Management of Security

Data diode for secure transmission of election results

KEITA Alhassane

RIGAS Theofanis

VANSPOUWEN Tristan

May 31, 2017

Contents

Chapter 1

Concept of Operations

1.1 Introduction

This Concept of Operations (CONOPS) document concerns the security enhancement of the transmission of national election results to media outlets and the general public. In the current configuration, the Election Management System (EMS) is connected to a public server, where results are published. We believe this approach is insecure, as an attacker can potentially compromise the public server and then extend his attack to the secure EMS environment, compromising the voting system.

We propose the installation of a one-way data diode between the EMS and the public server. This diode will ensure that information can only flow from the EMS to the server and never the other way around. Thus, the election results will be transmitted without opening any doors to attackers. Our goal is to ensure the security of the voting system assets and of the elections results stored within.

In the sections that follow we describe the system currently in place and the system we propose as a replacement. We then present the proposed system's capabilities, its potential interactions with administrators and end users, as well as the Application Programming Interfaces involved (APIs).

1.2 System Description

1.2.1 Current system

The current EMS is a proprietary system that handles many tasks connected to the elections, such as candidate registration, ballot generation, results collection, winner proclamation and election results publication. During the election process, it receives the results transmitted by the voting machines [1]. The results are exported in XML format and transmitted to the public server, where they can be consulted by the press and the general public via a website. This exportation process is repeated multiple times as more data arrives from the voting stations, until the results are finalized.

[1]This part of the system is out of the scope of this CONOPS. We simple assume that the results are available to the EMS and ready to be exported.

The central EMS computer, which collects, counts and exports the results is installed in a secure area, separated from the rest of the floor by a glass wall. Access to this area is restricted to authorized personnel. The election results are forwarded as they become available to the public server through a TCP connection and are received by the proprietary EMS software installed on this server. They are then published on the election results website.

1.2.2 Proposed system

We propose the replacement of the connection between the EMS and the public election server with a data diode. Data diodes are hardware devices that allow data to move only in one direction. They are commonly installed between a secure environment and an insecure side. The unidirectional data flow means that user data inside the secure environment is kept confidential, while they can benefit from access to data from the insecure side.

In the context of the voting process, our goal is slightly different. We want the election results to be able to get out of the secure voting environment, while blocking any data from coming inside. This is to prevent attackers from tampering with the EMS and the rest of the voting system's assets or manipulating the results stored in the system. In order to accomplish this, we propose the installation of a data diode in the inverse direction, as pictured in the image below:

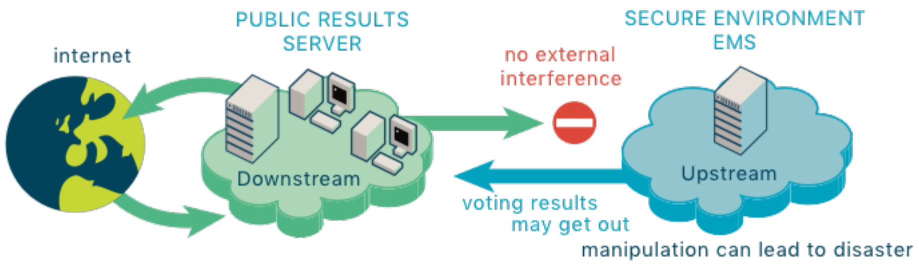


Figure 1.1: Data diode between the EMS and the public server

The diode will be placed between the secure voting environment and the results server and will block all incoming traffic to the EMS (see also Figure 1.2 in the following section).

1.2.3 Scope

We consider as parts of the proposed system: the data diode (consisting of two dedicated servers and an optical fiber connection between them) as well as the API and any related back-end services that will be hosted on the public server and will be used to publish the election results (results database etc.). A more precise analysis of the system boundaries is available in the "System Characterization" section of the Risk Assessment document (see 2.2.1).

The EMS and the public server as physical systems are out of scope, with the exception of some configuration actions that will be detailed in the sections to follow.

1.3 Analysis

1.3.1 System description and components

The system will be composed of a data diode and an API used to publish the election results. The diode is composed of two 1U rack servers. We will characterize one of them as ”high”, which is connected to the secure environment and the other as ”low”, which is connected to the part of the network which comes in contact with the public internet. The diode is configured for a high to low (H2L) data flow.

More specifically, the two servers will include the following components:

	Hardware	**Network connections**	**OS**	**Software**
High Server	1U Rack power button (front)	Ethernet jack (incoming traffic)	UNIX	Webserver XML signing tool FTP over UDP transmission software
Low Server	1U Rack power button (front)	Ethernet jack (outgoing traffic)	UNIX	Webserver Results daemon FTP over UDP reception software

Table 1.1: Diode components

These servers will be connected using a fiber-optic network link. To create a unidirectional flow of data, the receiver transceiver will be removed from the high part and the send transceiver from the low part. Any link failure mechanism will also be disabled.

The API will be hosted on the public server. It will be written in PHP and use a database to store temporarily the results it receives from the EMS. For a detailed description of the routes it will expose, see section 1.3.2.

1.3.2 System Capabilities

The following sections explore the capabilities of the proposed system.

Transfer of voting results

The main function of the system is the transfer of voting results from the EMS to the public server. This transfer will happen in real time, as soon as partial results are available for export in the EMS. The results arrive to the EMS via satellite connection from the voting stations (this part of the system is out of the scope of this CONOPS) and are automatically aggregated by the EMS software.

The same software exports the results into a series of XML files, and each file corresponds to a specific region of the country. There files are then periodically forwarded to a preconfigured IP address using a POST request. The only parameter we can control on this system is the address of the server were the results will be POSTed.

In the past, the files were directly POSTed to the public server. In the proposed setup, the results will be POSTed to the IP address of the high part of the data diode [2]. A PHP server will be listening for POST requests in order

[2] The high and the low part of the data diode will have static IP addresses that will be freely configurable using the web interface (see section 1.3.4)

to receive the incoming XML files. The files will then be placed into a special folder, which will be automatically synchronized with the other side of the diode.

When the results become available on the other side of the diode, they will be forwarded to the public server by a daemon.

Isolation of the EMS environment

The data diode is made up of two parts, the high and the low. Data can only flow from the high to the low part of the diode. In the proposed setup, the EMS will be on a different LAN than the LAN of the public server. The high part of the diode will be connected to the secure network. The low part of the diode will be present in the same LAN as the public server, reachable through the Internet.

During the transmission of the results, no data flows will be able to enter the secure environment, thus reducing the risk of attacks. The only way data can move is out from the secure environment.

Result verification via authentication

It is important to verify the authenticity of the results received on the public server before making them available to the public through the API. A common way of ensuring the authenticity of data is by using a cryptographic signature scheme, like RSA signatures. In this kind of scheme, the results are signed by the sending party using a private key. The receiving party can verify the signature upon reception of the results using a different, public key. Any alteration of the results can thus be detected during the signature verification process and the offending data can be rejected.

In the current system, the election results exported by the EMS are not signed. To overcome this problem, we propose that the signing should take place inside the high part of the data diode. When the results arrive on the diode, they will be signed using a 2048-bit private RSA key. The results are in the form of an XML document, and the signature can be placed inside [3] using a special signing tool. Then the results will go through the diode and will be received by the public server.

The public server will then use the same tool to verify the signature on the XML file using the public key, which will be obtained by the low part of the diode. If the signature passes the test, the results are really coming from the EMS and they can be published.

Private and public key generation and transmission

As we've described above, the high part of the diode will use a private RSA key to sign election results and the low part of the diode will be able to transmit the corresponding public key used for the verification of the signature to the public server.

These keys will be generated automatically using the web configuration interface (see section 1.3.4) on the high part of the data diode. The private key will

[3]Following the XML Signature Syntax and Processing recommendation at https://www.w3.org/TR/xmldsig-core/

be stored on the high part and the public key will be automatically transmitted and stored on the low part of the diode, ready to be used.

To secure the transmission of the public key and avoid any man-in-the-middle attacks, the key will be accompanied by an HMAC code, calculated using a shared secret present on the low part of the data diode and on the public server. This shared secret will be pre-installed on the relevant components of the system.

Publishing of results

The whole purpose of the system is to make election results available to the public. Once the results are received by the public server through the data diode, they will be parsed and then made available through a dedicated REST API. The API will be freely accessible via the address `http://www.france-elections.fr/api` and will expose the following endpoints:

- GET /api/regions/regionName : results for specified region
- GET /api/regions : names of regions
- GET /api/country : global results
- GET /api/candidates: names of candidates

1.3.3 Users

There are two kinds of users susceptible to interact with the components of our system (see also 1.2.3): administrators and regular users.

Administrators

Interactions with the data diode component will be strictly limited to administrators. The high part of the diode will be uniquely reachable from the interior of the secure part of the network, so the administrator will have to be accredited to have access to the area. The low part of the diode will be reachable from the public server's LAN.

Administrators will be provided with a set of one-time credentials for the first connection to the diode. Each side of the diode will have its own set of credentials. They will be required to change those credentials during the first connection.

Regular users

Regular users are meant to interact with the API hosted on the public server. No authentication will be required to access the routes.

1.3.4 Interfaces

In the proposed configuration, there are two kinds of interfaces: one for the system administrators to interact with and configure the data diode device, and one for the end users to consult election results. We plan to implement the following:

- Administrators will be able to connect to and configure the parts of the diode connected to the EMS and the public server through a web interface (similar to the interfaces proposed by routers). They will have the possibility to configure the IP address of the device, monitor data flows and change connection information. We believe that providing a web interface (instead of a command line one for example) will facilitate the work of administrators and the installation of the diode device in the existing infrastructure.

  Providing a web interface also means that every administrative interaction with the diode will be constrained to the provided functionalities and that there will be no need to provide an SSH connection to the diode, thus decreasing the attack surface.

- The general public and the media will be able to consult the election results via a REST API implemented on the public server. An official website will also be available on the public server where the results will be presented in real time.

1.3.5 Integration with the existing system

Our proposal involves virtually no changes to the existing secure voting environment or the EMS. The software currently installed on the EMS computer will forward the exported results to the diode. The diode software will replace the EMS software previously installed on the public server and will forward the results to it. The new API will finally be installed on the public server.

No additional personnel will be needed, as the installation, control and maintenance of the system can be carried out by the existing system administrators.

1.3.6 Diagrams

The following diagrams present the proposed system's activity as well as the information flow and the interactions with the users:

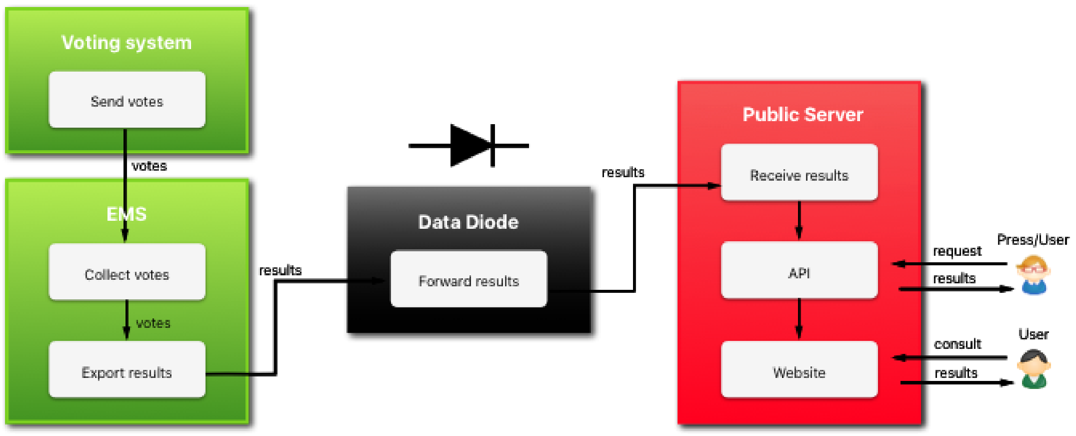


Figure 1.2: Information flow between the components of the systems and the end users

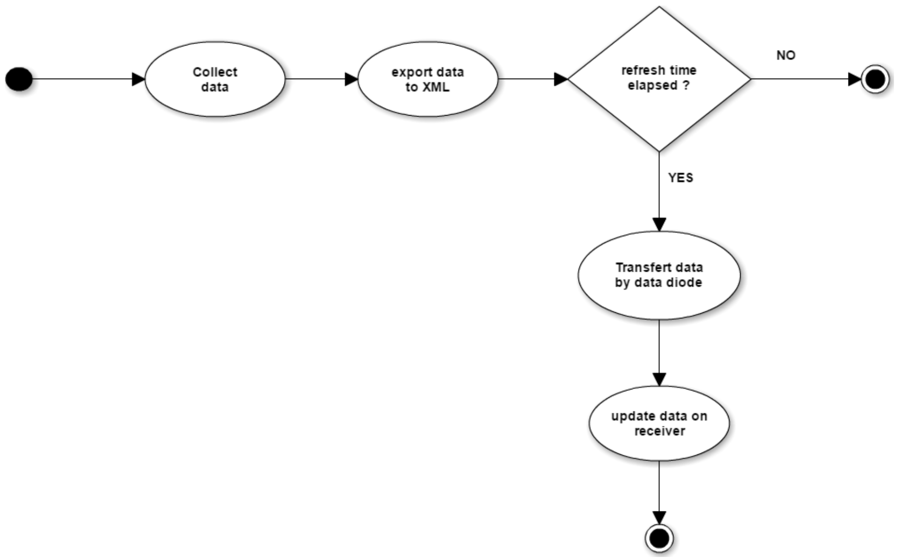


Figure 1.3: System activity diagram - Flow of results between the voting machines and the public server

1.4 Conclusion

In this CONOPS, we propose the installation of a data diode between the EMS and the public server that publishes election results. With minimal changes to the existing system, this will result in a better protection of the voting systems assets and thus enhance the security of the voting process.

Chapter 2

Risk Assessment

2.1 Introduction

The purpose of this document is to explore potential risks associated to the process of relaying the presidential election results to the public via a data diode. The data diode will be installed between the Election Management System (EMS), positioned inside a secure network, and the public server, which communicates with the public internet. Its role will be to ensure results will arrive to the public server from the EMS without the risk of any attacker introducing himself inside the secure environment.

In the sections that follow, we first identify the threat actors that could attack or endanger the normal functions of the system. We then focus on potential vulnerabilities and evaluate the overall risks they could pose to the proposed system. We finally conclude with a series of recommendations to be taken into account during the implementation and use of the system.

The assessment was based on the NIST SP800-30 risk management guide (`http://csrc.nist.gov/publications/nistpubs/800-30/sp800-30.pdf`).

2.2 System Characterization

The proposed system is composed of a data diode and an API to be installed on a public server [1]. The data diode comprises two servers, a high one connected to the secure environment and a low one, connected to the part of the network with access to the public internet. A detailed description of the system, its components, its interfaces and its users can be found in the Concept of Operations document attached to this Risk Assessment (see section 1.3) .

2.2.1 System boundaries

The demarcation between the data diode and the Local Area Networks (LANs) is the physical ports on the switches that connect the data diode to the two parts of the network. The switch and other voting system components such as the EMS are not considered to be part of the system.

[1]The term ”system” will be used in the rest of the document to designate the pair diode - API.

Concerning the API, the system boundary is the following directory and its subdirectories on the public server: var/www/html/api, as well the configuration and table files related to the database used by the API. The rest of the public server infrastructure is not considered as part of the system.

Finally, the administrators interacting with the data diode and the API for configurations purposes are also considered part of the system.

2.3 Threat Identification

Having clearly defined the system and its boundaries, we will proceed to identify potential sources of threats.

2.3.1 Unauthorized users

Script kiddies

Script kiddies are individuals that have no special programming skills and use ready-made scripts or tools to attack or deface websites and services. A script kiddie could attack the API hosted on the public server using an automated tool.

Hackers

Hackers could attack the system for personal reasons (ego, bragging rights for bringing down part of the election infrastructure). Possible actions include DDoS attacks, social engineering and reconnaissance/sniffing of the network.

Terrorists

Terrorists could try to sabotage the election process and the announcement of the results. They could perform any of the attacks described in section Hackers. They could also try to physically attack the building where the processing of the results takes place.

Foreign governments

Foreign governments could attack the system for political and economic reasons. They could target the election process in order to destabilize the country and harm the public's confidence in the results. They could perform attacks similar to those described in section Hackers but with significantly more resources.

2.3.2 Natural disasters and environmental hazards

The system could become nonoperational in case of a natural disaster like an earthquake damaging the building. The same applies in case of a power outage or a network outage, be it due to natural phenomena (storm, thunder) or some kind of systemic failure. A fire in the building could also put the system off line. Finally, the public server hardware, where the API is hosted, or the diode hardware could malfunction.

2.3.3 Insiders

Malicious insiders

The data diode and the API can also be attacked from the inside of the network. Persons not accredited to perform administrative tasks on the system could try to gain access to it, by curiosity or even for monetary again. Administrators of the system could try to sabotage it from the inside.

Unintentional errors

The data diode and the API need to be configured properly in order to enable the flow of election results. Any misconfiguration could cause the system to malfunction. The same holds for any bugs in the software components of the system.

2.4 Vulnerability identification

Based on the threat actors we have identified in the previous section, we will now identify possible vulnerabilities of the system. The term ”vulnerabilities” refers to weaknesses that can be exploited by an attacker in order to compromise the confidentiality, integrity or available of the system or of one of it’s components.

For each vulnerability identified, we give a short title as well as an ID number that will be used to refer to the particular vulnerability throughout the rest of this document. We then specify the threat actor and the type of system compromise and we provide a summary of the scenario we are considering.

ID	Vulnerability	Threat actor	Risk of Compromise of	Summary
1	Compromise of data diode login credentials during transmission	Unauthorized users, Insiders	Integrity, Availability, Confidentiality	The administrator connects to the low part of the data diode via the network interface. An attacker on the inside of the insecure side of the network sniffs the traffic or performs a man-in-the-middle attack and recovers the credentials in cleartext
2	DDOS attack on the API	Unauthorized users	Availability	The attacker uses a botnet to send an unusually high number of requests to the API. The API is overwhelmed and crashes, making the election results unavailable to the public

3	SQL injection attack on the API database	Unauthorized users	Integrity	The attacker uses a parameter of the API requests as a vehicle for an SQL injection. He alters or destroys the contents of the API database
4	Misconfiguration	Insider	Integrity, Availability	The data diode is misconfigured by the administrator (either its network configuration or the signature keys). Elections results can't be received or verified by the public server
5	Brute force data diode credentials	Unauthorized users	Integrity	An attacker on the inside of the insecure side of the network accesses the login page of the low part of the data diode and tries to brute-force the login credentials by trying a large number of possibilities. They gain access to the data diode software
6	Data diode hardware failure	Natural	Availability	One or several of the data diode's hardware components (servers, optical cables etc.) fail, rendering the diode inoperable. The election results cannot transit to the public server
7	Compromise of data diode login credentials via social engineering	Unauthorized users	Confidentiality	The attackers obtain the data diode credentials using a social engineering attack, such as fishing. This is possible if the administrator of the data diode reuses a set of credentials on several machines or services

8	Accidental exposure of data diode login credentials	Insider	Confidentiality	The data diode login credentials are displayed in plain sight in the interior of the building, for example they are noted somewhere on a piece of paper. The attacker simply copies them and gains access to the system
9	Power outage	Natural	Availability, Integrity	A local or generalized power cut leaves the data diode without power. The system is inoperable and election results cannot be published
10	Fire	Natural	Availability, Integrity	A fire breaks out in the building, due to accident or sabotage. The system is destroyed
11	Software bugs	Insiders, Unauthorized users	Integrity, Availability, Confidentiality	The data diode and the API use several software components that cannot be guarantied to be bug free. A bug in one of these components renders the system unstable or inoperable, or creates a breach exploitable by an attacker
12	0-day attacks	Unauthorized users	Integrity, Availability, Confidentiality	A hacker leverages a 0-day vulnerability in one of the software parts of the data diode or the API and uses it to introduce himself into the system

13	Invalid or malicious results files	Unauthorized users	Integrity, Availability	The attacker breaks into the unsecured part of the network and sends malicious result files to the API. These files may contain malicious code (Trojan horse), attachments or executables, or even commands embedded in XML (XML External Entity attack), that are processed by the parser and executed by the system
14	Unauthorized remote connection to the data diode through vulnerable service	Unauthorized users, Insiders	Integrity, Availability, Confidentiality	An attacker uses a misconfigured remote connection service (such as SSH) to obtain access to the data diode system

Table 2.1: List of vulnerabilities

2.5 Control analysis

In order to evaluate the risks that the vulnerabilities presented above pose to the proposed system, we first need to take a look at the controls already in place or currently planned. We will concentrate our attention on controls relevant to the threats and vulnerabilities that can potentially affect our system.

2.5.1 Infrastructure security

The EMS, the public server, as well as the proposed data diode system are housed in the Ministry of Interior Computer Center. Access to the floor is controlled via a card-key access system. This system allows monitoring, logging and, if necessary, auditing of the employees accesses to the Computer Center.

In addition, the EMS and the data diode are placed within a special secure area of the Computer Center, separated from the rest of the floor by a hardened glass panel. Access to this area is also controlled by the card-key access system and requires special privileges, that are reserved to system administrators and occasionally other members of the staff.

The Computer Center is equipped with a Halon inert gas fire suppression system that can extinguish fires by depriving them of oxygen. This system can suppress fires without causing significant damage on the hardware. The Computer Center is also equipped with a UPS (Uninterruptible Power Supply) that can provide emergency power to all systems housed in the center in case of power outages.

2.5.2 IT Security and password policy

The Ministry of Interior's IT department has a fully documented security policy. Concerning the sensitive system components, such as the EMS and the data diode, this policy enforces the use of passwords in order to access the systems. It also requires that passwords have a minimum length of at least 8 characters and consist of a combination of letters, numbers and at least 1 special character or symbol. Finally, it asks for passwords on sensitive systems to be changed every 30 days. This policy is however not enforced on a software level.

The same policy requires that no passwords transit in cleartext inside any part of the local network. Administrators are required to never divulge their passwords, to immediately change their passwords in case of compromise and to notify immediately the security team. All components must allow their users to easily change their passwords.

2.5.3 System security

The Computer Center network is protected by an IDS (Intrusion Detection System) that continuously monitors the network for malicious activities. When anomalies are detected, they are immediately reported to the system administrator.

A complete vulnerability scan of all system components that will be used during the presidential elections, naturally including the EMS, the data diode and the public server is currently scheduled for 45 days before the elections. The scan will be carried out by an external auditing company. The scan must at all cases be completed before the last month leading to the elections, so that any findings can be addressed before the system's deployment.

2.6 Likelihood determination

Taking into account the vulnerabilities and the controls identified above, we determine the probability of each vulnerability being exploited in the following table:

ID	Vulnerability	Likelihood	Comments
1	Compromise of data diode login credentials during transmission	High	A part of the data diode is connected to the non-secure part of the network which communicates with the public internet. Traffic on this part can be sniffed by an attacker and lead to compromise of the credentials if they are transmitted in cleartext
2	DDoS attack on the API	High	DDoS attacks are increasingly popular and can even be performed by attackers with minimal computer skills. The public API relaying the election results is a prime target

3	SQL injection attack on the API database	Moderate	SQL injections are common attack vectors for web APIs but the election API is quite simple and exposes only one route that accepts user input. It should be relatively easy to secure
4	Misconfiguration	Low	The data diode and the API are relatively easy to configure. Any configuration problem should be easily detectable and correctable
5	Brute force diode credentials	Moderate	A brute force attack on the diode can take place from the insecure side of the network. It is relatively simple to perform and common in attacking scenarios. Respecting password format policies should reduce the likehood of success
6	Data diode hardware failure	Low	The hardware will be properly tested and risk of failure will be low
7	Compromise of data diode login credentials via social engineering	Low	Login credentials for the data diode are only known by system administrators. They are generally informed about social engineering attack vectors (phishing etc.) and should not easily divulge this kind of information by accident
8	Accidental exposure of data diode login credentials	Low	The data diode will be placed inside a secure room where access is controlled by a card-key system. The probability that credentials are exposed this way is minimal
9	Power outage	Low	The building is protected by a UPS system and power outages should not affect the system
10	Fire	Low	The floor is equipped with a fire extinguishing system specifically designed for hardware installations. The fire should be contained without impacting the system

11	Software bugs	Low	The system will be appropriately tested before deployment and its set of functionalities is quite limited. Bugs affecting its performance or security should be difficult to find
12	0-day attack	Low	0-day vulnerabilities are difficult to find and the system will include the least amount of services and software possible, so the attack surface is limited
13	Invalid or malicious results files	Moderate	The attacker needs to introduce himself in the insecure part of the network to communicate with the API (or he can attack it from the exterior if the API is not properly secured). XML parsers are more often than not vulnerable to this kind of attack
14	Unauthorized remote connection to the data diode through vulnerable service	Moderate	There is a large number of attacks exploiting misconfigured or badly protected remote connection services. Not all of them are guaranteed to be stopped by the IDS already in place

Table 2.2: Likelihood determination

2.7 Impact Analysis

Taking once more into account the vulnerabilities identified in Table 2.1, we analyze the impact their exploitation would have on the system:

ID	Vulnerability	Impact	Comments
1	Compromise of data diode login credentials during transmission	High	The attacker could change the configuration of the system during the critical election period or could lock the administrators out of the system. The integrity and availability of the system would be compromised

2	DDoS attack on the API	High	The duration of the election process is rather short and DDoS attack mitigation can take a lot of time. Blocking the public's access to election results could destabilize the election process. The availability of the system could be severely compromised
3	SQL injection attack on the API database	High	If the database content is altered, incorrect results would be published to the API users
4	Misconfiguration	Low	The results would be unavailable until the configuration is fixed, but administrators will likely respond quickly and mitigate the problem.
5	Brute force diode credentials	High	Same impact as the compromise of credentials during transmission
6	Data diode hardware failure	High	By design, the data diode is a single point of failure for the transmission of results. A potential hardware failure would block the results publishing process, thus impacting the availability of the system
7	Compromise of data login credentials via social engineering	High	Same impact as the other cases of compromise of credentials
8	Accidental exposure of data diode login credentials	High	As above
9	Power outage	High	The system would be inoperable
10	Fire	High	The system would be destroyed
11	Software bugs	High	A bug in the data diode software could impact the transmission of results thus compromise the election process
12	0-day attack	High	The attacker could use the 0-day vulnerability to take control of the API or the diode and block or compromise the transmission of election results
13	Invalid or malicious results files	High	If the attacker succeeds in executing arbitrary code in the API, he could change the results communicated to the public or bring down the system

14	Unauthorized remote connection to the data diode through vulnerable service	High	The attacker would have access to the configuration of the data diode. He could sabotage the system or alter the results, jeopardizing the election process

Table 2.3: Impact analysis

2.8 Risk determination

Based on the likelihood and the projected impact of every risk, we determine an overall risk rating for every risk we've identified in section 2.4. In order to quantify the risks, we use the following values:

- **Risk likelihood**: Each risk likelihood rating presented in table 2.2 is assigned a numerical value of 0.1 for low, 0.5 for moderate and 1.0 for high likelihood.
- **Risk impact**: Each risk impact rating presented in table 2.3 is assigned a numerical value of 10 for low, 50 for moderate and 100 for high impact.

For each risk, the numerical values corresponding to the likelihood and the impact are multiplied. The results are interpreted using the scale below:

- **Between 1 and 10**: Low overall risk
- **Between 25 and 50**: Moderate overall risk
- **100**: High overall risk

ID	**Vulnerability**	**Risk score**
1	Compromise of data diode login credentials during transmission	High
2	DDoS attack on the API	High
3	SQL injection attack on the API database	Moderate
4	Misconfiguration	Low
5	Brute force diode credentials	Moderate
6	Data diode hardware failure	Low
7	Compromise of data login credentials via social engineering	Low
8	Accidental exposure of data diode login credentials	Low
9	Power outage	Low
10	Fire	Low
11	Software bug	Low
12	0-day attack	Low
13	Invalid or malicious results files	Moderate
14	Unauthorized remote connection to the data diode through vulnerable service	Moderate

Table 2.4: Overall risk score

2.9 Control recommendations

Having identified a series of vulnerabilities that could be present in our system and quantified the overall risk associated with each one, we proceed to emit a series of control recommendations, designed to mitigate this risk. They concern both the implementation of the data diode and the API, as well as their use during the election process.

Every recommendation is accompanied by an ID number which refers to the ID of the particular vulnerability it addresses.

2.9.1 Configuration through web interface (ID: 4,14)

We recommend allowing administrators to access and eventually modify the data diode configuration only through a configuration interface - similar to a router's configuration interface for example.

This measure would first of all restrict the risk of misconfiguration by constraining the interactions of the administrators with the system. It would also eradicate the need of providing some kind of remote access to the data diode system for configuration purposes.

2.9.2 Removal / blocking of all remote access services and ports (ID: 14)

All remote access services (like SSH) should not be present (or be explicitly disabled) on the data diode OS to limit the attack surface and the potential for exploitation.

In addition, all ports other than the port used to access the web interface and accept / send the election results should be explicitly blocked.

2.9.3 Use of HTTPS for connections to the data diode configuration interface (ID: 1)

The IT Security policy specifies that no passwords should transit in cleartext inside the network. In order to comply to this policy and mitigate the risk of credential theft by an attacker, we recommend that HTTPS should be used when connecting to the data diode configuration panel. All HTTP connections to it should be rejected and redirected to HTTPS.

2.9.4 DDoS protection (ID: 2)

The public server is currently protected by an IDS, which can potentially detect DDoS attacks but cannot actively defend against them. Replacing the IDS by an IPS (Intrusion Prevention System) which can deploy active countermeasures to stop ongoing DDoS attacks is a solution than could be effective, but is infeasible due to its cost.

Nevertheless, the overall risk of such an attack is high and immediate action must be taken to mitigate it. We therefore recommend the use of an online anti-DDoS service, such as Cloudflare, for the duration of the elections. When such a service is used, traffic to the public server and the election results API is filtered by the service provider and any DDoS traffic is automatically blocked.

2.9.5 API security (ID: 13)

To mitigate the risk of an attack via the results files parsed by the API, a tool should be used to verify the contents of these files as soon as they are received by the API and before they are loaded in the parser.

The API should be configured to accepts result files coming only from the interior of the network, possibly only from the IP address of the data diode. The IDS should also be configured to drop any packets with POST requests to the API coming from the exterior of the network.

2.9.6 Database security (ID: 3)

We recommend the use of hardening scripts for the database used by the API. Logging should also be activated to monitor any attempts of tampering with the databases content. Remote connections to the database must be disabled and a strong user password used.

To avoid SQL injections, all requests to the API must be sanitized before being forwarded to the database. These requests must be performed through a connection disposing of only SELECT privileges on specified tables.

2.9.7 Training of administrators (ID: 4)

The overall risk of a misconfiguration of the system has been evaluated as low. On the other hand, mitigation can be achieved quite easily by proper training of the system administrators.

We recommend a short briefing of the administrators on the use, the capabilities and the proper configuration of the data diode following the installation of the system.

2.9.8 Software enforcement of password policies (ID: 5)

As noted in section 2.5.2, the IT Security policy of the department imposes format restrictions for passwords used on sensitive systems, but those restrictions are not imposed on software.

We recommend that the data diode software impose these restrictions by rejecting any new password not conforming to them. It should also automatically trigger a password reset on login for passwords older than 30 days.

2.9.9 Protection against timing attacks on login (ID: 5)

An attacker attempting to brute force the login credentials of the web interface could use a timing attack to easily deduce if he has managed to find the correct username (in other words he could be able to focus on the username and password parts separately, which reduces overall security).

We recommend paying special attention to safeguarding the login procedure against this kind of manipulation.

2.9.10 Number of administrators (ID: 4)

Given the importance of the system in the transmission of election results, we recommend that at least two and no more than three administrators have access to the credentials needed to configure the data diode and the API. The same administrators should of course have access to the secured area of the floor.

At least two administrators with access to the data diode should be present on site at all times during the election and until the transmission of the results is concluded.

2.9.11 Fire and power outages (ID: 9,10)

The current fire extinguishing and UPS systems can adequately mitigate these risks in a reasonable scale. We accept the risk of the system being affected by a prolonged or generalized power outage or by a fire that cannot be controlled by the fire extinguishing system.

2.9.12 Accepted risks (ID: 6,7,8,11,12)

Based on the risk scores and the overall analysis, we recommend accepting the following risks:

- **Hardware failure, software bugs**: the system will be thoroughly tested and its stability assessed by an independent contractor prior to the elections. No further action is recommended.

- **Social engineering, accidental exposure of credentials**: the password policies already in place and the security training of the administrators are the most effective ways of mitigating these risks. Likewise, no further action is recommended.

Chapter 3

Architecture and Implementation

This chapter describes the architecture and the implementation of a hardware prototype of the data diode. We detail the services installed on every component of the system and how these components are interconnected. An overview of the architecture can be found in Figure 3.1.

We also briefly discuss our software prototype and the process of moving from a software-only to a hardware implementation.

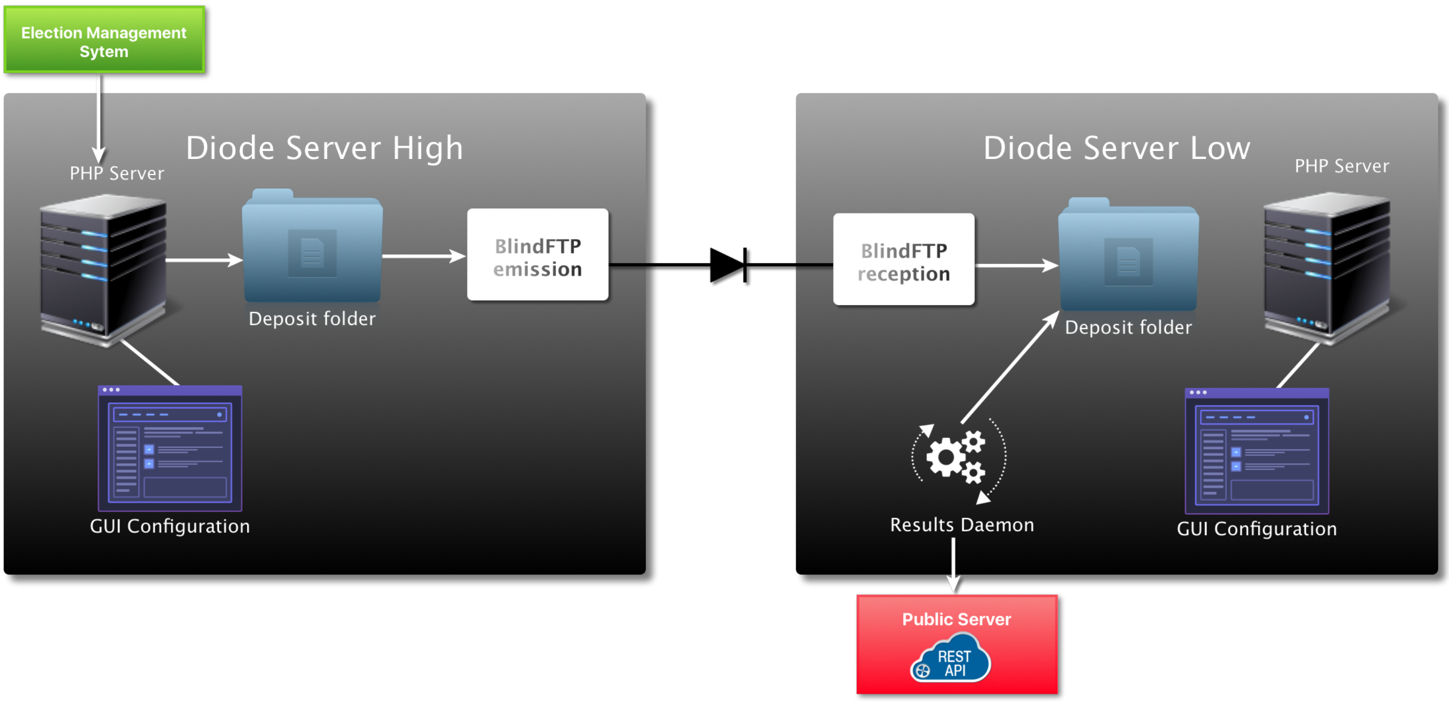


Figure 3.1: System architecture and components

3.1 EMS

As we noted in the CONOPS and the Risk Assessment chapters, the EMS is not in the scope of this project. In order to test the functions of our data diode and for demonstration purposes, we simulated the EMS using a basic Python3 program.

This program simulates a flow of election results. It uses a randomized process to periodically create new results. It then exports the results using the XML format into a number of files corresponding to different regions (see Figure 3.5a). Finally, it performs a series of POST requests to transfer the XML files to the data diode.

3.2 Data Diode

The data diode consists of two servers, a low and a high one. The unidirectional flow of data moves from the high to the low part (H2L). For the prototype, we used two Dell PowerEdge Rack Servers.

3.2.1 High part

The high part of the diode runs Ubuntu Server 16.04.2. The following services and software are present on this server:

- Apache HTTP server (version 2.4.25)
- PHP (version 7)
- BlindFTP (`https://www.decalage.info/python/blindftp`)
- iptables firewall
- xmlsec signature tool (`https://github.com/lsh123/xmlsec`)

First of all, the Apache server serves the administration GUI page. On our hardware prototype, this page is available on port 80 for convenience purposes. In any subsequent iterations, the server should accept connections to the GUI uniquely on port 443 (HTTPS). The administration interface is built using HTML, jQuery and the Bootstrap framework. The back end of the administration GUI runs on PHP. See Figure 3.3 for an example of the interface.

Additionally, the server listens for POST requests on port 80 from the EMS server containing XML files with the election results. The files are signed upon reception using the xmlsec tool and placed in a special "source" folder, whose contents are synchronized upstream (see 3.5b for an example of a signed XML file).

The BlindFTP software manages the synchronization of this special folder. It uses the UDP protocol to forward data to the "destination" folder present on the low part of the diode and has its own system to introduce redundancy and assure the reception of the files upstream.

The iptables firewall is used to block all traffic to ports other than the ports used by the web server (whitelist policy).

3.2.2 Low part

The low part also runs Ubuntu Server 16.04.2 and includes the following services:

- Apache HTTP server (version 2.4.25)
- PHP (version 7)
- BlindFTP (`https://www.decalage.info/python/blindftp`)
- iptables firewall
- results daemon

As for the high part, the Apache server serves a similar administration page on port 80[1].

The results daemon is written in Bash. This daemon constantly checks the contents of the "destination" folder, managed by BlindFTP and where the result XML files become available after transiting from the high to the low part. It detects updates to the results and forwards them to the public server.

The iptables firewall is likewise configured to only allow traffic to server ports.

3.2.3 Network interfaces and fiber-optic link

Each Dell server has two network interfaces. We connected two of them together using the fiber-optic link (we refer to those as "diode" interfaces). This connection is used to transmit data from the high to the low part of the diode. The remaining interfaces are used to connect the two parts of the diode to the their corresponding subnets.

The IP addresses of the interfaces used to communicate with the rest of the network are static but can be changed freely using the web interface. An entry with the MAC address of the low "diode" interface is manually added to the ARP table of the high part of the diode.

Finally, we disconnected the receive and send transceivers from the high and the low part of the diode respectively. Depending on the type of fiber-optic link used, this might trigger the link failure protection mechanism and completely block any data from going out of the high part of the link. A potential solution that does not break the guarantee of physical security is to connect the "orphan" receive transceiver to a "dummy" link that is not connected to anything, but is enough to foul the link failure protection.

3.3 Public server and REST API

As we have already noted above, the public server infrastructure is not part of the system. We implemented an API that has to be installed on the public server. Our implementation also requires the installation of the xmlsec tool. For the purposes of the testing process, we used an Apache server installed on a local machine to simulate the public server. Figure 3.4 provides an example of a site we built using the API to display the election results.

[1]The same remark considering the use of HTTP holds.

The back-end of the API is written in PHP, using the Silex framework (`http://silex.sensiolabs.org`). The results are received by means of POST requests to a special route of the API. Upon reception of the results, the back-end will verifies the signature of the XML files using the xmlsec tool.

The public key necessary for the verification of the signature is obtained from the low part of the data diode. The authenticity of the transmission is verified using an HMAC code. The HMAC is generated using a shared secret, pre-installed on the low part of the diode and the public API.

When the signature of an XML file is successfully verified, the file is parsed by the API and the results are stored in a MySQL database.

3.4 Network architecture

The system operates within a network comprising two subnets. The first subnet corresponds to the secure part of the network, which is not connected to the public Internet. It contains a network switch and the EMS. The high part of the data diode is connected to this subnet.

The low part of the data diode is connected to the second subnet. This subnet contains a router that allows communication high the Internet, as well as the public server. An example diagram of this configuration can be found directly below :

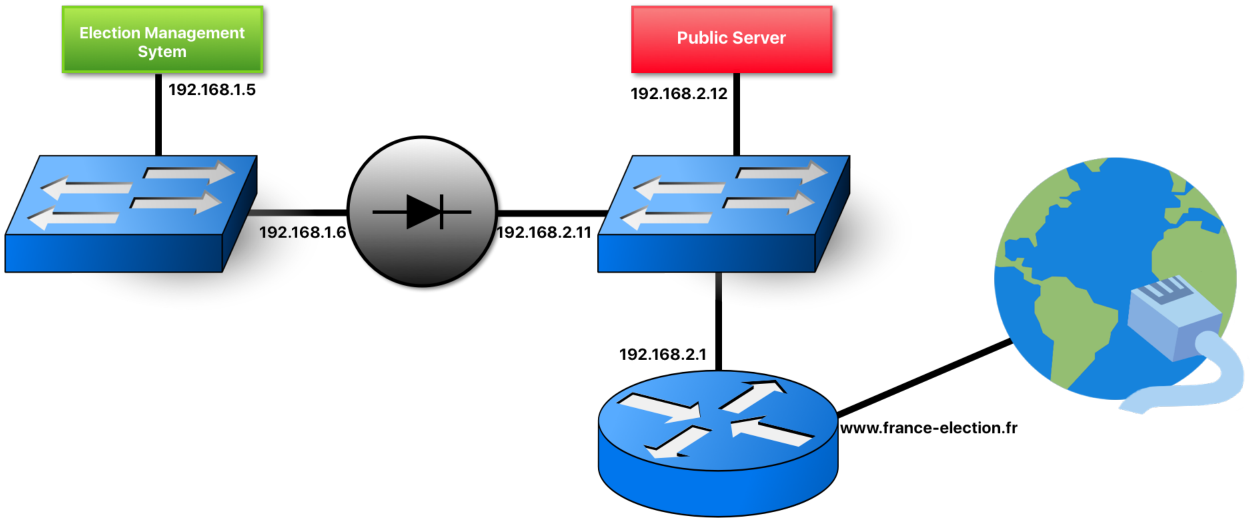


Figure 3.2: Example network configuration. Secure subnet: 192.168.1.0/24, public subnet: 192.168.2.0/24.

3.5 Software prototype

Due to scheduling constraints, the time he had to realize our hardware prototype was limited. Thus, we started the development of the diode with a software prototype, in order to test our concept and our architecture and to develop all the individual parts (web interface, daemons etc.).

In order to streamline the implementation and the testing of the data diode software prototype, we used Vagrant [2] to setup our environment. Vagrant is a tool that allows the creation of a set of virtual machines based on a given script, along with the required network configuration.

In the prototype, the high part and the low parts of the diode are virtual servers running Ubuntu Server 16.04.2. All the software and the services running on those are identical to those described above.

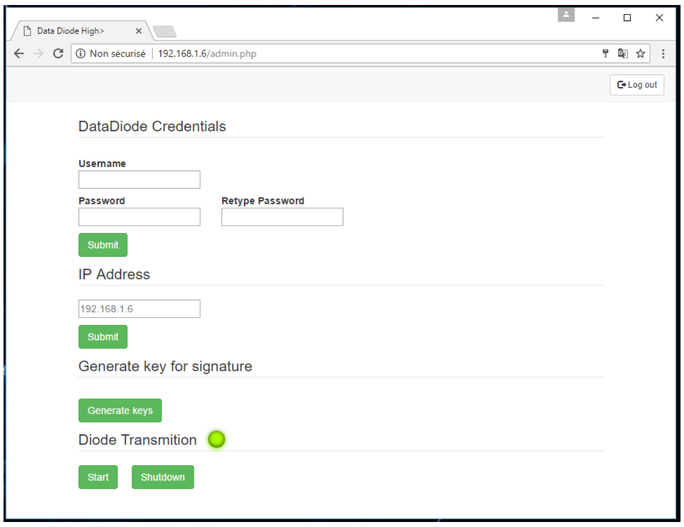
In order to simulate the physical restriction of traffic flow between the diode parts, we used the iptables firewall installed on both sides. Specifically, we configured iptables on the high part of the diode to block all incoming traffic from the IP address of the low part of the data diode and thus simulate the unidirectional data flow.

3.5.1 From software to hardware

During the development of our software prototype, we gradually built an installation script in bash for each part of the diode. We initially used these scripts to configure the virtual machines inside Vagrant. They automatize the installation of all required services and components and perform all the relevant configuration steps (creation of folders, modification of configuration files etc.).

When we moved from software to hardware, we were able to use the same scripts (with some minimal changes) in order to automatically prepare our environments and thus considerably speed up the realization process.

3.6 Illustrations


Data Diode High>

Non sécurisé | 192.168.1.6/admin.php

Log out

DataDiode Credentials

Username

Password | Retype Password

Submit

IP Address

192.168.1.6

Submit

Generate key for signature

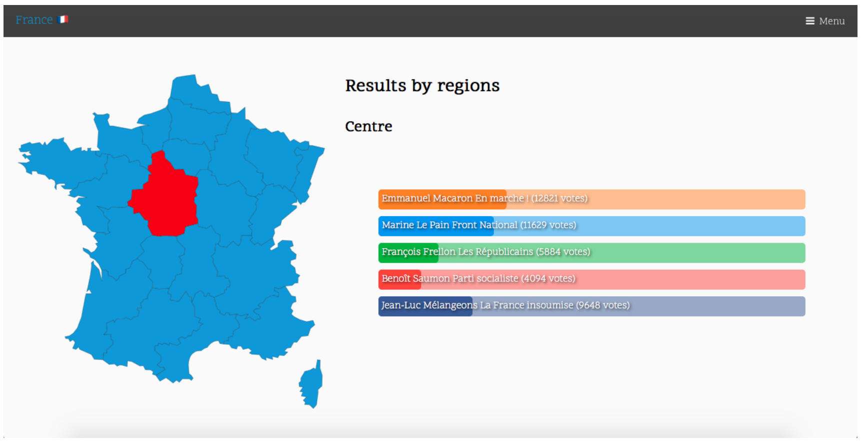
Generate keys

Diode Transmition ●

Start | Shutdown

Figure 3.3: The configuration interface of the high part of the diode. The administrator can update the login credentials, change the IP address of the diode, generate a new pair of keys and start or stop the transmission

[2]https://www.vagrantup.com

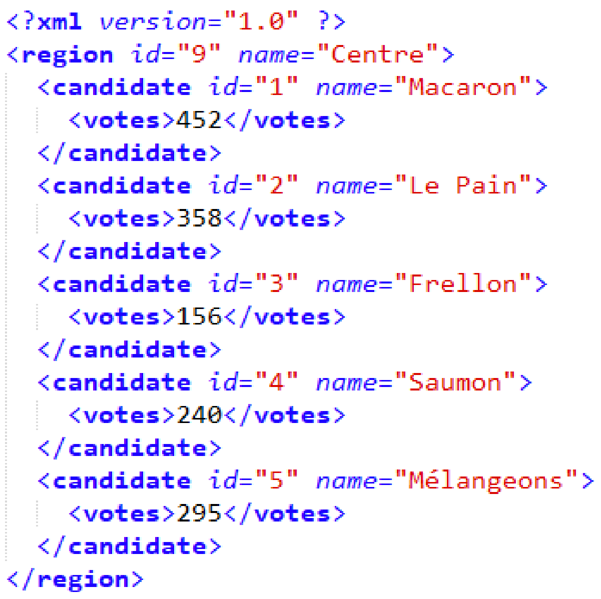


Results by regions

Centre

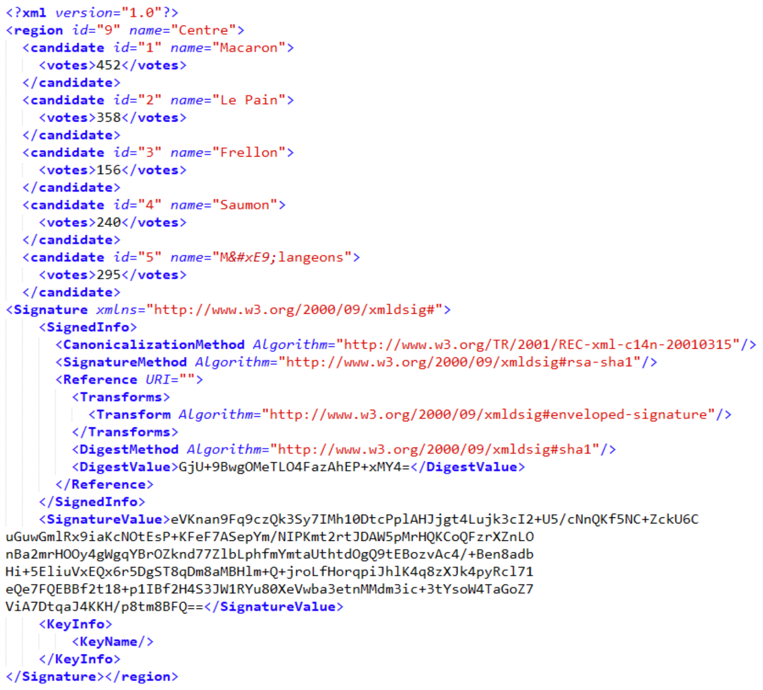
Emmanuel Macaron En marche ! (12821 votes)
Marine Le Pain Front National (11629 votes)
François Frellon Les Républicains (5884 votes)
Benoît Saumon Parti socialiste (4094 votes)
Jean-Luc Mélangeons La France insoumise (9648 votes)

Figure 3.4: An example of the results on a site using the elections API



```xml
<?xml version="1.0" ?>
<region id="9" name="Centre">
  <candidate id="1" name="Macaron">
    <votes>452</votes>
  </candidate>
  <candidate id="2" name="Le Pain">
    <votes>358</votes>
  </candidate>
  <candidate id="3" name="Frellon">
    <votes>156</votes>
  </candidate>
  <candidate id="4" name="Saumon">
    <votes>240</votes>
  </candidate>
  <candidate id="5" name="Mélangeons">
    <votes>295</votes>
  </candidate>
</region>
```

(a) Unsigned results



```xml
<?xml version="1.0"?>
<region id="9" name="Centre">
  <candidate id="1" name="Macaron">
    <votes>452</votes>
  </candidate>
  <candidate id="2" name="Le Pain">
    <votes>358</votes>
  </candidate>
  <candidate id="3" name="Frellon">
    <votes>156</votes>
  </candidate>
  <candidate id="4" name="Saumon">
    <votes>240</votes>
  </candidate>
  <candidate id="5" name="M&#xE9;langeons">
    <votes>295</votes>
  </candidate>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
    <SignedInfo>
      <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
      <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
      <Reference URI="">
        <Transforms>
          <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
        </Transforms>
        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue>GjU+9BwgOMeTLO4FazAhEP+xMY4=</DigestValue>
      </Reference>
    </SignedInfo>
    <SignatureValue>eVKnan9Fq9czQk3Sy7IMh10DtcPplAHJjgt4Lujk3cI2+U5/cNnQKf5NC+ZckU6C
uGuwGmlRx9iaKcNOtEsP+KFeF7ASepYm/NIPKmt2rtJDAW5pMrHQKCoQFzrXZnLO
nBa2mrHOOy4gWgqYBrOZknd77ZlbLphfmYmtaUthtdOgQ9tEBozvAc4/+Ben8adb
Hi+5EliuVxEQx6r5DgST8qDm8aMBHlm+Q+jroLfHorqpiJhlK4q8zXJk4pyRcl71
eQe7FQEBBf2t18+p1IBf2H4S3JW1RYu80XeVwba3etnMMdm3ic+3tYsoW4TaGoZ7
ViA7DtqaJ4KKH/p8tm8BFQ==</SignatureValue>
    <KeyInfo>
        <KeyName/>
    </KeyInfo>
</Signature></region>
```

(b) Signed results

Figure 3.5: Example of unsigned and signed XML results file

Chapter 4

Demonstration plan

4.1 Objectives

The goal of the demonstration is to prove the proper functioning of the system. More specifically:

- It will show the transmission of results from the Election Management System (EMS) to the API hosted on a server by means of the data diode
- It will show how the administrator can interact and manage all parts of the system, such as the data diode configuration GUIs
- It will show how users can interact with the API which publishes the election results

4.2 Materials

Two computers are going to be used during the demonstration. The first one will contain:

- A Virtual Machine (VM) simulating the high part of the data diode
- A VM simulating the low part of the data diode
- A server hosting the API
- A Python3 script simulating the EMS

The second one will contain a web browser showing the website of the election where results will arrive in real time. It will also be used to perform manual requests to the API.

4.3 Step by step walk-through

The demonstration will evolve as follows. The system will be already setup and working. We will:

1. Demonstrate the site with the election results being updated in real time with new results
2. Show a series of requests to the election results API using a web browser
3. Connect to the configuration GUI of the high part of the data diode
4. Show how we can modify the IP address of the data diode. The results will stop being updated, as the EMS sends them to a specific address
5. Restore the IP address of the data diode and show the reestablished flow of results
6. Demonstrate the generation of a new set of public-private keys
7. Show the unsigned contents of the XML files generated by the EMS
8. Show the signed contents of the XML files arriving to the API from the data diode to demonstrate the signature process
9. Simulate a series of XML results files with invalid signatures arriving to the API and show they are not accepted