

E-VOTING WEB APPLICATION 1.3.3.2

SECURITY AUDIT REPORT

DECEMBER 2023

CLASSIFICATION	PUBLIC
REFERENCE	P021520
CUSTOMER	Federal Chancellery
LAST MODIFIED	2024-01-29

Client contact information

Federal Chancellery

SCRT SA contact information

SCRT SA
Rue du Sablon 4
1110 Morges
Suisse

Versions

DATE	VERSION	AUTHOR	DESCRIPTION
2023-12-19	0.1		Initial document
2023-12-27	0.2		Added vulnerabilities
2024-01-09	0.3		Minor modifications and conclusions
2024-01-11	1.0		Review
2024-01-23	1.1		Added remark for P021520-02

TABLE OF CONTENTS

Executive summary	4
Results summary.....	4
High level impressions	4
Security dashboard	5
Scope.....	5
Risks by level	5
Risks by remediation.....	5
Global risk level	5
Status by attacker profile.....	5
Identified risks.....	6
Proposed remediation plan	6
Technical summary	7
Scope.....	7
Restrictions.....	7
Results.....	7
Vulnerability summary.....	8
Additional remarks.....	8
Detailed results	9
Vulnerabilities and exploitation.....	9
P021520-01 Weak Multi-Factor Authentication (MFA).....	9
P021520-02 Insecure Web Messaging API use.....	11
Complements	14
Legend.....	14
SCRT Score.....	14
CVSS Score.....	14
Risk calculation.....	15
Context.....	15
Attempted attacks	15
Attack scope.....	15
Search for known vulnerabilities (vulnerability scanning).....	15
Network protocol analysis	16
Weak and default passwords discovery	16
Web applications	17
Network sniffing.....	17
Exploiting vulnerabilities.....	17
Additional attacks	18
Man-In-The-Middle.....	18
Social Engineering.....	18

EXECUTIVE SUMMARY

RESULTS SUMMARY

SCRT was contracted by the Federal Chancellery to perform a security assessment of the E-voting system developed by the Swiss Post. To this end, SCRT acted like real attackers and searched for vulnerabilities and weaknesses within the application to determine the risk for the voters and the secrecy and integrity of their votes.





During this assessment, two low-risk vulnerabilities were identified. The first risk emerges from the use of a weak multi-factor authentication system. Specifically, an attacker obtaining a voting card could potentially exploit the second authentication factor (the year of birth) using open-source intelligence. This risk poses a threat to the confidentiality and integrity of the voting process. If exploited, it could undermine the security and reliability of individual votes.

The second risk is due to a recent update that inadvertently included a script file, increasing the attack surface of the web application. While no immediate attack scenarios have been identified, this expanded attack surface presents a risk, as it could be exploited in future attacks when the codebase evolves.



Despite these risks, the E-voting system maintains a robust security posture. The overall risk level remains low, thanks to the well-hardened application and infrastructure. As noted in the additional remarks, SCRT recommends educating voters about basic cybersecurity principles, with a particular focus on the risks associated with phishing and compromised devices.

HIGH LEVEL IMPRESSIONS

STRENGTHS

-  WAF configuration
-  Previous findings remediation
-  Parameter filtering and validation
-  Code and documentation quality

WEAKNESSES

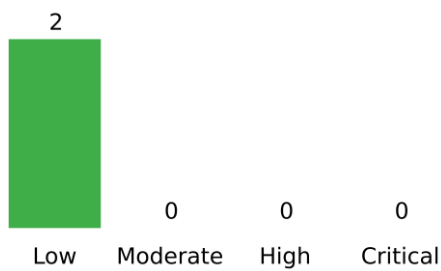
-  Multi-Factor Authentication
-  Anti-phishing protection

SECURITY DASHBOARD

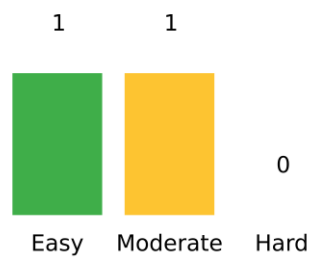
SCOPE

Type	White-box
Scope	Web application
Positioning	SCRT Offices
Schedule	2023-12-19 – 2023-12-27
Effort	12 days
Consultants	2

RISKS BY LEVEL



RISKS BY REMEDIATION



GLOBAL RISK LEVEL



ATTACKER PROFILES	RISK LEVEL
Without voting card	■ ■ ■ ■
With voting card	■ ■ ■ ■

STATUS BY ATTACKER PROFILE

OBJECTIVES	WITHOUT VOTING CARD	WITH VOTING CARD
Gain access to the internal network	✓	✓
Execute arbitrary commands	✓	✓
Vote confidentiality and integrity	✓	⚠

✓ NOT COMPROMISED
 ⚠ PARTIALLY COMPROMISED
 ⚠ COMPROMISED

IDENTIFIED RISKS

ID	RISK LEVEL	RISK DETAILS	RELATED FLAWS	FIX
1	LOW	An attacker intercepting a voting card could obtain the second factor (year of birth) using OSINT and compromise the confidentiality and integrity of a vote.	P021520-01	
2	LOW	Due to the involuntary inclusion of a script file, the attack surface of the web application is expanded. It could also cause unexpected behaviours.	P021520-02	

 EASY

 MEDIUM

 HARD

PROPOSED REMEDIATION PLAN

ID	ACTION	DIFFICULTY	RELATED RISKS
1	Use a stronger 2FA mechanism.	MEDIUM	1
2	Remove the inclusion of the Web Worker JavaScript file in the HTML page.	EASY	2

TECHNICAL SUMMARY

SCOPE

The scope of the audit includes the e-voting web application (release 1.3.3.2), which was reachable during the audit at the following address:

- » <https://it.evoting.ch/vote/#/legal-terms/D77A773516AB54473806FA0AEE5CEBFA>

A hundred voting cards were also provided to the auditors.

As the penetration test was performed as a white-box audit on an open-source project, the source code of the application was also available on GitLab:

- » <https://gitlab.com/swisspost-evoting>

RESTRICTIONS

No social engineering or denial of service attacks were performed during this audit.

RESULTS

The audit started with an automated reconnaissance phase, during which the server was scanned to identify and enumerate open services. This was followed by a web security scan aimed at detecting typical vulnerability signatures. However, the E-Voting System's perimeter includes protective measures such as a Web Application Firewall (WAF) and anti-Denial of Service (DOS) mechanisms, which prevented any further enumeration.

Then, the auditors proceeded with a manual security review of the open-source repositories. This comprehensive review confirmed that the issues identified during the bug bounty program have been properly addressed. However, a vulnerability was uncovered: a JavaScript code, intended to be loaded as a Web Worker for performing cryptographic operations, was also included in the main window of the web application. This allowed other windows to interact with it. Although no practical attack was identified during the audit, the inclusion of this script could lead to unexpected behaviours and increase the attack surface.

When manually testing the application, a second vulnerability was found: on the login page, the Multi-Factor Authentication (MFA) was deemed too weak, as the birth year could be easily deduced by searching the internet for the victim's social media profiles or other public records. Therefore, if an attacker steals or intercepts the envelope, he/she could compromise the confidentiality or integrity of the vote.

Nonetheless, the overall security level is considered as high and regular assessments are recommended to maintain this level. SCRT also advises conducting cybersecurity awareness campaigns for voters for the reasons detailed in the additional remarks section.

VULNERABILITY SUMMARY

ID	VULNERABILITY	IMPACT	PROBABILITY	CVSS
P021520-01	Weak Multi-Factor Authentication (MFA)	★★★★	★★☆☆	3.2
P021520-02	Insecure Web Messaging API use	★★☆☆	★★★☆☆	3.4

Explanations regarding impact, exploitation and CVSS scores can be found in chapter *Complements*

ADDITIONAL REMARKS

As highlighted in the previous SCRT report from October 2022, also by some other researchers (see <https://andreaskuster.ch/blog/2023/CVD-EVoting-Swiss-Post/>), the confidentiality of votes can be compromised, or voters can be misled into following different instructions if they fall victim to a phishing attack or have malicious software installed on their computer.

The E-Voting System has a robust mechanism in place to verify the correctness of the vote. This is achieved using the *Choice return codes* and *Vote Cast Code*, which are unique to each voter. Theoretically, this protection should be sufficient as voters are implicitly instructed to verify these codes at the end of the voting process. However, the reality is that users often rely more on on-screen instructions. Therefore, if a phishing website omits the instructions to check the Choice return codes, an unsuspecting user might simply enter the Return Code, thereby enabling the attacker to cast the vote on their behalf.

Typically, companies conduct cybersecurity awareness training for their employees to safeguard against phishing attacks. In light of this, SCRT recommends that the Federal Chancellery initiate cybersecurity awareness campaigns prior to the voting process and/or include a cybersecurity notice within the voter envelope. Although not fool-proof, this approach will help protect voters against phishing and other social engineering threats.

Following discussions with the Federal Chancellery, it appears that some cantons have already updated their voting instructions to indicate that control codes should never be inserted into the portal.

DETAILED RESULTS

VULNERABILITIES AND EXPLOITATION

P021520-01 WEAK MULTI-FACTOR AUTHENTICATION (MFA)			
SCRT		CVSS	
Impact	★★☆☆	Base	3.2
Probability	★☆☆☆	AV:P/AC:L/PR:L/UI:N/S:U/C:L/I:L/A:N	
PREREQUISITES		COMPROMISED ASSETS	
<ul style="list-style-type: none"> » Voting card » OSINT 		<ul style="list-style-type: none"> » Vote confidentiality » Vote integrity 	

AFFECTED SYSTEMS

it.evoting.ch

DESCRIPTION

The e-voting application uses an initialisation code and the user's birth year as two factors for authentication. The birth year, while personal, is not confidential information. It can be easily guessed or obtained through various means, making it a weak second factor of authentication.

EXPLOITATION

The e-voting login page relies on two secrets: the initialisation code and the year of birth. Although within the source code, there is also an option for entering the full date of birth, the provided test environment uses only the year. The system allows for 5 attempts, providing an attacker with multiple chances to guess the birth year correctly.

▲ Initialisierungscode
[? Was ist der Initialisierungscode?](#)

y 8 b f 3 a c h 9 r y 8 x f w d g z m q h 7 z d
×

Es spielt keine Rolle, ob Sie Gross- oder Kleinbuchstaben verwenden.

Geburtsjahr

1 9 9 0
× ⓘ

Falsche Eingabe des Geburtsjahrs. Sie haben 4 Versuch(e) übrig. Überprüfen Sie es bitte und versuchen Sie es erneut.

[Zurück](#)
Starten

Error message indicating that 4 more attempts are possible.

An attacker could use open-source intelligence (OSINT), such as social media searches on platforms like Facebook or LinkedIn, to acquire the year of birth. This information, in combination with a stolen voting letter containing the initialisation code, can be used to gain unauthorised access to the e-voting system. On the other hand, obtaining the complete date of birth is usually harder, as it cannot be easily deduced from education history listed on platforms like LinkedIn or other facts.

Once these two pieces of information are obtained, the attacker can exploit the system in two ways. They can repeatedly log in to the system until the voter reaches the validation step. This allows the attacker to view the voter's choice, thereby compromising the confidentiality of the vote. Alternatively, the attacker can cast a vote on behalf of the user. This compromises the integrity of the vote, as the user's choice is replaced with the attackers. However, this second form of attack can potentially be detected by the user when they attempt to cast their vote.

POSSIBLE SOLUTIONS

It is recommended to replace the year of birth with the complete date of birth as a second factor of authentication. This increases the difficulty for an attacker to guess or obtain this information.

Additionally, using lesser-known information such as the OASI number or part of it could also be an option. These changes would significantly enhance the security of the e-voting process by reducing the chances of an attacker successfully exploiting this vulnerability.

REMARK

This risk was already considered by the Swiss Post in section 4.1.5 of its **System specification document**. SCRT agrees with its conclusion that, even though the additional factor is weak, it still enhances the authentication but recommends that the cantons choose the option to use the full date of birth instead of only the year.

P021520-02 INSECURE WEB MESSAGING API USE

SCRT		CVSS	
Impact	★☆☆☆	Base	3.4
Probability	★★☆☆	AV:N/AC:H/PR:N/UI:R/S:C/C:N/I:L/A:N	
PREREQUISITES		COMPROMISED ASSETS	
–		<ul style="list-style-type: none"> » Increased attack surface » Unexpected behaviours 	

AFFECTED SYSTEMS

it.evoting.ch

DESCRIPTION

The Web Messaging API allows documents to communicate between each other within a browser. It can be used to circumvent some of the limitations imposed by the Same Origin Policy which prevents documents on different origins from communicating together.

The idea of the API is that any document can setup a handler for message events which is responsible for properly checking that incoming events are legitimate before taking any action following the event.

For example, the following very simple code might be used to define a message handler in a document which simply logs all incoming requests.

```

window.addEventListener("message", (event) => {
  console.log(event.data);
}, false);

```

Other documents can then interact with the document by using the `window.postMessage()` function as shown below.

```

var popup = window.open(/* popup details */);
popup.postMessage("Hello world!", "https://secure.example.net");

```

As indicated above, it is the responsibility of the event handler to perform any security checks and input validation before handling the data, as by default any domain can interact with the document and send arbitrary data through this API.

EXPLOITATION

The e-voting application uses a Web Worker (`crypto.ov-worker.js`) to perform cryptographic operations. However, since version 1.3.1.2, this script is also executed directly in the main window, likely by mistake. This script uses the Web Messaging API to listen for incoming messages without checking the origin.

```
/**
 * Handles a request from the api.
 * @param {MessageEvent} workerMessage, the message received from the worker.
 * @returns {Promise<unknown>} operation response.
 */
self.onmessage = function handleRequestFromApi(workerMessage) {
  const { operation, args } = workerMessage.data;
  let response;
  try {
    response = workerApi[operation].apply(workerApi, Array.isArray(args) ? args : [args]);
  } catch (error) {
    throw new Error(`Error calling the Worker API. [operation: "${operation}", error:${error}]`);
  }

  // Handling only the result and error (no progress or pending).
  // [...]
};

// [...]

const workerApi = {
  authenticateVoter: function(startVotingKey, extendedAuthenticationFactor, electionEventId, lang)
  {
    return authenticateVoterPhase(startVotingKey, extendedAuthenticationFactor, electionEventId,
    lang);
  },
  sendVote: function(selectedVotingOptions, voterWriteIns) {
    return sendVotePhase(selectedVotingOptions, voterWriteIns);
  },
  confirmVote: function(ballotCastingKey) {
    return confirmVotePhase(ballotCastingKey);
  },
  translateBallot: function(ballot, lang) {
    return translateBallot(ballot, lang);
  }
};
```

While this would not be an issue in the context of the Web Worker (as only the current window can post messages to it), the script being loaded in the main window allows a different window controlled by an attacker to send messages to it, triggering cryptographic operations. The included insecure message handler in the script listens for incoming messages without checking the origin. The handler includes operations such as `authenticateVoter`, `sendVote`, `confirmVote`, and `translateBallot`.

Proof of Concept

An attacker could exploit this vulnerability to trigger operations such as `sendVote`. This code opens the e-voting page in a new window and sends a message to it every second. The message contains an operation `sendVote` with arguments:

```
let win = window.open("https://it.evoting.ch/vote/#/legal-terms/D77A773516AB54473806FA0AE5CEBFA");
setInterval(function() {
  let injectedMessage = {
    operation: "sendVote",
    args: [
      ["5", "29"],
      []
    ]
  };
  win.postMessage(injectedMessage, "https://it.evoting.ch");
}, 1000);
```

Although the operation fails due to the session data (including private key and other cryptographic state) being stored only in the worker, it can lead to unexpected behaviour and increases the attack surface.

Remark

The `Cross-Origin-Opener-Policy` HTTP header prevents communication using the Web Messaging API from a different origin meaning that this attack is only possible on browsers that do not support this header. This is the case of Internet Explorer, which is listed as a compatible browser for the e-voting platform.

POSSIBLE SOLUTIONS

Remove the inclusion of the worker script in the HTML page

This can be achieved by removing `crypto.ov-worker` from the `scripts` array in the `e-voting/voter-portal/angular.json` file.

```
{
  "scripts": [
    {
      "input": "vendor/voting-client-js/dist/ov-api.js",
      "bundleName": "crypto.ov-api",
      "inject": true
    },
    {
      "input": "vendor/voting-client-js/dist/ov-worker.js",
      "bundleName": "crypto.ov-worker",
      "inject": true
    }
  ]
}
```

REFERENCES

- » https://developer.mozilla.org/en-US/docs/Web/API/Channel_Messaging_API
- » https://gitlab.com/swisspost-evoting/e-voting/e-voting/-/commit/61f57a21ab3c424e95a49fbce3e000243b605b9f?page=3&view=parallel#d449b45e4bd1f2f19cc2d69e2984febdc948d724_151_154
- » <https://gitlab.com/swisspost-evoting/e-voting/e-voting/-/blob/e-voting-1.3.3.2/voting-client-js/src/worker-api.js>

COMPLEMENTS

LEGEND

SCRT SCORE

For each vulnerability discovered and detailed in this report, SCRT provides a threat assessment based on two indicators, an **Impact** and a **Probability** of exploitation.

IMPACT	IMPACT OF THE VULNERABILITY IN CASE OF SUCCESSFUL EXPLOITATION ("HOW BAD?")				
☆☆☆☆	★☆☆☆	★★☆☆	★★★☆☆	★★★★☆	★★★★★
N/A	Weak	Medium	High	Critical	
PROBABILITY	PROBABILITY THAT THE VULNERABILITY WILL BE DISCOVERED AND EXPLOITED BY AN ATTACKER?				
☆☆☆☆	★☆☆☆	★★☆☆	★★★☆☆	★★★★☆	★★★★★
N/A	Low	Medium	High	Very high	

However, it is important to keep in mind that this assessment is solely based on the information available to the engineers at the time of the audit. The engineers are not necessarily aware of all the details regarding the vulnerable applications or systems. Consequently, these ratings should always be reconsidered based on the context of the information system as a whole.

CVSS SCORE

In addition to its own scoring system, SCRT also provides an evaluation based on the **Common Vulnerability Scoring System (CVSS)**, for each vulnerability.

As a reminder, CVSS is a vulnerability scoring system designed to provide an open and standardised method for rating IT vulnerabilities. CVSS helps organisations prioritise and coordinate a joint response to security vulnerabilities by communicating the base, temporal and environmental properties of a vulnerability. More information about the CVSS scoring system can be found here: <https://www.first.org/cvss/user-guide>

RISK CALCULATION

Each risk presented in this report is calculated as the product of an **impact** and a **probability** of exploitation, as defined in the matrix below.

		Overall Risk Severity			
Impact	CRITICAL	High	High	Critical	Critical
	HIGH	Moderate	Moderate	High	Critical
	MODERATE	Low	Moderate	Moderate	High
	LOW	Low	Low	Moderate	High
		LOW	MODERATE	HIGH	CRITICAL
		Probability			

SCRT provides an estimation of the effort required to fix each vulnerability and thus mitigate their associated risk. It should be noted that this assessment is based on SCRT's experience, and as such might not fully reflect the context of the company or organisation.

CONTEXT

The context of each vulnerability is defined by its prerequisites and a list of compromised assets. The prerequisites represent the conditions that are required for the exploitation of a given vulnerability (*e.g.*: social engineering). Compromised assets represent the theoretical or tangible result of its exploitation (*e.g.*: a user account).

ATTEMPTED ATTACKS

ATTACK SCOPE

The attacks performed by SCRT engineers during this audit cover the spectrum of attacks that could be attempted by an actual attacker against the targeted information system. These attacks thus cover "system" aspects (focused on machines and operating systems) as well as "applicative" aspects (focused on applications running on top of the system).

As an example of this layered attack approach, consider a (poorly coded) web application vulnerable to SQL injection, deployed on a correctly configured and patched web server. The "system" components of this application (the OS, the web server, and the DB engine) do not suffer from any known vulnerability. However, the "applicative" layer is flawed and thus compromises the security of the whole system.

SEARCH FOR KNOWN VULNERABILITIES (VULNERABILITY SCANNING)

Software development is a complex task, especially when developing very large applications such as operating systems, and often requires scores of developers in different teams working autonomously. It is therefore not surprising that these applications contain many hidden bugs and vulnerabilities (often due to development errors), even after they are put on the market.

These flaws, when they are then discovered – by security researchers for example or by the companies themselves – are often published to inform end-users and push developers to correct them. Many flaws are discovered and published daily, which are generally followed by the release of a new patch for the affected piece of software.

However, these publications do not only interest the developers trying to correct the flaws. They are also very interesting for hackers as they reveal vulnerable pieces of code in the software. Sometimes these flaws allow hackers to gain remote access on a machine. In parallel with the release of new patches, specialised websites often release exploit code for these same vulnerabilities. These are small programs which exploit the vulnerability and are often very easy to use. This makes it very important to apply patches as quickly as possible. Not doing so leaves the door open to malicious hackers who may exploit the vulnerabilities to gain access to the affected machine.

System administrators must therefore take extreme care in making sure that all systems are up to date and that the accessible services are not prone to known vulnerabilities. This is a constantly ongoing job as a seemingly secure machine one day may suddenly become the target of attacks the next after the publication of a new vulnerability affecting it.

To check whether any of the systems within the scope are vulnerable to known vulnerabilities, SCRT engineers will research information based on the reported versions of software discovered previously.

This is partly done with the help of automated scanners whose main goal is precisely the discovery of known vulnerabilities. However, a vulnerability scan is only a small part of a security audit and – on its own – cannot substitute a manual audit.

NETWORK PROTOCOL ANALYSIS

Multiple services use cleartext protocols to communicate. This means that data is not encrypted before being sent on the network, sometimes even while sending credentials. In this context it is often possible for an attacker to sniff network traffic in hope of discovering cleartext user names and passwords.

This is also true for many web applications that do not use HTTPS, or do not implement it in a secure way, even when they deal with sensitive information.

The level of security applied to the communications of a given service is therefore an important part of its security and must also be subjected to analysis.

WEAK AND DEFAULT PASSWORDS DISCOVERY

Many services used on a network are protected by a password. These can be remote access services such as SSH, FTP or private sections of a website, such as an administration panel.

In most cases, access to these secure areas will allow an attacker to gain access to sensitive or confidential information and in some cases compromise the machine entirely. For this reason, it is important that the passwords be secure enough to stop an attacker from gaining illicit

access. Indeed, however secure an application may be, if a user or administrator decides to use a weak password that can easily be guessed by an attacker, the security level cannot be guaranteed. It is extremely important that chosen passwords are not part of any dictionary, as they are often used by attackers in an automated way to gain access to a service.

To check the security level of the passwords, SCRT engineers test default and weak passwords on any service requiring authentication.

WEB APPLICATIONS

There are many different ways web applications may be attacked. New types of attacks are regularly discovered allowing attackers to circumvent older security mechanisms, therefore forcing developers to constantly improve their code to prevent these new attacks.

There is however a regularly updated repository of the most commonly discovered and exploited vulnerabilities in web applications: the Open Web Application Security Project's (OWASP) TOP 10.

However, vulnerabilities are not limited to what is published in the OWASP Top 10 and SCRT engineers are more than capable of identifying flaws that are not necessarily well documented thanks to their experience gained from years of penetration testing.

NETWORK SNIFFING

Within a local network, such as a corporate network, several different services are provided for the users, such as file sharing, FTP servers, remote administration and so on. Many of these services use cleartext protocols to communicate, meaning that data transiting on the network is not encrypted. In some cases, even the user's credentials are sent in this way.

It is therefore possible for a user located on this network to intercept the network traffic in order to gather credentials or confidential information. This is usually done with the help of an ARP poisoning attack, which allows an attacker to make a targeted client believe it is the default gateway and make the gateway believe it is the end client, which then leads to the attacker proxying all requests between the two.

Cleartext credentials can easily be found this way, but in cases where authentication details are encrypted, the use of "cracking" tools comes in handy and will allow an attacker to break any potentially weak passwords.

EXPLOITING VULNERABILITIES

One of the main differences between an intrusion test and a simple vulnerability scan, which is too often referred to in the same terms, is the fact that an intrusion test will truly simulate what an attacker may do when attacking a company.

Any vulnerability discovered during the audit is exploited by SCRT engineers as long as it is actually exploitable and in line with the rules of engagement determined during the kick-off.

This is the only way to know how dangerous the vulnerability truly is. It will allow one to know what kind of information an attacker may access by exploiting the flaw and whether they may leverage it to attack other systems.

ADDITIONAL ATTACKS

The following attacks are usually not performed during penetration tests as they would go beyond the scope of the targeted application or system. However, SCRT deems it important to mention them here because they could be a key element in the exploitation of certain vulnerabilities.

MAN-IN-THE-MIDDLE

A Man-In-The-Middle attack refers to a situation where the attacker is able to eavesdrop and alter the data transmitted between a client and a server, without any of them being able to notice the manipulation. An adversary can undertake such an attack only if they have access to specific locations on the network. Effective attacks can be launched from the local network (for example *ARP Spoofing* or *DNS Poisoning*). Additionally, any node of the network through which the client-server communication flows can be used to undertake a Man-In-The-Middle attack. ISPs as well as governments are therefore often considered as having the possibility (legitimately or not) to undertake these kinds of attacks.

SOCIAL ENGINEERING

Users are frequently one of the attacker's primary targets. Sophisticated attacks (*e.g.: phishing, phoning*) are often developed in order to manipulate victims. When stated as a prerequisite for a vulnerability, social engineering means that an attacker must have some kind of interaction with their victim in order to trick them into performing an action desired by the attacker, such as clicking on a link or opening an e-mail attachment.