

Lab 2: Passive Reconnaissance

Total Points: 30

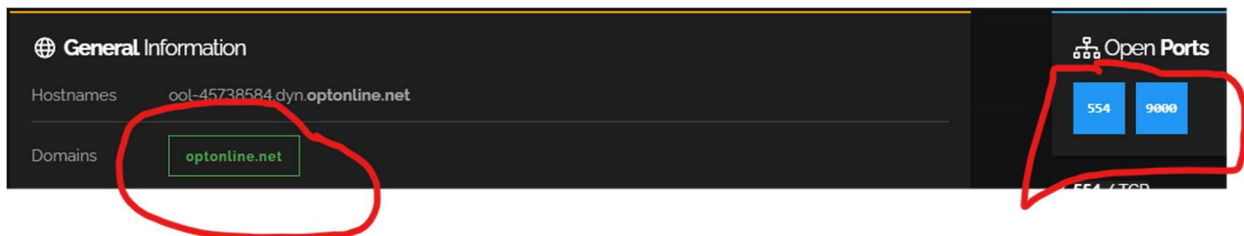
Question 1: Login to Shodan (<https://www.shodan.io/>) using your Gmail account or any other account you have created with the portal. Search **Web Camera** or **Web Cam** in the search bar, and you will be shown a report where a number of accessible web cameras are listed.

- **Task 1:** Find a device where there is at least one open port and the domain name (URL) is displayed. If you find multiple such devices, just choose one arbitrarily. Take a screenshot highlighting the domain name and the open ports. Attach the screenshot in your submission.

4 points

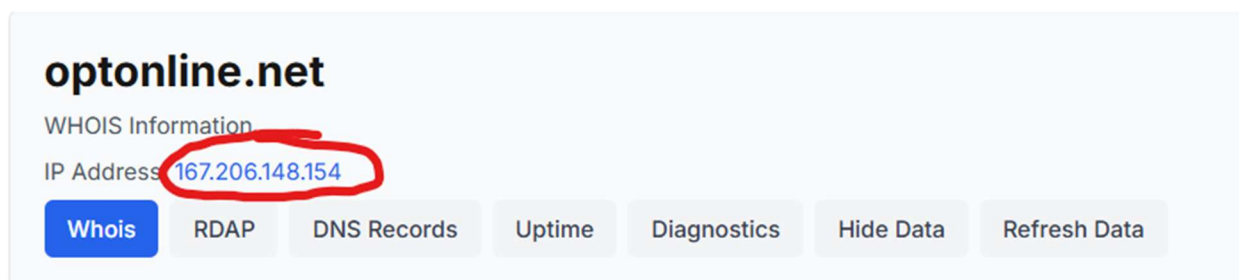
Domain name: optonline.net

Open ports: 554, 9000



- **Task 2:** Using WHOIS (<https://who.is/>) or Netcraft (<https://sitereport.netcraft.com/>), find the IP address of the domain name you found in Task 1. Take a screenshot highlighting the IP address and attach it in your submission. Go through the complete report you retrieved from WHOIS or Netcraft. Do some research online about the vulnerabilities or weakness the device has. Briefly describe all the security weakness or vulnerabilities you found.

6 points



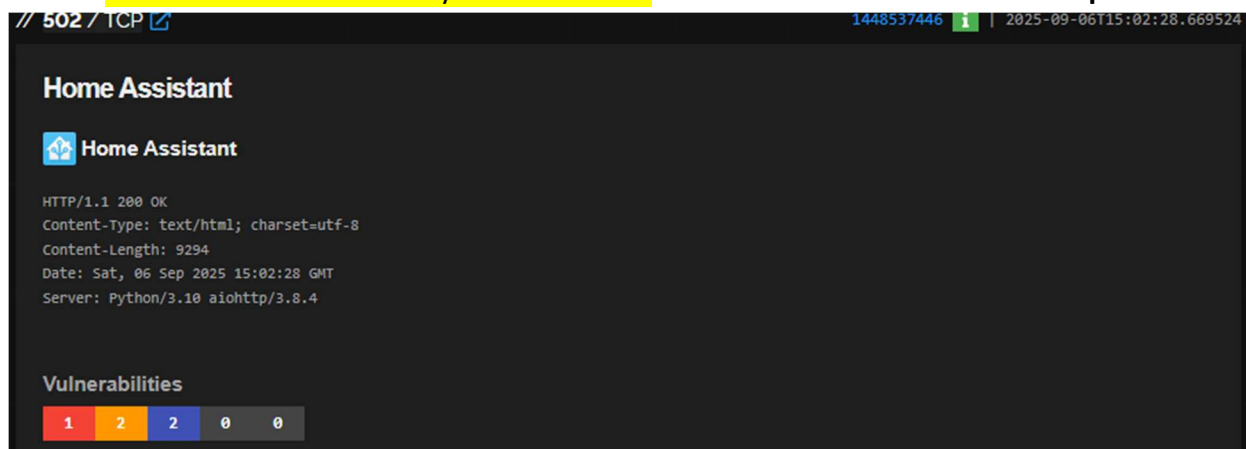
By going through Shodan, I found that the Web version of this device was build 230712. Hikvision as a company has a pdf with the related products that would be running that build in a [pdf on their website](#). This gives a list of devices to look up vulnerabilities on. One such vulnerability is CVE-2023-48121, which is an authentication bypass vulnerability, which will allow remote attackers to get access to information by sending specific messages to the device.

Question 2: Login to Shodan again, but this time search for **port:502**. Select a device that meets the following criteria:

1. There is at least some information in the **device identification** field.
2. There is at least one **CVE** listed in the **Vulnerabilities** section.

- **Task 1:** Capture some screenshots showing the device id, open ports, and the CVE lists. Attach the screenshots in your submission.

5 points



Open Ports

2	11	13	15	17	20	21	22	23	25	26	37	43	49
51	53	65	66	70	79	80	81	82	83	84	88	91	95
99	100	102	104	110	111	113	119	121	122	143	154	161	175
179	180	192	195	199	221	225	264	285	311	343	389	427	440
443	444	448	451	461	462	465	502	503	513	515	541	548	554
555	556	587	591	593	602	631	636	666	675	689	700	789	805
806	809	830	831	833	873	885	888	902	943	990	992	993	994
995	1002	1023	1024	1025	1050	1080	1099	1111	1153	1177	1180	1181	1195
1198	1200	1234	1235	1283	1290	1291	1311	1337	1365	1366	1370	1400	1414
1433	1443	1446	1452	1453	1457	1459	1460	1471	1515	1521	1604	1660	1701
1723	1741	1800	1801	1820	1830	1883	1911	1925	1926	1935	1947	1952	1957
1959	1962	1965	1967	1969	1971	1974	1981	1986	2000	2002	2003	2008	2020
2021	2050	2052	2056	2057	2058	2059	2063	2067	2069	2070	2081	2082	2086
2087	2100	2108	2111	2130	2134	2150	2154	2156	2181	2210	2211	2222	2224
2226	2250	2323	2332	2344	2345	2375	2376	2379	2382	2404	2423	2435	2443
2455	2480	2525	2549	2553	2556	2560	2561	2572	2598	2628	2761	2762	2850

📅 2023 (1)

CVE-2023-36632

7.5 The legacy `email.utils.parseaddr` function in Python through 3.11.4 allows attackers to trigger "RecursionError: maximum recursion depth exceeded while calling a Python object" via a crafted argument. This argument is plausibly an untrusted value from an application's input data that was supposed to contain a name and an e-mail address. NOTE: `email.utils.parseaddr` is categorized as a Legacy API in the documentation of the Python email package. Applications should instead use the `email.parser.BytesParser` or `email.parser.Parser` class. NOTE: the vendor's perspective is that this is neither a vulnerability nor a bug. The email package is intended to have size limits and to throw an exception when limits are exceeded; they were exceeded by the example demonstration code.

📅 2021 (1)

CVE-2021-32052

6.1 In Django 2.2 before 2.2.22, 3.1 before 3.1.10, and 3.2 before 3.2.2 (with Python 3.9.5+), `URLValidator` does not prohibit newlines and tabs (unless the `URLField` form field is used). If an application uses values with newlines in an HTTP response, header injection can occur. Django itself is unaffected because `HttpResponse` prohibits newlines in HTTP headers.

📅 2020 (1)

CVE-2020-29396

9.9 A sandboxing issue in Odoo Community 11.0 through 13.0 and Odoo Enterprise 11.0 through 13.0, when running with Python 3.6 or later, allows remote authenticated users to execute arbitrary code, leading to privilege escalation.

📅 2009 (2)

CVE-2009-3720

6.0 The `updatePosition` function in `lib/xmlltok_impl.c` in `libexpat` in Expat 2.0.1, as used in Python, PyXML, w3c-libwww, and other software, allows context-dependent attackers to cause a denial of service (application crash) via an XML document with crafted UTF-8 sequences that trigger a buffer over-read, a different vulnerability than CVE-2009-2625.

CVE-2009-2940

7.5 The `pygresql` module 3.8.1 and 4.0 for Python does not properly support the `PQescapeStringConn` function, which might allow remote attackers to leverage escaping issues involving multibyte character encodings.

Task 2: Do some research about the device you chose and describe the device type and found vulnerabilities in a paragraph. Try to keep the paragraph limited into 5-10

sentences.

5 points

Home assistant is an open source IOT home automation app. It can be used to manage and configure various devices that it is installed on, like a raspberry PI. A very severe vulnerability listed here is CVE-2020-29396. If this device's software is not up to date, this vulnerability can be exploited to remotely execute arbitrary code, leading to privilege escalation. This is bad on an IoT device because an attacker may be able to then increase their presence in your home network and compromise other devices in your network.

- **Task 3:** Select a CVE from the CVE list shown in the *Vulnerabilities* section and search for that CVE in <https://cve.mitre.org/>. Identify the attack/vulnerability described in the CVE.

Old Dominion University
CYSE 450: Ethical Hacking and Penetration Testing

Go to <https://attack.mitre.org/matrices/enterprise/network/> and find the attack from the matrix. If the attack is not listed there, try to search in other attack matrices given in the MITRE ATT&CK website. Once you find the attack listed as a **technique**, try to find out one relevant **detection** and one **mitigation** methods. Take screenshots showing the detection id and the mitigation id. Attach your screenshots in your submission and briefly summarize the selected detection and mitigation methods. **10 points**

This attack is perpetrated through a malicious user with an internal user account on an Odo database crafting special code expressions (GitHub) So I chose the attack technique T1078, Valid Accounts. I chose mitigation M1018 because if you are monitoring what accounts are active and what they are doing, it will be harder for an attacker to use an account to execute malicious code. For a detection technique, I chose DS0028. By tracking who is logged on, what processes they are running, and looking for other suspicious activities, you have a better chance of knowing if malicious code is being run.

M1018	User Account Management		Regularly audit user accounts for activity and deactivate or remove any that are no longer needed.
DS0028	Logon Session	Logon Session Creation	Monitor for newly constructed logon behavior that may obtain and abuse credentials of existing accounts as a means of gaining Initial Access, Persistence, Privilege Escalation, or Defense Evasion. Correlate other security systems with login information (e.g., a user has an active login session but has not entered the building or does not have VPN access). <pre>_sourcetype="WinEventLog:Security" EventCode=4624 stats count by _time, user, src_ip, dest_ip, LogonType where LogonType IN ("2", "10") // Interactive or RDP logon eval is_suspicious=if(src_ip!="expected_ip", "True", "False") where is_suspicious="True" table _time, user, src_ip, dest_ip, LogonType</pre>
		Logon Session Metadata	Look for suspicious account behavior across systems that share accounts, either user, admin, or service accounts. Examples: one account logged into multiple systems simultaneously; multiple accounts logged into the same machine simultaneously; accounts logged in at odd times or outside of business hours. Activity may be from interactive login sessions or process ownership from accounts being used to execute binaries on a remote system as a particular account.