

OMDM

Contexto

O projeto analisado consiste em uma experiência desenvolvida por membros da comunidade do jogo DayZ, mantida de forma independente pela administração da iniciativa.

Superfícies de Ataque

Foram identificadas duas superfícies de ataque principais: o código de uma modificação do jogo desenvolvida por um terceiro, selecionado pela administração da experiência, e uma aplicação web de suporte desenvolvida e mantida pela própria administração.

Aplicação Web

A aplicação web disponível em [O Mundo dos Mortos](#) segue o modelo SPA (Single Page Application), utilizando Dart no front-end, com backend baseado em n8n e Supabase. Essa aplicação é utilizada para gerenciar funcionalidades de apoio à comunidade, incluindo doações voluntárias, concessão de benefícios associados a planos VIP, registro de clãs, personalização de uniformes de clã e abertura de tickets para comunicação com a moderação do servidor.

Falhas:

Foram identificadas 5 falhas de segurança, sendo elas: Business Logic Flaw, Misconfigured RLS, IDOR (Insecure Direct Object Reference), No Rate Limiting e Mass Data Enumeration.

1. Business Logic Flaw:

- a. Descrição: Foi identificada uma falha de lógica de negócio no fluxo de criação de pedidos, na qual o backend aceita diretamente os valores de **preço** (`valor`) e **quantidade de meses** (`quantidade_mes`) enviados pelo cliente, sem realizar o recálculo ou validação desses parâmetros no servidor com base no produto/serviço solicitado. Como consequência, um usuário pode manipular os valores enviados na requisição e submeter

pedidos com preços inferiores aos esperados, burlando as regras de cobrança definidas pela aplicação.

```
{  
    "valor": 3500,  
    "playerId": 1094,  
    "cid": "a5582013-e6fd-4a0a-b6b5-6f202c73f11d", //uid  
    id aleatório  
    "servidorId": 1,  
    "quantidade_mes": 1  
}
```

- b. Impacto: Bypass completo da lógica de precificação, possibilidade de obtenção de benefícios (ex.: planos VIP) por valores inferiores ao estabelecido, prejuízo financeiro direto à iniciativa.

2. No rate Limiting:

- a. Descrição: Foi identificada a ausência de mecanismos de rate limiting em endpoints sensíveis da aplicação, permitindo que um cliente realize um número ilimitado de requisições em um curto intervalo de tempo, sem qualquer tipo de controle, bloqueio ou degradação de serviço.
- b. Impacto: Embora não represente uma falha explorável isoladamente, a ausência de rate limiting permite a automação de requisições e atua como amplificador de impacto em cadeias de exploração.

3. Misconfigured RLS:

- a. Descrição: Foi identificada uma configuração inadequada das políticas de **Row Level Security (RLS)** no Supabase, na qual as regras aplicadas não restringem corretamente **operações de leitura e escrita** com base na identidade do usuário autenticado.
- b. Impacto: De forma isolada, a misconfiguração de RLS representa uma **falha estrutural de controle de acesso**, possibilitando ações não autorizadas sobre dados de terceiros, incluindo modificação e potencial corrupção de informações. Além disso, essa falha atua como **pré-requisito em cadeias de exploração**.

4. IDOR (Insecure Direct Object Reference):

a. Descrição: A aplicação expõe endpoints que permitem a manipulação direta de identificadores de objetos por meio de parâmetros fornecidos pelo cliente. A partir da observação e coleta das requisições realizadas pela própria aplicação, torna-se possível identificar os campos utilizados como referência a registros internos do banco de dados, como identificadores de usuários. Em função da ausência de validação adequada de **ownership** no backend, um usuário autenticado pode modificar os valores enviados nas requisições, utilizando identificadores pertencentes a outros usuários. Essa manipulação permite a **filtragem, seleção e acesso arbitrário a dados de terceiros**, caracterizando uma falha de *Insecure Direct Object Reference (IDOR)*. Essa exploração é viabilizada pela fragilidade existente nas políticas de Row Level Security, que não impõem restrições efetivas no nível do banco de dados para impedir o uso de identificadores não autorizados. A exploração da falha pode ocorrer por **duas abordagens distintas**:

- i. **Manipulação direta de parâmetros de requisição**, na qual o atacante altera explicitamente os identificadores enviados nos filtros da API REST, conforme exemplificado abaixo;

```
Request URL: https://.../rest/v1/player?...
Request Method: GET
Apikey: <supabase_api_key>
Authorization: Bearer <user_jwt>
```

```
select=*
user_id=eq.e3b9f57c-b46e-4bdc-b8f3-437ecc5d117f
```

- ii. **Manipulação indireta via estado client-side**, na qual a aplicação utiliza identificadores armazenados no **localStorage** para compor automaticamente requisições ao backend. Ao modificar esses valores no ambiente do cliente, a aplicação passa a enviar requisições contendo identificadores pertencentes a outros usuários, sem que seja necessário interceptar ou alterar manualmente a requisição HTTP.

```
{  
    "access_token": "...",  
    "token_type": "bearer",  
    "expires_in": 3600,  
    "expires_at": 1770600711,  
    "refresh_token": "...",  
    "user": {  
        "id": "id adulterado",  
        "aud": "authenticated",  
        "role": "authenticated",  
        "email": "...",  
        "email_confirmed_at": "2026-02-09T00:25:  
47.696813Z",  
        "phone": "",  
        "confirmed_at": "2026-02-09T00:25:47.696  
813Z",  
        "last_sign_in_at": "2026-02-09T00:31:51.  
92065772Z",  
        "app_metadata": {  
            "provider": "email",  
            "providers": [  
                "email"  
            ]  
        },  
        "user_metadata": {  
            "email": "...",  
            "email_verified": true,  
            "phone_verified": false,  
            "sub": "id adulterado"  
        },  
        "identities": [  
            {  
                "identity_id": "...",  
                "id": "id adulterado",  
            }  
        ]  
    }  
}
```

```
"user_id": "id adulterado",
```

```
...
```

- b. Impacto: A exploração de IDOR possibilita o acesso não autorizado a informações sensíveis de terceiros, comprometendo a confidencialidade e integridade dos dados da aplicação. Essa falha depende diretamente da misconfiguração de RLS para sua exploração prática e atua como etapa intermediária em cadeias de exploração mais complexas, como a enumeração massiva de dados.

5. Mass Data Enumeration:

- a. Descrição: Foi identificada a possibilidade de **enumeração massiva de dados** por meio da automação de requisições explorando, de forma combinada, a falha de **IDOR** e a **ausência de mecanismos de rate limiting** na aplicação. A partir da manipulação de identificadores em requisições válidas e da inexistência de controles de limitação de requisições, torna-se viável iterar sistematicamente sobre registros pertencentes a múltiplos usuários. O processo permite a coleta contínua de dados sem bloqueios, alertas ou degradação do serviço. Durante a exploração, os dados obtidos são armazenados de forma estruturada, sendo que cada jogador identificado é representado por um objeto JSON individual contendo as informações extraídas da aplicação.

```
{
    "id": "...",
    "created_at": "...",
    "user_id": "...",
    "steam_id": "...",
    "discord_nick": "...",
    "json_personalizado": { ... }
}
```

- b. Impacto: A enumeração massiva de dados resulta na **exposição sistemática de informações de múltiplos usuários**, comprometendo a confidencialidade dos dados armazenados pela aplicação. Diferentemente de acessos pontuais, essa exploração permite a coleta em larga escala,

ampliando significativamente o impacto das falhas subjacentes. Essa condição representa o **estágio final da cadeia de exploração**, viabilizada pela combinação das seguintes fragilidades: Misconfigured RLS, IDOR e Ausência de Rate Limiting. O impacto potencial inclui violação de privacidade, exposição de dados sensíveis e risco reputacional para a iniciativa.

Observação de Segurança:

Foi identificada uma observação de segurança sendo ela: Persistência indefinida e exposição ampliada de dados sensíveis no client-side.

1. Persistência indefinida e exposição ampliada de dados sensíveis no client-side:
 - a. Descrição: Foi observado que o refresh token permanece válido mesmo após a renovação da sessão, podendo ser reutilizado continuamente para a emissão de novos access tokens, sem invalidação ou rotação efetiva do token previamente emitido. Esse comportamento permite a manutenção prolongada de sessões autenticadas, desde que o refresh token permaneça acessível no ambiente do cliente. Adicionalmente, verificou-se que o localStorage armazena não apenas os tokens de autenticação, mas também metadados sensíveis de identidade do usuário, incluindo endereço de e-mail e campo de número de telefone, ambos acessíveis diretamente via JavaScript. O localStorage não oferece qualquer mecanismo de isolamento ou proteção contra leitura por scripts executados no contexto da aplicação. Essa combinação amplia significativamente o potencial de exposição de dados sensíveis em cenários de comprometimento do ambiente client-side, permitindo não apenas a persistência de sessões autenticadas por períodos prolongados, mas também o acesso direto a informações pessoais dos usuários, que podem ser utilizadas para fins de correlação, engenharia social ou abuso de identidade.
 - b. Impacto: Embora essa condição não represente uma falha explorável de forma isolada, ela aumenta substancialmente o impacto potencial de incidentes client-side, possibilitando a exposição prolongada de sessões e de dados pessoais armazenados no navegador. A persistência do refresh

token e a disponibilidade de informações sensíveis no localStorage elevam o risco de comprometimento contínuo da confidencialidade dos usuários, dificultando a contenção e a mitigação de danos em caso de acesso indevido ao ambiente do cliente.

Modificação do Jogo (Mod)

A modificação do jogo analisada, disponível em [TerjeBruoygard/TerjeMods](#), segue o modelo de desenvolvimento imposto pela Bohemia Interactive e utiliza a linguagem Enforce Script, disponibilizada pela mantenedora do motor do jogo. O mod tem como objetivo permitir que membros de um mesmo grupo visualizem uns aos outros na interface do jogo e no mapa, por meio da exibição de marcadores visuais.

Contexto e Funcionamento

A modificação analisada é o **Terje Party Mod** para o jogo DayZ, cuja finalidade é permitir a formação de grupos (*partys*) entre jogadores dentro do ambiente do jogo. O mecanismo de funcionamento observado segue o seguinte fluxo:

- Jogador **A** interage (tecla **F**) com o Jogador **B**, enviando um convite de party;
- Jogador **B** interage (tecla **F**) com o Jogador **A**, aceitando o convite;
- A party é criada e os jogadores passam a se visualizar mutuamente por meio de marcadores na interface e no mapa.

Observação de Comportamento Anômalo

Durante testes empíricos, foi identificado um comportamento inconsistente após a **formatação completa do computador**, com remoção de todas as pastas e dados locais. Na situação observada:

- O jogador analisado **não visualizava mais a party**, como se não estivesse em grupo;
- Um outro jogador ainda **visualizava o primeiro jogador como membro da party**;
- Não houve qualquer ação explícita de remoção de grupo por parte do servidor.

Essa assimetria levou à hipótese inicial de que **as informações de party não são integralmente armazenadas ou validadas no servidor**, mas dependem de dados persistidos localmente no cliente.

Investigação Técnica

Monitoramento de Arquivos

Foi realizado o monitoramento do processo do DayZ durante a criação de uma party, utilizando ferramentas de observação de sistema (Microsoft Task Manager / Monitor de Recursos / Process Explorer), com o objetivo de identificar acessos a arquivos locais relevantes.

Como resultado, foi identificado o acesso recorrente ao arquivo:

`TerjeParty.dat`

Esse arquivo era aberto e modificado no momento da criação ou atualização de uma party.

Análise do Arquivo

O arquivo `TerjeParty.dat` foi analisado por meio de um **editor hexadecimal**, sendo observada uma estrutura binária consistente após a formação de uma party, indicando armazenamento de dados de configuração local.

Estrutura do Arquivo

O formato do arquivo segue um layout determinístico:

- **Cabeçalho inicial (4 bytes, int32, little-endian):**
 - Representa a quantidade total de GUIDs armazenados no arquivo.
- **Entradas sequenciais por GUID:**
 - 4 bytes (int32, little-endian) indicando o tamanho **N** da string do GUID;
 - **N bytes** contendo o GUID em formato **UTF-8**, sem terminador adicional.

Cada GUID representa um identificador de jogador e é armazenado como uma **string Base64 (URL-safe)**, e não como um valor binário fixo.

O comportamento de escrita observado consiste na simples **adição de novas entradas ao final do arquivo**, sendo necessário apenas atualizar o contador inicial

de GUIDs para manter a consistência do formato.

Propriedades Observáveis

- O layout do arquivo é **simples, sequencial e determinístico**, facilitando o parsing manual;
- A estrutura baseada em `[contagem → tamanho → string]` permite **inserções e modificações diretas**, desde que o cabeçalho seja ajustado;
- Os dados são armazenados localmente em formato legível (Base64), **sem criptografia ou validação de integridade** embutida no próprio arquivo.

Essas características explicam tanto o desaparecimento de partys após a exclusão de dados locais quanto a possibilidade de recriação manual das mesmas.

Exploração da Estrutura

Durante a análise, foi possível identificar dentro do `TerjeParty.dat`:

- O número total de IDs armazenados;
- A lista completa de GUIDs correspondentes aos membros da party.

Foi identificado que o GUID utilizado pelo DayZ é derivado do **SteamID** do jogador por meio do seguinte processo:

`SteamID → SHA-256 → Base64 (URL-safe) → GUID`

Exemplo observado:

- SteamID: `76561198012345678`
- SHA-256: `c991f9c7f37b12d7b3deddeb607ab338af88f4cbd253171c02cce3b...`
- GUID (Base64 URL-safe):
`yZH5x_N7Etez3t3rYHqzOK-I9MvSUxccAszjtPnnRlc`

Exploit / Prova de Conceito

A partir da compreensão completa da estrutura do arquivo `TerjeParty.dat`, foi possível:

- Escrever manualmente entradas de GUID no arquivo;

- Criar **partys customizadas localmente** sem interação legítima no jogo;
- Inserir jogadores em uma party **apenas com o conhecimento de seus SteamIDs**.

Após a modificação manual do arquivo, o jogo reconhecia os jogadores inseridos como membros válidos da party, demonstrando que **não há validação robusta no lado do servidor** quanto à composição do grupo.

A prova de conceito e os experimentos realizados estão documentados em:

<https://github.com/PSalleSDev/TerjeExp>

Nota do Redator

Este documento foi elaborado com **caráter estritamente técnico, acadêmico e imparcial**, tendo como objetivo exclusivo a **análise e documentação de comportamentos observáveis** nos sistemas que compõem a experiência avaliada. A produção deste material **não possui finalidade lucrativa**, nem visa a exposição indevida, descredibilização ou prejuízo à iniciativa ou a seus mantenedores.

A motivação central deste trabalho é **contribuir para o fortalecimento do projeto**, de modo que os recursos obtidos por meio de **doações voluntárias da comunidade** possam ser aplicados de forma mais eficaz, sustentando a evolução contínua da experiência e a manutenção de um ambiente mais **seguro, confiável e estável** para todos os participantes.

A divulgação das observações e análises aqui apresentadas ocorre no contexto de **compartilhamento de conhecimento técnico**, com a intenção de auxiliar a administração da experiência na identificação de pontos de melhoria, mitigação de riscos e aprimoramento das práticas adotadas. Parte-se do princípio de que a transparência técnica e o diálogo construtivo são ferramentas fundamentais para a prosperidade de projetos comunitários e colaborativos.

Este documento deve ser interpretado como uma **contribuição positiva e colaborativa**, orientada à promoção de boas práticas de segurança e à entrega de um entretenimento de qualidade à comunidade, pautado na **confiança mútua** entre administradores e jogadores.