advantage of a similar risk, using open [URL redirectors](#) on the websites of trusted organizations to disguise malicious URLs with a trusted domain.<sup>[24][25][26]</sup> Even digital certificates do not solve this problem because it is quite possible for a phisher to purchase a valid certificate and subsequently change content to spoof a genuine website, or, to host the phish site without SSL at all.<sup>[27]</sup>