Auditoría Smart Contracts
B2M Token

Versión	1.0
Auditor	David Ortega (Dekalabs)
Fecha	18/08/2021

1. Dekalabs

En Dekalabs somos Ingenieros de Software especializados en desarrollo de Smart Contracts y auditorías. Entre los miembros del equipo, tenemos dilatada experiencia en el desarrollo de aplicaciones y de Smart Contracts desde 2016 en las siguientes plataformas:

- Ethereum/Quorum
- Stellar
- Hyperledger Fabric
- Hyperledger Besu

Hemos desarrollado proyectos pioneros en tecnología blockchain como son climatetrade.com o stadioplus.com

Además tenemos miembros del equipo que son profesores de Blockchain y Smart Contracts en diversas escuelas de negocios.

2. Advertencia Legal

El propósito de la auditoría no es asegurar que los Smart Contracts estén libres de errores y que no se puede ejecutar de forma satisfactoria ningún tipo de ataque contra ellos una vez desplegados. El objetivo es más bien realizar un análisis exhaustivo sobre el código para intentar encontrar vulnerabilidades o hacer más seguro el código para minimizar el riesgo de fallos de seguridad en el proyecto.
Por lo tanto, **la información que se muestra en esta auditoría es para análisis general, mejora del código y disminución del riesgo de vulnerabilidades, pero no tiene el objetivo de proveer seguridad legal a ningún individuo ni entidad ante la ejecución de los Smart Contracts**.

3. Introducción funcional

Este documento contiene un análisis de la seguridad y la calidad del código de los Smart Contracts sobre el token B2M para el ecosistema Bit2Me.
Adicionalmente a los Smart Contracts, Bit2Me ha desarrollado una serie de servicios en su propia plataforma para gestionar todas las funcionalidades de alto nivel requeridas. En este punto de introducción, vamos a realizar un repaso funcional a lo que se requiere de los tokens, así como la capa de aplicación que hay por encima. Hay que dejar claro, que la presente auditoría de seguridad sólo corresponde a los Smart Contracts, no a los servicios anexos desarrollados por Bit2Me, es decir, vamos a analizar todo el código que estará on-chain.

El análisis realizado se ha basado en la información ofrecida por Bit2Me a través de su página de información del token: https://bit2me.com/es/token

3.1. Token

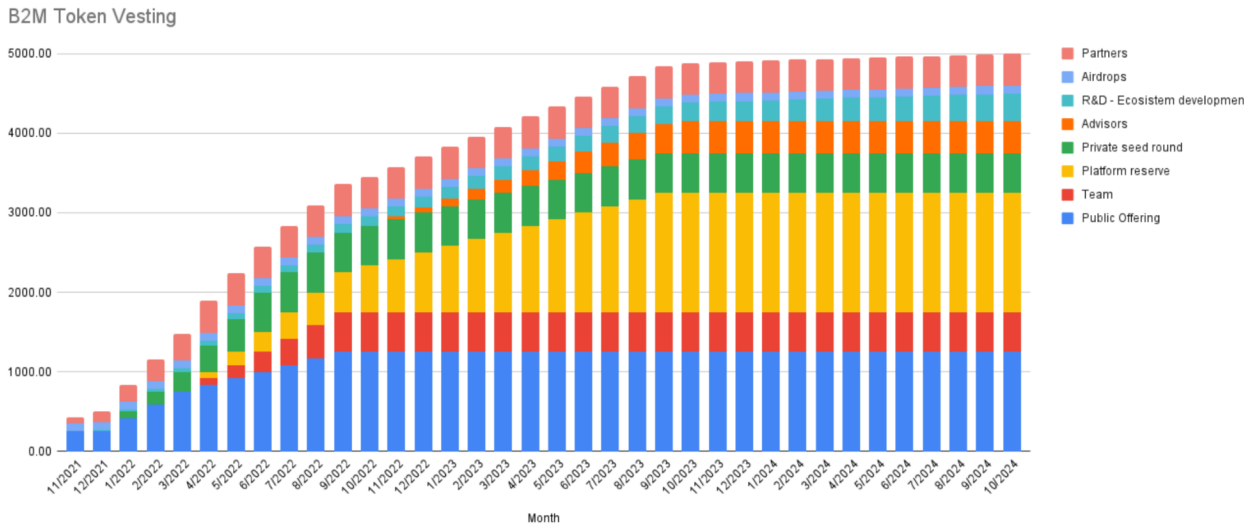
Como podemos observar en el whitepaper publicado, se realiza una emisión de tokens con las siguientes características:

- Nombre completo: Bit2Me Token
- Símbolo: B2M
- Suministro máximo: 5,000,000,000
- Red Blockchain: Ethereum
- Estándar: ERC20

3.2. Emisión y vesting

En el documento publicado se especifican una serie de períodos de bloqueo a los tokens emitidos según las diferentes fases de venta. Estos períodos de bloqueo los gestiona Bit2Me de forma interna en su plataforma y no se espera verlos reflejados en los contratos.

Asimismo, el sistema emitirá inicialmente el total del supply a una dirección de “issuance” de la plataforma Bit2Me, desde donde se realizará la gestión de estos bloqueos, así como la asignación de cantidades.



3.3. Burning program

La idea es reducir el suministro total de Tokens B2M en el mercado. Esto tiene una relación directa con el total de acciones de respaldo/colateral por cada token B2M. La empresa podrá comprar tokens ya creados B2M y destinarlos a una billetera 0x (a través de una compra realizada por la empresa) de la que no podrán salir. Así, en el momento de la emisión del Token B2M, se emitirán tantos como haya de “Total Supply Available”.

Como hemos visto, esta quema será directamente una variable en el smart contract para variar el supply del token mediante la creación del “Total Supply Available”.

Se establece un máximo de un 1% diario de quema de tokens, de manera que no se pueda modificar sustancialmente el supply de tokens ‘in float’.

4. Proceso de auditoría

Para realizar el análisis de seguridad y calidad del código, realizaremos una serie de pasos para ir validando puntos sobre la base del código, en concreto la secuencia sería la Siguiente:

- Comprobar la integridad de los ficheros de código: Esto es importante para tener un punto de partida de auditoría claro y consensuado.
- Realizar tests unitarios que validen la parte funcional de lo que se pide en los Smart Contracts.
- Desplegar los Smart Contracts en local (ganache) así como en una red de test y realizar diferentes ataques para corroborar que no existen el código algunas vulnerabilidades estándar conocidas (SWC)
- Analizar también la calidad del código (DRY, complejidad ciclomática, ...)
- Conclusiones finales, cambios sugeridos.
- Entregables: código con tests (archivo .zip) con integridad (SHA1) y documento de auditoría.

5. Integridad de los contratos

```
8255b832c12202390efbca9627abb5ac47060f03  Token.sol
8bb18db88c1c70bf7b20a83ef75ee99b04782e5f  TokenManager.sol
```

Se incluye la huella SHA1 de los ficheros analizados para validar la integridad de los mismos. De esta manera, si hubiera algún cambio en éstos, la huella se modificaría y el resultado de esta auditoría quedaría invalidado.

6. Tests funcionales

El proyecto incluía ya los tests funcionales necesarios para comprobar la funcionalidad de los contratos. En concreto hay que matizar que el sistema a nivel de blockchain no tiene todas las restricciones que se enumeran en el white paper, ya que muchas de ellas como el vesting o la propia distribución se gestionará de forma centralizada desde las wallets de la plataforma Bit2Me.

En este caso, lo que sí que se ha podido comprobar han sido la gestión del max supply así como del burning program, en el que a priori existen limitaciones como el no poder quemar más de un 1% diario del max supply.

```
tests/test_token.py::test_wei_sent_creation PASSED                    [  8%]
tests/test_token.py::test_burn_fails_no_owner PASSED                  [ 16%]
tests/test_token.py::test_burn PASSED                                 [ 25%]
tests/test_token.py::test_owner_creation PASSED                       [ 33%]
tests/test_token.py::test_change_owner PASSED                         [ 41%]
tests/test_token_manager.py::test_burn_fails_no_owner PASSED          [ 50%]
tests/test_token_manager.py::test_burn PASSED                         [ 58%]
tests/test_token_manager.py::test_burn_two_times_one_day_fails PASSED [ 66%]
tests/test_token_manager.py::test_burn_more_than_1_percent_fails PASSED [ 75%]
tests/test_token_manager.py::test_burn_1_percent PASSED               [ 83%]
tests/test_token_manager.py::test_owner_creation PASSED               [ 91%]
tests/test_token_manager.py::test_change_owner PASSED                 [100%]
```

Se incluye la ejecución de los tests, así como el código de los mismos, donde se puede comprobar la ejecución de todos los tests donde se han comprobado los supuestos anteriores.

7. Tests de seguridad

En esta parte vamos a detallar las potenciales vulnerabilidades encontradas en el código así como las recomendaciones para hacer el código más robusto.

Éstas vulnerabilidades/problemas, se basan en el Smart Contract Weakness Classification (SWC), que es una lista consensuada por la comunidad y por expertos de seguridad de las vulnerabilidades conocidas en diferentes Smart Contracts. Siempre es recomendable evitar todas ellas pero algunas, lógicamente, pueden ser más peligrosas que otras.

Source File	Scan Mode	Submitted at	Detected Vulnerabilities	Status
ERC20.sol	Standard	16/8/2021 12:37:46	0 H 0 M 0 L	Finished
Token.sol	Standard	16/8/2021 12:37:38	0 H 0 M 0 L	Finished
TokenManager.sol	Standard	16/8/2021 12:37:40	0 H 0 M 0 L	Finished

Como se puede observar en la captura adjunta, se ha ejecutado el analizador de código MythX en busca de vulnerabilidades y no se han encontrado vulnerabilidades conocidas en ninguno de los contratos desarrollados (TokenManager y Token) ni tampoco en el contrato relacionado ERC20 de OpenZeppelin.

8. Calidad de código

Se ha realizado un correcto uso de librerías de la comunidad, como las de ERC20 de OpenZeppelin. Esto hace que el código generado sea poco, conciso y sencillo de seguir.

No se ha encontrado ninguna mejora a realizar para el objetivo de los smart contracts.

9. Conclusión

En el código no se han encontrado vulnerabilidades.

Adicionalmente al análisis manual realizado sobre el código hemos pasado analizadores estáticos de código como MythX, pero no hemos encontrado ninguna vulnerabilidad conocida.

10. Entregables

Los entregables finales de la auditoría son:

- Documento de auditoría de seguridad (presente documento).
- Proyecto Brownie con el código de las pruebas unitarias funcionales.