# Pilot deployment of new voting system

## 1. Introduction

This project builds on the qualitative case study of the course, “The Votes Counting Software”.

You have already written a report—a SECRAM risk analysis—on the migration from OSV2020 (the current votes-counting software and its corresponding IT settings) to DHV (the new centralised software) for votes-counting of Dutch elections. In this final project you will integrate that report with a new quantitative analysis.

The quantitative information comes in the shape of CVEs, which you will use to perform a CVSS Environmental assessment (or close) of an existent IT deployment. Based on an identification of the most severe vulnerabilities and its technical characteristics, you will propose simple actions or changes to mitigate the vulnerabilities, and assess the ensuing modification of risk.

## 2. General scene

The national authorities of the Netherlands opened a call for a pilot, to test a practical deployment of the new votes-counting software project. In response, the mayor of the municipality of Boekelo—a young man pursuing an ambitions political career—has offered “his town” for the test (yes, those were his words). Since nobody else answered the pilot call, the authorities decided to go for it and run the pilot in Boekelo.

This test must be realistic, but pose no serious risk to the general country politics. For that reason, the government decided to use the software for counting the Waterschappen votes[1]. Unfortunately though, the town hall of Boekelo is too small to host the counting machines needed for the elections of the Waterschap Vechtstromen to which it belongs, since that region includes the full provinces of Overijssel and Drenthe. In view of this, the ambitious mayor decided to use the biggest (and only) basketball court in town, to receive the ballots and do the counting there. That court is located in the biggest (and only) gymnasium in town.

### *2.1. Scenario before compliance*

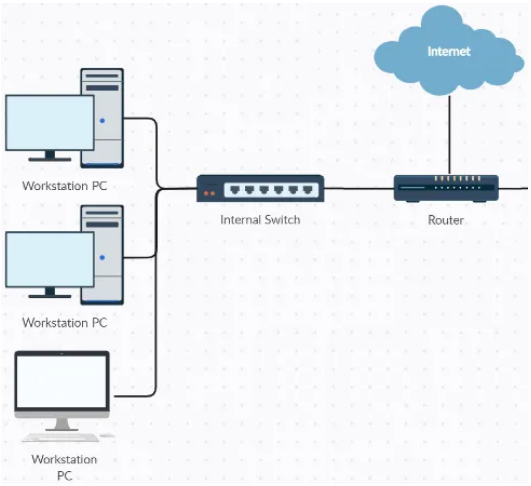
To define the general IT and network architecture of the site, the mayor hired a local SME. Alleging stringent budget limitations, the company decided to locate all votes-counting PCs towards the edges of the basketball court, and connect them to Ethernet outlets that are spaced evenly along the walls. This has the benefit of shortest-path from PCs to pre-existent Internet connections, thus saving money in cables and other equipment. Brilliant!

Following national regulations, the proposed IT deployment was setup and analysed by a 3rd party security company. After recovering from the ensuing mental stroke, the technicians from

1 Some weird Dutch thing about water management—they truly have elections on that, Google it!

this Trento-based company explained that those outlets in the walls were connected to different routers scattered in the gymnasium, and even in other buildings such as the town hall next door. Some Dutch buildings are weird. This resulted in the pentest that is reported in the file `BuildingScan-VotesCounting.xlsx`, which amounts to 204 CVEs of varying severity.

In view of this situation, the team proposed to use only the outlets located at the front of the basketball court—at the expense of a little more wiring, you cheap basterds—which are all connected to one router. This router is located in a secure place in the access lobby on the first floor of the town hall, and via a switch it provides Internet to this part of the Gymnasium. The very simple setup that this produces looks like this:




### *2.2 Compliance check*

This IT setup proposed by the 3rd party company was immediately accepted by the mayor. It reduced the initial preposterous situation to a single gateway and six related CVEs: CVE-2013-2566, CVE-2015-4000, CVE-2016-7406, CVE-2018-6683, CVE-2021-22927, CVE-2022-38652.

From those CVEs, the ones listed for the CIDR 145.187.193.0/24 pertain to the router in the town hall. Those listed for the CIDRs 161.166.204.0/22 and 161.166.200.0/22 are from the PCs (and the votes-counting software) that are located in the basketball court. This information can be found in the accompanying `BuildingScan-VotesCounting.xlsx` file: CIDRs are in the sheet `VLANs`; the results of the scan for the corresponding IPs are in the sheet `Scan`.

## 3. Assignment instructions

Your work for this final project will extend your previous SECRAM report for the votes-counting software. You will deliver that report, updated as per the instructions enumerated below.

The material that you add or change with respect to your previous report must be coloured in blue and italicised *like this*. Awful, I know, but it facilitates keeping track of changes. At your option, you can also underline it, *like this*, or highlight it, *like this*. Even more horrible and clear.

1. Out of the six CVEs listed above, choose four that must be mitigated. Your choice must be justified: select the CVEs that pose the biggest threat to the votes-counting system. Add your chosen CVEs and the justification text to the end of § 2 "Summary of findings".
   *(Hint: a single line can justify all choices)*
   *(Hint 2: this should feel a lot easier than the choices you made at the beginning of the course for this task)*
2. Identify, from your earlier project, the supporting and primary assets that are impacted by each of these four CVEs. By the pigeonhole principle, it is quite possible that two or more CVEs affect the same asset. If, on the other hand, some chosen CVE is unrelated to your identified assets, then (I'm flabbergasted but) you are allowed to introduce a supporting/primary asset for it. Then, per CVE, add a row to the "Threats impact" table:
   a. describe a (realistic plz) threat scenario in which the vulnerability can be exploited;
   b. for the "Vulnerability" column you can just list the CVE text;
   c. indicate which security properties of which primary asset(s) are affected, and note how this is strongly connected to the *Security requirements* of the CVSS Env metrics;
   d. indicate the inherited and reviewed impact values.
3. Likewise, add these CVEs to the "Threats likelihood" table. The values chosen for the likelihood areas must be coherent with the description and CVSS Base metrics of the vulnerability—use the one provided by the NVD. For instance, if a vulnerability requires physical access (AV:P), user interaction (UI:R), and high attack complexity (AC:H), you should indicate a high likelihood of detection and a high requirement of proper skills.
   a. Per CVE add one row to the bottom of that table with this information.
   b. Also, after the table, add one or two lines per CVE to explain your choices.
4. Likewise, add these CVEs to the bottom of the "Risk evaluation" table. NOTE: it is expected that these vulnerabilities have a risk "High" or "Very High". If this did not happen you should revise your assessments.
5. Propose risk treatments for this vulnerabilities, and add them to the corresponding section of the report. But we do it differently than for the previous report:
   a. In the "Pre-controls" column, describe the actions that you take to mitigate the vulnerability. E.g. "buy a new router" or "burn all Windows PCs".
   b. In the "Post-controls" column list the eight *Modified Base Metrics* of the vulnerability that result after applying your pre control. E.g. "`MAV:L, MAC:H, MUI:X, …, MA:L`".
   c. Use the values of the *Modified (Base) Impact Metrics* to assess a value for the "Residual Impact" column. Does it feel a bit more solid than what you had done before?
   d. Use the values of the *Modified (Base) Exploitability Metrics*, and the connection you made to their likelihood areas in the "Threats likelihood" table, to assess a value for the "Residual Likelihood" column. Same insidious comment as above.
   e. Compute the residual risk, submit, and go to enjoy summer holidays.