Sistema descentralizado para la gestión de credenciales a través de la red TOR

Andrés Mauricio Gómez Peña

Titulación: Máster Interuniversitario de Seguridad en las Tecnologías de la Información y de las comunicaciones (MISTIC)

Tutora: Silvia Puglisi
Responsable: Victor Garcia Font

Enero 2021

FICHA DEL TRABAJO FINAL

Título del trabajo:	Sistema descentralizado para la gestión de credenciales a través de la red TOR
Nombre del autor:	Andrés Mauricio Gómez Peña
Nombre de la consultora:	Silvia Puglisi
Nombre del PRA:	Victor Garcia Font
Fecha de entrega (mm/aaaa):	01/2021
Titulación:	Máster Interuniversitario de Seguridad en las Tecnologías de la Información y de las comunicaciones (MISTIC)
Área del Trabajo Final:	Protocolos y aplicaciones de seguridad
Idioma del trabajo:	Español
Palabras clave	Sistema descentralizado, TOR, credenciales

INDICE

RESUMEN

A medida que se conocen casos de censura y vigilancia apalancados por empresas y gobiernos, se hace necesario contar con herramientas que mitiguen escenarios en contra de la privacidad de los usuarios. Este trabajo aborda el diseño y construcción de una plataforma descentralizada apalancada en la red Tor, para la gestión de información crítica y confidencial como lo son credenciales y secretos de los usuarios.

Teniendo en cuenta la importancia de la información tratada y las repercusiones que podría implicar el no tener acceso a esta información o el que un tercero lo acceda sin el consentimiento del propietario, el trabajo expone las razones y argumentos que motivaron la elección por ofrecer un sistema descentralizado, fuera del control de gobiernos, empresas y terceros, apalancándose bajo una red de anonimato y cifrado como la ofrecida por la red Tor, utilizando mecanismos de seguridad y usabilidad que deben ser considerados al utilizar este tipo de redes.

Finalmente se presenta a disposición del público, una versión preliminar de una plataforma descentralizada y open source que he denominado dVault, para la gestión de credenciales a través de la red Tor que la comunidad puede usar, desplegar y mejorar.

ABSTRACT

As cases of censorship and surveillance leveraged by companies and governments become known, it is necessary to have tools that mitigate scenarios against the privacy of users. This work deals with the design and construction of a decentralized platform leveraged on the Tor network, for the management of critical and confidential information such as user credentials and secrets.

Taking into account the importance of the information processed, the repercussions as a result of not having access to this information, and the risk of a third party accessing it without the consent of the owner, the work sets out the reasons and arguments that motivated the choice to offer a decentralized system, A system which is outside the control of governments, companies and third parties, leveraging under a network of anonymity and encryption such as that offered by the Tor network, using security and usability mechanisms that must be selected when using this type of network.

Finally, a preliminary version of a decentralized and open-source platform that I have called dVault, credential management through the Tor network, the community can use, deploy and improve, is presented to the public.

INTRODUCCIÓN

Contexto y justificación del Trabajo

Hoy día es posible encontrar servicios cloud cuya finalidad es la gestión de información sensible y confidencial como secretos, credenciales y llaves de acceso. Plataformas como 1Password o Lastpass le permiten a sus usuarios obtener, almacenar, actualizar y/o eliminar este tipo de información de forma sencilla y amigable. Por lo regular, estas plataformas implementan arquitecturas centralizadas **donde el acceso a la información y a los servicios, son proveídos, controlados y/o gobernados por la misma plataforma**.

Se debe tener presente que usar aplicaciones centralizadas implica asumir algunos riesgos que afectan la privacidad y disponibilidad de los datos, riesgos que normalmente no son contemplados al momento de elegir un servicio cloud.

Resalta así por ejemplo el artículo “Centralized Information Systems and the Legal Right to Privacy” de 1969 por Jeffrey A. Meldman, el cual resalta la posibilidad de que investigadores, organismos gubernamentales , empresas privadas o agencias de investigación puedan adquirir información detallada sobre un individuo de forma rápida y sencilla, tomando especial valor e interés las arquitecturas centralizadas en las cuales estas entidades se pueden apalancar.

Igualmente se resaltan otros riegos propios de centralizar la información en único punto, según algunas fuentes de información, se han presentado brechas de seguridad y problemas de disponibilidad en sistemas centralizados que han evidenciado los problemas intrínsecos de estos sistemas. Es así como en Septiembre del 2017, se presentó una brecha de seguridad en los servicios de la empresa Equifax[1], el cual permitió la filtración de datos personales de más de 148 millones de personas.

Casos como el de la empresa Equifax y el de otras más, deben hacerle recapacitar al lector, si debe ser adecuado delegar y centralizar información tan confidencial y privada como credenciales y secretos a un tercero.

Teniendo en cuenta la sensibilidad de los valores tratados y las implicaciones a nivel de privacidad y disponibilidad que un sistema centralizado puede ejercer, este trabajo aborda el diseño e implementación de un **sistema descentralizado apalancado en la red TOR** cuyo objetivo es subsanar los problemas intrínsecos de un sistema centralizado en servicios enfocados en la gestión de credenciales, de tal manera que este tipo de información sensible sea fragmentada, cifrada y distribuida de forma redundante entre los nodos de la red con el objetivo de mejorar la privacidad y disponibilidad de la información.

Es así como se espera que el sistema le permita al usuario:

- Obtener y almacenar credenciales en el sistema
- Replicar la información en otros nodos para obtener mayor redundancia y disponibilidad
- Reducir el riesgo de que el servicio sea controlado, gobernado y/o censurado por una entidad o gobierno
- Evitar centralizar y entregar información sensible a terceros

El sistema se apalancará en la red de anonimato TOR, cuya finalidad en este sistema, es permitirle a los usuarios:

- Poder ofrecer el servicio como un onion service con la seguridad que no serán objeto de ataques dirigidos contra su infraestructura
- Permitir acceder al sistema sin exponerlo al exterior, de tal manera que este podrá ser consumido desde la red TOR y optar por no recibir peticiones desde internet o la red cercana a la cual pertenece, de nuevo con el objetivo de evitar ataques dirigidos
- Poder acceder a las credenciales y secretos sin que los nodos servidores u terceros puedan determinar el origen de las peticiones

¿Por qué ofrecer un sistema de gestión de credenciales como un sistema descentralizado?
Teniendo en cuenta el nivel de confidencialidad requerido por el tipo de información tratada, se busca ofrecer al usuario la posibilidad de no delegar esta información importante en un tercero, teniendo en cuenta que este último puede verse en la obligación de aportar datos del usuario a un gobierno o denegar el acceso a la información del usuario.

Es por ello que esta iniciativa descentralizada no busca ser ofrecida por una entidad ni servidor específico, sino que por el contrario apoyarse en una infraestructura descentralizada y apalancada por los mismos usuarios del sistema.

¿ Cómo garantizar la confidencialidad ?
La información (Secretos / Credenciales) son cifrados y divididos en fragmentos antes de ser enviadas a los nodos, es por ello que un nodo nunca tendrá la información completa de una credencial de un usuario, sino un fragmento de este, de tal manera que dado un nodo comprometido, no es factible que se obtenga la información completa de un usuario.

¿ Por qué se ofrece mayor seguridad del lado del sistema?
Al ser un sistema descentralizado, se dificulta poder aplicar ataques dirigidos de Denegación de Servicio DDOS al servicio en general, pues este no se ofrece de forma centralizada.

¿ Por qué se ofrece mayor seguridad del lado del usuario - TOR ?
Al usar la red TOR, los nodos no se conocen directamente a nivel de red de comunicación IP, por lo tanto el nodo/ servidor de un fragmento para un usuario, no conoce al nodo/usuario que le está solicitando la información y por lo tanto el nodo no puede determinar que almacena parte del secreto de un usuario en específico.

¿ Por qué se ofrece mayor seguridad del lado del servidor - TOR ?
Los servicios WEB por lo regular permiten el acceso desde internet y la red cercana al host, facilitando ataques dirigidos contra un nodo en específico. Al usar la red TOR, el servicio únicamente se podría consumir si proviene desde la red de TOR, por lo tanto, ni usuarios en internet ni en la red cercana al servidor puede iniciar ataques dirigidos contra un servidor/usuario en específico si no conocen el servicio Onion del nodo.

Objetivos

Objetivo General

Poder ofrecer una plataforma alternativa para la gestión de credenciales de forma descentralizada apalancada en la red de anonimato TOR

Objetivos Específicos

- Identificar y documentar los beneficios y desafíos de ofrecer sistemas descentralizados
- Entender el funcionamiento y protocolo de la red TOR
- Entender la forma en el cual es posible ocultar servicios detrás de la red TOR mediante rendezvous points
- Identificar y documentar los riesgos de privacidad y disponibilidad al delegar los datos e información en aplicaciones centralizadas
- Identificar una alternativa adecuada para el cifrado de la información, contemplando esquemas simétricos y asimétricos
- Construcción de una plataforma funcional para la gestión de credenciales que los usuarios puedan desplegar

Descripción de la metodología

De acuerdo a la problemática planteada e identificada, la metodología a seguir implica desarrollar un sistema funcional para la gestión de credenciales, no sin antes abordar algunos puntos teóricos importantes que sentarán las bases para poder justificar el trabajo, es por ello que el enfoque, metodología y fases a seguir para dar cumplimiento a los objetivos son:

Plan de trabajo

Se presenta el contexto del problema, la propuesta para solventarla y la necesidad de apalancarse en sistemas descentralizados bajo la red TOR . Se definen los objetivos del trabajo, la metodología utilizada, las tareas requeridas y los riesgos identificados.

Análisis de las implicaciones de sistemas centralizados y descentralizados

Se inicia con una introducción e identificación de las características de sistemas centralizados y descentralizados, se presentan los beneficios, falencias y riesgos que este tipo de arquitecturas pueden tener, profundizando en los problemas de privacidad y disponibilidad que pueden ejercer las arquitecturas centralizadas. Esto con el objetivo de poder sustentar la necesidad de implementar un sistemas de gestión de credenciales de forma descentralizada.

Onion Services

Se procede posteriormente con una presentación de la red TOR, para ello se realiza una introducción, funcionamiento, beneficios y detalles del protocolo. Igualmente se presentará el funcionamiento de servicios onion. En este apartado se resalta el cómo se beneficia una aplicación descentralizada de utilizar los servicios en la red TOR.

Funciones Hash y Mecanismos de Cifrado

De acuerdo a las necesidades de seguridad y privacidad requeridas para el sistema, se realizará una revisión de los mecanismos actuales de cifrado y hash para identificar la función hash que permita mitigar ataques de fuerza bruta contra la credencial de login, igualmente se elegirá el esquema de cifrado que mejor se adecue a la necesidad, contemplando esquemas tanto simétricos como asimétricos.

DHT Kademlia

Teniendo en cuenta que la aplicación debe ser construida bajo un esquema descentralizado, en este apartado se presenta y profundiza en tablas hash distribuidas bajo el algoritmo kademlia, el cual será el mecanismo utilizado en la implementación del sistema que permitirá habilitar las capacidades descentralizadas.

Arquitectura y Stack tecnológico

Una vez establecida la motivación para la creación del sistema y la identificación de los fundamentos técnicos requeridos en la implementación, se procede a definir la arquitectura aplicativa, stack tecnológico, casos de uso del aplicativo, todo esto justificado en el estudio previo, contemplando lenguajes acordes a las necesidades, facilidad de integración con la red TOR y esquemas de cifrado, entre otros factores que serán considerados para la definición de la arquitectura y stack tecnológico.

Implementación del Sistema

Una vez clara la arquitectura y stack tecnológico, se procede con la implementación del sistema, para ello se pretende aplicar un ciclo iterativo e incremental donde se realice un análisis, diseño, implementación y despliegue el aplicativo para su funcionamiento, involucrando tanto la interfaz gráfica Frontend como servicios Backend.

Redacción de la Memoria

A medida que se va avanzando por las diferentes fases del sistema, se va documentando el proceso de implementación y recopilación de información necesaria.

Tareas

Se procede a detallar cada fase del punto anterior.

1. **Análisis de las implicaciones de sistemas centralizados y descentralizados**
   - 1.1. Arquitecturas centralizadas: En esta sección se profundiza en las características de los sistemas centralizados, recorriendo su definición, beneficios y riesgos.
   - 1.2. Arquitecturas descentralizadas: En esta sección se profundiza en las características de los sistemas descentralizados, recorriendo su definición, beneficios y riesgos.
   - 1.3. Impacto en la privacidad, seguridad y disponibilidad: En este apartado se resaltan los problemas a la privacidad, seguridad y disponibilidad tanto del servicio como de los datos en aplicaciones centralizadas.
2. **Onion Services**
   - 2.1. Red TOR: Se realiza una introducción, presentación y beneficios de usar la red TOR.
   - 2.2. Onion Services: Se aborda la forma de ofrecer servicios en la red TOR, se explica el modelo de “rendezvous points”

3. Funciones Hash y Mecanismos de Cifrado

3.1. Funciones hash: Teniendo en cuenta la necesidad de contar con un sistema de login para acceso al servicio, se realizará una revisión del estado actual de las funciones hash, abordando los diferentes mecanismos que involucran el uso de pepper, salt y funciones PBKDF2 con el objetivo de mitigar ataques de fuerza bruta. Se evaluará igualmente la conveniencia de los mismos en este sistema.

3.2. Mecanismos de cifrado: Teniendo en cuenta que el sistema debe cifrar información, se realizará una revisión del estado actual de los mecanismos de cifrado, abordando el uso y conveniencia de mecanismos simétricos y asimétricos.

4. Arquitectura y Stack tecnológico

4.1. Algoritmo Kademlia: Se presenta la estructura de datos DHT y el algoritmo Kademlia, el cual será el mecanismo que permitirá habilitar las capacidades descentralizadas del sistema.

4.2. Argumentación y defensa: Se presenta, argumenta y defiende la necesidad de contar con un sistema para la gestión de credenciales descentralizado

4.3. Argumentación y defensa: Se presenta y argumenta el por qué usar TOR en este sistema, abordando los beneficios intrínsecos de usar la red TOR, y la conveniencia de aplicarlo a sistemas descentralizados.

4.4. Elección y defensa: Se elegirá y argumentará el mecanismo de cifrado elegido.

4.5. Definición de la arquitectura: Esta sección se presentan los casos de uso que debe soportar el sistema, los módulos necesarios a implementar y el Stack Tecnológico elegido.

5. **Implementación del Sistema**

5.1. Backend y Frontend: Implementación del Backend y Frontend como servicio descentralizado, habilitando los casos de uso y módulos identificados en la definición de la arquitectura.

5.2. Kademlia & TOR: Se realiza la integración del algoritmo Kademlia con la red TOR.

6. **Redacción de la Memoria**

6.1. Redacción de la memoria: Se hace una revisión general y validación de la memoria con bibliografía

6.2. Realización de la presentación: Se procede con la presentación que condensa los logros del trabajo final del grado.

6.3. Realización del video: Se procede con la preparación y grabación del video que condensa los logros del trabajo final del grado.

Planificación

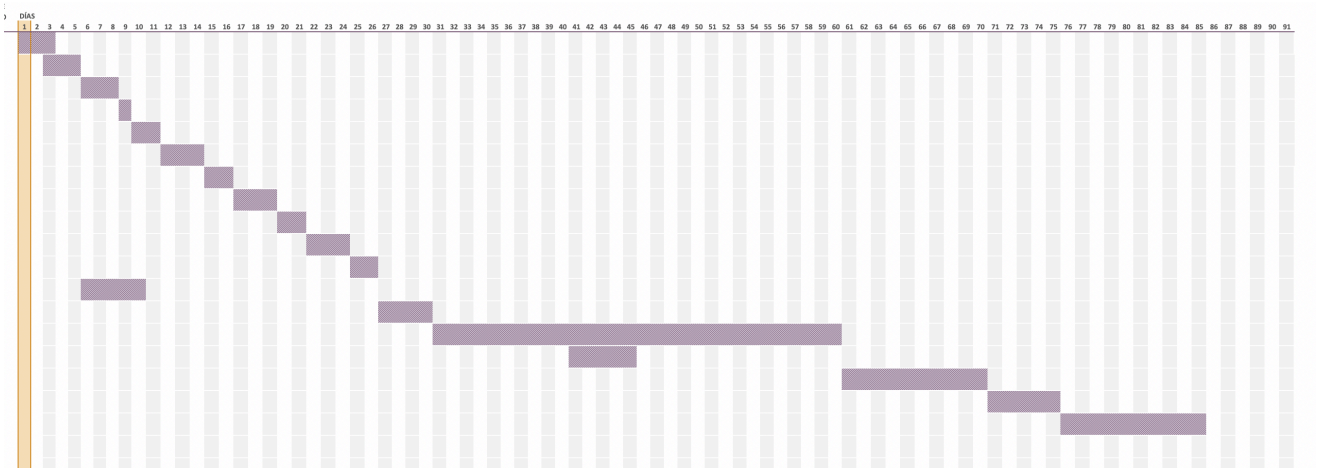
A continuación se presenta el diagrama de gantt, en el cual se aprecia el orden cronológico y el tiempo requerido para poder ejecutar cada una de las tareas. Se resalta que la duración está dada en días.

Planificador de proyectos

Seleccione un periodo para resaltarlo a la derecha. A continuación hay una leyenda que describe el gráfico.

Día resaltado: 1

Obejtivo	Tarea	INICIO DEL PLAN	DURACIÓN DEL PLAN	INICIO REAL	DURACIÓN REAL	PORCENTAJE COMPLETADO	DÍA 1
Implicaciones De sistemas Centralizados y Descentralizados	**1.1**	1	3	1	3	0%	
	1.2	3	3	3	3	0%	
	1.3	6	3	6	3	0%	
	1.4	9	1	9	1	0%	
Onion Services	**2.1**	10	2	10	2	0%	
	2.2	12	3	12	3	0%	
	2.3	15	2	15	2	0%	
Funciones Hash y mecanismos de Cifrado	**3.1**	17	3	17	3	0%	
	3.2	20	2	20	2	0%	
	3.3	22	3	22	3	0%	
	3.4	25	2	25	2	0%	
Arquitectura y Stack Tecnológico	**4.1**	6	5	6	5	0%	
	4.2	27	4	27	4	0%	
Implementación del Sistema	**5.1**	31	30	31	30	0%	
	5.2	41	5	41	5	0%	
Redacción de la memoria	**6.1**	61	10	61	10	0%	
	6.2	71	5	71	5	0%	
	6.3	76	10	76	10	0%	



Riesgos Preliminares

Se presentan los principales riesgos que pueden afectar la culminación del presente trabajo. Por cada riesgo se presenta la probabilidad e impacto de la materialización del mismo, igualmente se proponen acciones de mitigación.

- **Alcance sobre-dimensionado**: El alcance del proyecto cubre una parte teórica y práctica que hay que respetar en tiempos correctamente para poder estar acorde al calendario previsto.
  - **Probabilidad**: Media - **Impacto**: Alto
  - **Acciones**: Con el fin de anticiparse a problemas de alcance e implementación, se debe identificar las tareas superficiales susceptibles de no ser incluidas en el ámbito del trabajo y que no fueron contempladas inicialmente.
- **Implementación del protocolo DAT bajo la red TOR:** El sistema se apoyará bajo el toolkit y protocolo DAT, sin embargo la documentación y comunidad del mismo no es lo suficientemente alta como para poder asegurar que se facilitará la integración con la red TOR y servicios onion.
  - **Probabilidad**: Alta - **Impacto**: Alto
  - **Acciones**: Para mitigar un poco el riesgo, se buscará apoyo tanto en la tutora con su amplia experiencia en la red TOR como en la comunidad DAT que ya haya tenido un acercamiento con el uso del protocolo.
- **Uso de Onion Services**: Existe la posibilidad de que por la forma en el cual se implemente el protocolo DAT usando la red TOR, no sea necesario el uso de Onion Services.

- **Probabilidad**: Desconocida - **Impacto**: Bajo
- **Acciones**: Para mitigar un poco el riesgo, en etapas tempranas se evaluará la viabilidad del uso de servicios onion y la forma en el cual se debe integrarse el protocolo DAT con la red TOR, esto con el objetivo de identificar la forma adecuada de usar TOR con el toolkit.

Productos Obtenidos

Se espera finalmente, que una vez terminada cada unos de los entregables preliminares exigidos por la universidad, se pueda contar al final del trabajo final del grado con:

- Código versionado de la implementación del sistema, lo cual incluye código frontend y backend
- Imagen Docker que facilite el despliegue de la plataforma
- Una plataforma descentralizada para la gestión de credenciales a través de la red TOR
- Documento PDF, presentación y video en el cual se expone lo realizado en el trabajo final de grado.

ESTADO DEL ARTE

Hoy día es muy frecuente encontrar servicios centralizados y destinados a la gestión de información de credenciales y secretos. Servicios como 1Password, Lastpass o Keeper permiten a sus usuarios poder almacenar, obtener y administrar este tipo de información de forma sencilla y simple.

Al buscar iniciativas descentralizadas enfocadas en esta misma capacidad, las alternativas se reducen considerablemente, siendo la mayoría proyectos OpenSource en Github como **Encrypt My Data** [3] o **Savana** [4], el cual se apoyan en diferentes tecnologías incluyendo blockchain. Estos proyectos se destacan por no estar soportados ni operativos a modo de servicio.

Actualmente la iniciativa más madura es **BlockVault** [5], quien se autodenomina como una alternativa descentralizada a 1Password. BlockVault está construida en Blockstack , el cual es una plataforma descentralizada que facilita la construcción y despliegue de aplicaciones descentralizadas apoyándose en tecnologías como Blockchain y Bitcoin. Se destaca que BlockVault no se apalanca en la red TOR para ofrecer sus servicios.

Al buscar por iniciativas descentralizadas y apalancadas en la red TOR, no fue posible identificar ninguna iniciativa con tales características, es por ello que se asume por el momento que no existe un proyecto ni iniciativa apalancada en estas tecnologías.

Arquitecturas

Tanto servicios web como sistemas de información pueden ser implementados bajo distintas arquitecturas y topologías, siendo recurrente encontrar modelos centralizados, descentralizados y distribuidos, cada uno de ellos ofreciendo beneficios y riesgos a la privacidad, seguridad y disponibilidad de los datos y del servicio en sí mismo. A continuación se comparan los esquemas centralizados y descentralizados, partiendo desde su definición, características, beneficios y algunos riesgos o consideraciones a tener presente.

Arquitecturas centralizadas

Este tipo de arquitecturas se caracterizan por ofrecer servicios y/o aplicaciones que son propiedad de compañías o personas específicas, quienes son los encargados de mantener, asegurar, gobernar y aprovisionar los recursos necesarios para ofrecer el servicio, sistema de información o aplicación. En un esquema centralizado, el poder y recursos de operación son delegados a un tercero en el cual el usuario confía, depositando allí información personal y privada. Estas unidades centrales ejercen un gobierno y control sobre el acceso a los datos y al servicio en general, ejerciendo así autoridad sobre las acciones del usuario y autonomía sobre las acciones ejecutadas por el servidor.

No resta recordar que la gran mayoría de aplicaciones y servicios hoy día ofrecidos en internet se apoyan en este tipo de arquitecturas, pues ofrecen ventajas importantes respecto a modelos descentralizados, destacando así:

Control del servicio

Según los requerimientos del sistema, en ocasiones se hace necesario contar con mecanismos que permitan invalidar acciones del usuario pues estas pueden ser resultado de fraudes o acciones malintencionadas por terceros. Es por ello que ejercer control sobre el estado de las acciones y los datos se ve apalancado en sistemas centralizados donde se tiene un control directo sobre estos, ejemplo de ello han sido los mecanismos antifraudes implementados por bancos para reducir fraudes en tarjetas [16] .

Actualización del sistema

Al estar centralizada la configuración y el aplicativo en sí mismo, tareas administrativas de mantenimiento y seguridad se ven beneficiadas con labores de actualización del sistema que son fácilmente desplegables, aplicadas y distribuidas a través de todo el sistema.

Monitoreo del sistema

Al tener control y acceso sobre todos los recursos del sistema, tareas de monitoreo y almacenamiento de logs son fácilmente aplicables a la totalidad del sistema.

Implicaciones de sistemas centralizados

Existen algunas consideraciones a tener presente relacionados con sistemas de información apoyados en esquemas centralizados, dentro de los cuales se destacan:

Mecanismos de Seguridad desconocidos

Por lo regular, cuando se realiza una solicitud hacia un sistema, el usuario no es consciente del funcionamiento y operación del servidor, desconociendo el nivel de acceso al cual tiene este, la metadata almacenada y el nivel de seguridad implementado por el sistema. De hecho no es extraño encontrar fallos de seguridad relacionados con la forma en el cual se tratan los datos por parte del tercero en el cual se delegan los datos, ejemplo de ello han sido Google [6], Facebook [7] y Twitter [8] quienes han reconocido públicamente haber encontrado bugs y fallos de seguridad relacionados con el tratamiento del password de acceso a la plataforma.

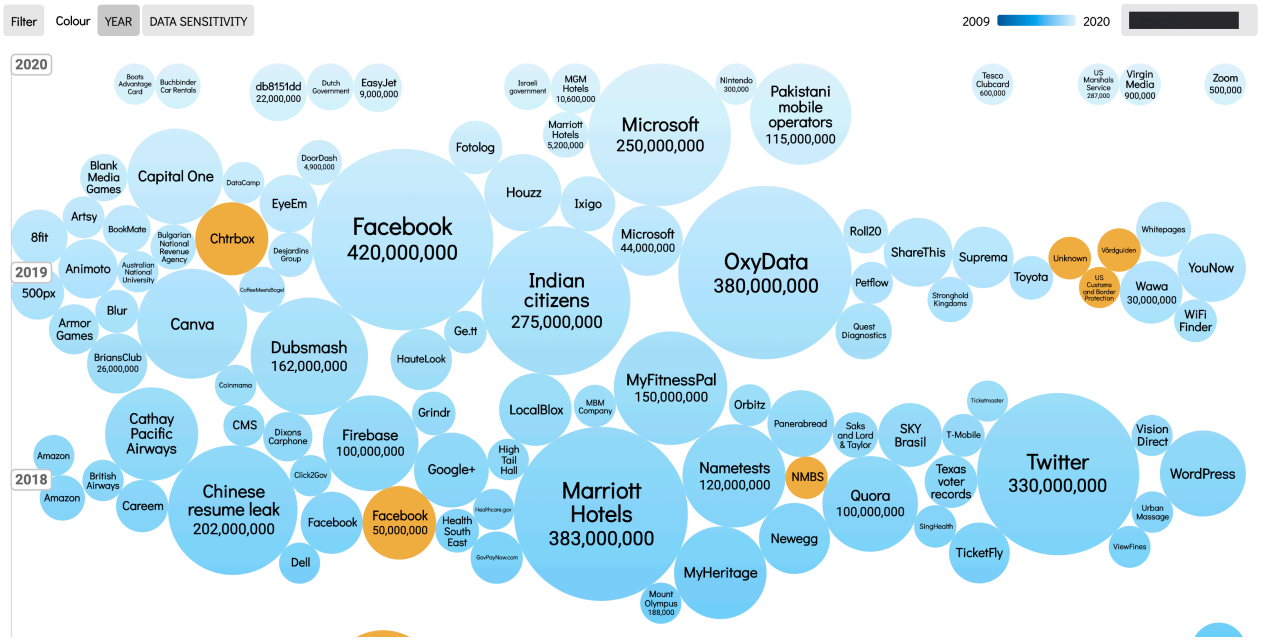
Riesgos para la privacidad de los usuarios

Se debe tener presente que cuando se tiene centralizado y controlado el acceso a la información, es factible y susceptible a que personas internas como empleados o terceros como agencias de seguridad se interesen y/o soliciten acceso a la información allí depositada, ejemplo de ello han sido las reiteradas intenciones del FBI hacia Apple solicitando desbloqueo y acceso a la información de diferentes usuarios [9] o el acceso a la información por parte de empleados de Amazon recopilados por dispositivos Alexa [11].

DDoS & Fugas de datos

De la misma manera que agencias de seguridad se interesan en la información recopilada y centralizada en sistemas de información, hackers y usuarios malintencionadas buscan desestabilizar el sistema y obtener acceso de igual manera a la información, siendo frecuentes que estos sistemas sean objetos de ataques de Denegación de Servicio DDoS como lo expone Google[10] contra sus servicios y fugas de datos como el de Equifax [1] que al estar centralizados facilitan e incrementan el daño ocasionado.

Un recurso interesante por explorar es el gráfico sobre **“Biggest Data Breaches”** [12] el cual permite identificar fácilmente el tamaño e impacto que han tenido diferentes brechas de seguridad en diferentes sistemas centralizados a lo largo del tiempo.



World's Biggest Data Breaches & Hacks (17 Oct 2020)

Censura

El control y gobierno ejercido por un sistema central puede llegar a ser tal que intereses empresariales o peticiones judiciales pueden limitar el acceso al servicio o censurar la opinión del usuario, ejemplo de esto han sido la reciente censura aplicada por Facebook y Twitter en las elecciones presidenciales del año 2020 [13] o el cierre aplicado hace algunos años a la plataforma Megaupload [14] la cual impidió que la totalidad de usuarios lograran acceder a sus datos.

Los niveles de censura han llegado hasta tal punto, que países y gobiernos logran detener todo el tráfico de internet nacional con el objetivo de reducir el intercambio de opiniones y movimientos civiles, ejemplo de ello fue lo sucedido en la primavera árabe [15] donde Egipto, Siria y Libia realizaron este tipo de censura.

Escalabilidad

Queda a responsabilidad del propietario del servicio garantizar la escalabilidad del sistema ante la demanda de los usuarios, incluyendo nodos de computo, espacio en disco, RAM, entre otros recursos necesarios para la ejecución del sistema, recursos que llegan a niveles tales como los de YouTube el cual debe poder garantizar alrededor de 500 horas de video cargado por minuto a la plataforma [18].

Arquitecturas descentralizadas

Este tipo de arquitecturas se caracterizan por ofrecer servicios y/o aplicaciones que no son propiedad de una entidad o persona en particular, sino que se apalancan en un gobierno descentralizado donde existen múltiples nodos con responsabilidades de control y gobierno, resalta igualmente mayor estabilidad y tolerancia a fallos que en un sistema centralizado, pues características como redundancia y alta disponibilidad son inherentes a la arquitectura, al descargar la responsabilidad de ejecución del sistema en múltiples nodos.

Es importante resaltar los siguientes beneficios:

Alta disponibilidad & Redundancia

Al descargar la responsabilidad de ejecución sobre múltiples nodos, la posibilidad de que un nodo falle es contrarrestada con la posibilidad de que otro nodo pueda atender a la solicitud, para ello se hace necesario contar con mecanismos de replicación de datos y de enrutamiento dinámico.

Promueve compartir recursos

El sistema al no ser propiedad de una entidad ni persona en específico, se apalanca en los recursos de computo ofrecidos por los nodos que son parte de la red, nodos que se ven beneficiados por el uso del sistema a cambio de ofrecer una porción de sus capacidades de hardware a la red.

Altamente escalable

Cada vez que un nodo ingresa a ser parte de la red, este ofrece sus recursos de computo a la misma, de esta manera entre mayor sea el tamaño de la red, mayor será la capacidad de procesamiento y almacenamiento de información, lo cual le ofrece al sistema en general un grado mayor de protección contra ataques DDoS.

Se debe resaltar que al no estar la información ni el servicio centralizado, ni ser controlado por alguna entidad en específico, los riesgos de seguridad, privacidad, disponibilidad y acceso a la información se ven limitados al nivel de seguridad ofrecido por los mecanismos de seguridad establecidos por la arquitectura del sistema, eliminando de esta manera cualquier intento por control y autoridad establecido por una persona, entidad, empresa o agencia de seguridad.

Implicaciones de sistemas descentralizados

Se presenta a continuación algunas consideraciones a tener presente al implementar sistemas descentralizados.

Mantenimiento

Por su naturaliza descentralizada se dificulta poder ofrecer nuevas funcionalidades o aplicar actualizaciones de seguridad a la totalidad de nodos, pues al no estar controlados por una entidad en específico sino por la

autonomía del usuario dueño del nodo, queda a responsabilidad de este el estar atento a nuevas actualizaciones del sistema.

Control

Al igual que como ocurre en otros sistemas descentralizados como la red TOR en el cual se pueden realizar actividades ilegales sin mayor control [7] , en un sistema descentralizado se dificulta ofrecer un control sobre la acciones que realizan los usuarios, quedando a disposición de los controles propios del sistema cualquier acción en busca de fomentar la seguridad, privacidad y legalidad de las acciones.

Disponibilidad

Se debe tener presente que estos sistemas al apalancarse en una infraestructura del cual no se tiene un control directo, donde los nodos por lo regular no se encuentran disponibles 24/7, es factible que se presenten problemas de indisponibilidad de acceso a la información o al servicio. Es por ello que se deben buscar estrategias como replicación que busquen mitigar esta problemática.

Anonimato & Tor

Quizás sea por una actividad delictiva, una denuncia empresarial o simplemente el no ser rastreado, por el cual el ser humano ha buscado realizar algunas de sus actividades, obras y opiniones de forma anónima, de tal manera que se limite o dificulte correlacionar directamente al autor de las mismas.

Noticias recientes han demostrado la necesidad de apalancarse en la tecnología para evitar la censura y actos en contra de la vida cuando se desea expresar la verdad y realidad, ejemplo de ello han sido las denuncias realizadas por periodistas y personas sobre el manejo de la pandemia Covid-19 en China, las cuales han tenido un desenlace limitando la libertad de expresión de estos [20].

De la misma manera que algunas denuncias buscan ser anónimas, las obras y aportes a la comunidad también lo pueden ser, ejemplo de ello fue el paper inicial del bitcoin, el cual fue presentado bajo el seudónimo de Satoshi Nakamoto [19].

En el ámbito de internet, el proyecto TOR se ha posicionado como uno de los mecanismos mayormente adoptados para obtener anonimato en la red, permitiendo que periodistas, activistas y personas puedan expresar libremente sus opiniones y actividades sin ser objeto del rastreo por gobiernos como el revelado por Edward Snowden [21] .

Dentro de los beneficios de usar la red TOR se destacan:

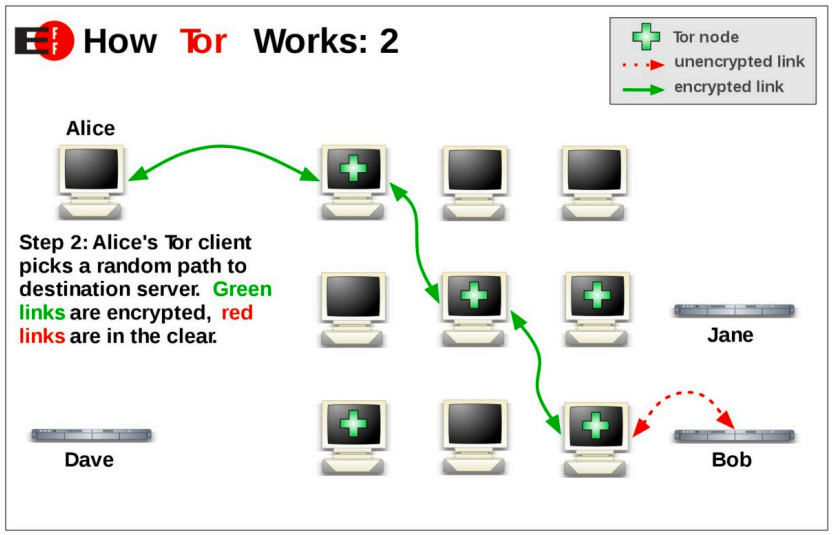
- **Bloqueo de Trackers**: Mediante Tor Browser es posible evitar que algunas cookies de terceros y anuncios puedan rastrear la navegación por internet del usuario.

- **Anonimato:** Tor evita que sea posible identificar las páginas web visitadas por un usuario aportando así anonimato.
- **Rastreo:** Existen mecanismos que permiten identificar al usuario basado en una huella digital del navegador y del dispositivo. TOR Browser evita que esto sea posible aparentando una similitud entre los usuarios y sus navegadores.
- **Cifrado:** El tráfico que es enviado por la red TOR es encriptado mínimo 3 veces.
- **No Censura:** En algunas ocasiones tanto empresas como gobiernos bloquean el acceso a cierto contenido en la web, mediante TOR se puede evitar este tipo de censura.

Funcionamiento de Tor

La forma en el cual la red Tor funciona es mediante un conjunto de nodos voluntarios por los cuales es enrutado el tráfico origen hasta llegar al nodo destino.

Así por ejemplo cuando Alice desea comunicarse con Bob, el proceso inicia eligiendo un nodo de entrada (entry node) aleatorio, posteriormente se elige un nodo intermediario (middle relay) y un nodo de salida (exit note), de esta manera se crea un canal cifrado de comunicación donde existen al menos 3 nodos intermediarios.



De esta manera cualquier intento por interceptar la comunicación o conocer el origen o destino de la comunicación por terceros, sean proveedores de internet (ISP), gobierno, hackers o servidores, estará limitado a un contexto local y no total.

En nuestro ejemplo:

- El ISP de Alice únicamente detectará que ella quiere comunicarse con el nodo de entrada (y no con Bob)
- El nodo de entrada conoce a Alice y al nodo intermediario (No conoce a Bob)
- El nodo intermediario se comunica con el nodo de entrada y el nodo de salida (No conoce a Bob ni a Alice)

- El nodo de salida conoce al nodo intermediario y el nodo destino Bob, pero no conoce a Alice

De esta manera cada nodo conoce solo una parte de la comunicación otorgando un anonimato a nivel de red. Se debe tener presente que la velocidad de navegación por la red se ve afectada teniendo en cuenta la cantidad de nodos que intervienen en el proceso.

A continuación se presenta una tabla resumen sobre lo que cada nodo conoce sobre el mensaje y la comunicación en un circuito Tor, tomado de [22].

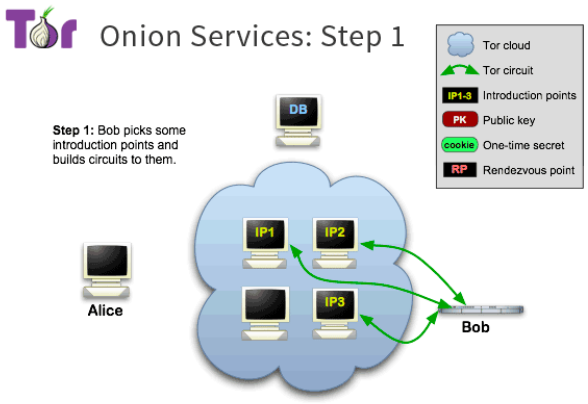
	user	bridge node or entry guard	middle node	exit node
Tor user's IP/location	yes	yes	no	no
IP of bridge node or entry guard	yes	yes	yes	no
message for bridge node or entry guard	yes	yes	no	no
IP of middle node	yes	yes	yes	yes
message for middle node	yes	no	yes	no
IP of exit node	yes	no	yes	yes
message for exit node	yes	no	no	yes
IP of destination server	yes	no	no	yes
message for destination server	yes	no	no	yes

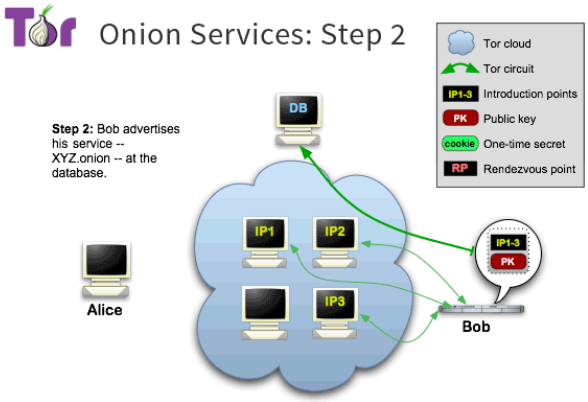
Which Tor node knows what ? [22]

Onion Services

Una de las capacidades permitidas por el proyecto TOR, es la posibilidad de ofrecer servicios y recursos WEB sin exponer directamente los servidores a internet ocultando así su localización. Para ello Tor se apoya en lo que se conoce como ”rendezvous points”, el proceso para ofrecer un servicio onion es el siguiente:

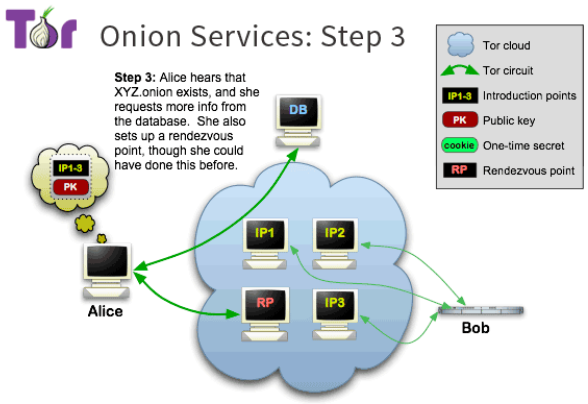
- Se crea un par de llaves por parte del servidor onion de Bob, posteriormente se crea un circuito Tor (Canal que consta de al menos 3 nodos intermediarios)
- Se eligen varios nodos Tor que serán los ***introduction points*** al servicio onion IP1 a IP3.



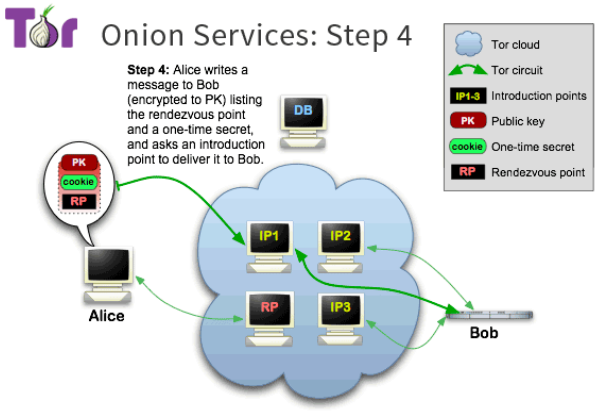


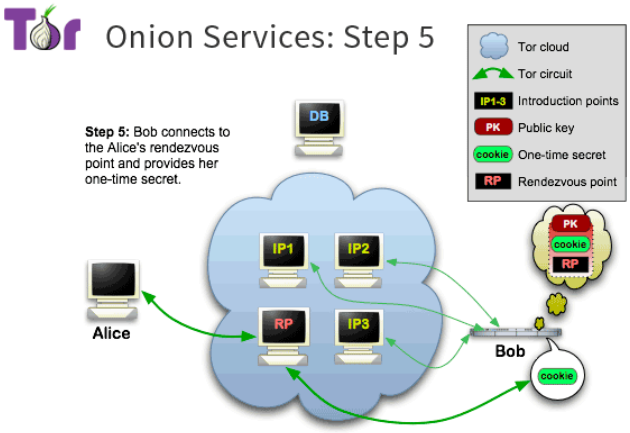
- Se crea el ***onion service descriptor***, el cual consta de la llave pública y la lista de los puntos de introducción, se firma el descriptor con la llave pública y se guarda el descriptor en una distributed hash table ***DHT***. Se resalta que el nombre de dominio del servicio onion es de la forma XYZ.onion donde XYZ se deriva de la llave publica.

- Una vez Alice conoce el nombre del servicio onion, este descarga el descriptor respectivo desde el DHT y crea un circuito Tor hacia un nodo de la red, al cual le solicita que actúe como un ***rendezvous point*** otorgándole un ***one-time secret***.



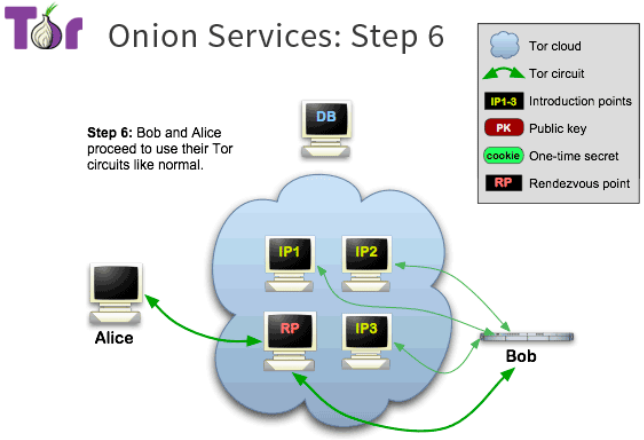
- El usuario crea un ***introduce message***, incluyendo la dirección del rendezvous point y el one-time secret, el cual cifra con la llave pública del servicio onion. Posteriormente se envía este mensaje a un punto de introducción del servicio para que sea enviado finalmente al servicio onion.





- El servicio Onion descifra el introduce message, toma la dirección del rendezvous point y el one-time secret, posteriormente crea un circuito Tor hacia el rendezvous point enviándole el secreto mediante un ***rendezvous message***.

- En el último paso el rendezvous point notifica a Alice acerca de la conexión exitosa y se procede con la comunicación entre Bob y Alice por medio del rendezvous point, cada uno de ellos con un circuito Tor.



Se debe tener presente que al final la comunicación consta de 6 capas intermediarias, 3 para Alice y otras 3 para el onion service de Bob.

Cifrado & Hash

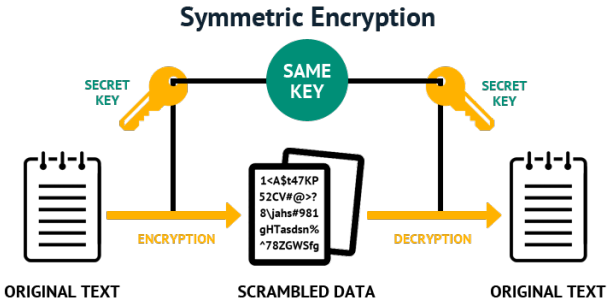
Cifrado

Desde la antigüedad el ser humano ha buscado proteger y garantizar la confidencialidad de la información que le pertenece, es por ello que se han buscado mecanismos que le permitan alterar la representación o codificación de esta. A este procedimiento se le denomina cifrado y ha evolucionado hasta contar con 2 mecanismos hoy día :

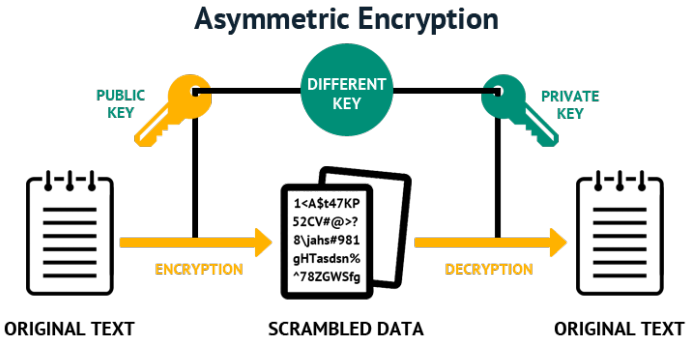
Cifrado Simétrico

Este mecanismo se caracteriza porque las claves de cifrado y descifrado son las mismas, es por ello que las partes interesadas deben garantizar una forma segura de compartir la llave entre ellas.

Este mecanismo es ampliamente utilizado para el cifrado de grandes volúmenes de información.



Algunos ejemplos actuales de cifrado simétrico computacionales son : DES, 3DES y AES.



Cifrado Asimétrico

Este mecanismo se caracteriza por utilizar 2 llaves, una en el proceso de cifrado conocida como llave pública y otra en el descifrado conocida como llave privada. Al utilizarse 2 llaves, las partes interesadas no deben preocuparse por asegurar el mecanismo para compartir la llave. Se resalta que la llave pública no puede ser utilizada para descifrar la información, sólo para cifrarla.

Este mecanismo es ampliamente utilizado para el intercambio de llaves de cifrado simétrico entre partes que no se conocen pero desean comunicarse de forma segura como entre un navegador web y un servidor web.

Algunos ejemplos actuales de cifrado asimétrico son: RSA y ECC - Elliptic curve cryptography.

Hash

Una función hash es aquella que puede ser utilizada para transformar datos de tamaño variable a un valor de tamaño fijo con la característica que no es reversible, normalmente se utiliza para verificar la integridad de la información pues permite obtener un “digests” o hash del recurso, pudiendo este último ser utilizado para identificar cambios en el mismo. Igualmente este mecanismo es ampliamente utilizado al momento de almacenar contraseñas en repositorios de datos, de tal manera que no sea posible obtener la contraseña en claro del hash almacenado.

A continuación se presentan ataques contra la seguridad de las contraseñas - Password Cracking, que deben ser considerados para el diseño del aplicativo:

Ataque de Fuerza Bruta

Ataque que consiste en evaluar todas las posibles permutaciones de caracteres y longitud de la cadena que conforman una contraseña hasta identificar la correcta.

Características:

- Si se utilizan los parámetros adecuados, como el tipo de caracter y la longitud máxima de la cadena a formar, se puede garantizar que se encontrará la contraseña.
- Ideal para contraseña cortas.
- Se puede paralelizar y distribuir la carga de trabajo a diferentes núcleos y nodos. Por ejemplo se puede hacer uso de GPU y de un cluster de servidores.
- No requiere de grandes volúmenes de espacio en disco.

Ataque de diccionario

Es una variación del ataque por fuerza bruta donde se hace uso de un listado de contraseñas previamente identificado, se diferencia con el anterior en que no se evalúan todas las posibles contraseñas sino aquellas que pueden tener un mayor potencial y probabilidad. Este método se apalanca en la frecuencia con la cual utilizamos palabras sencillas, comunes y asociadas a la persona, por ejemplo: Fecha de Cumpleaños, nombre, gustos, etc.

Características:

- Menor tiempo requerido para finalizar el proceso comparado con un ataque de fuerza bruta, sin embargo aún se requiere de tiempos altos.
- No se evalúan contraseñas con una probabilidad muy baja de ocurrencia.
- Se limita el rango de contraseñas a aquellas que corresponden con la personalidad de la persona.

Ataque de Lookup Table

Técnica utilizada para evaluar contraseñas de forma eficiente (en procesamiento y no en espacio), en sistemas que almacenan dicha información como el resultado de aplicar una función hash.

1. Se debe pre-calcular el hash de cada una de las posibles contraseñas (Listado mediante fuerza bruta o diccionario), y almacenar dicho valor en una tabla como una asociación hash (llave) - contraseña (valor).
2. Se debe identificar una coincidencia hash real - hash calculado y obtener el valor asociado a dicha llave.

De esta manera se reduce tiempos de computo asociado al cálculo del hash para sistemas que comparten la misma función hash y sus parámetros, permitiendo así compartir la tabla ya pre-calculada entre ellos.

Características:

- Efectivo cuando muchos usuarios utilizan la misma contraseña.
- Se reduce tiempos de computo para sistemas que comparten una misma función hash y sus parámetros.
- Basta con identificar una coincidencia entre el hash calculado y el hash real.

A continuación se presentan algunos mecanismos que permiten mitigar los ataques anteriormente comentados:

Key Stretching

Esta técnica busca incrementar la seguridad asociada a una contraseña contra ataques de fuerza bruta incrementando la cantidad de recursos (Tiempo y/o espacio) necesarios para poder realizar esta operación. Una forma de alcanzar este objetivo consiste en ejecutar el proceso de hash o cifrado repetitivamente, o hacer que el proceso tome un tiempo considerable en aplicar la función de hash. [25] De esta manera una ataque de fuerza bruta es impráctico por la cantidad de recursos que tomaría ejecutarse para todas las permutaciones, aún cuando la contraseña pueda ser algo débil.

Es así como se han implementado algoritmos como el PBKDF2 , cuyo diseño de por sí es lento, esto con el propósito de aplicar la técnica de Key Stretching. Recibe como parámetros los siguientes valores [26]:

- PRF: Función pseudo-aleatoria, por ejemplo sha2
- Password: Contraseña en texto plano
- Salt
- c: Número de iteraciones
- dkLen: Longitud en bits de la llave de salida

Una vez recibido los parámetros anteriores, el algoritmo aplica al password junto con la Salt, la función PRF. Este proceso se aplica tantas veces como se haya indicado, utilizando siempre la salida de la ejecución anterior. Al final se obtiene un hash con una longitud de dkLen.

Salt

El valor Salt es un valor aleatorio, conocido y único , que es añadido junto a la contraseña al momento de aplicar alguna función Hash [27], esto con el objetivo de que 2 contraseñas iguales NO generen el mismo hash. Este valor debería ser generado y almacenado cada vez que se va a aplicar la función.

Pepper

Pepper es un valor fijo conocido y almacenado de forma secreta, que es añadido junto a la contraseña al momento de aplicar alguna función Hash, esto con el objetivo de que al utilizar una función hash, este no genere

por defecto la misma salida [28]. A diferencia de Salt, este valor no es generado siempre, por lo que existe una probabilidad de que 2 contraseñas generen el mismo hash.

Al usar el mecanismo de Salt, si una base de datos es vulnerada, existe una pequeña ventaja al querer vulnerar el hash almacenado pues junto a este, se tuvo que haber almacenado el valor de salt. Caso contrario del mecanismo de pepper, donde este valor debe ser almacenado fuera del sistema donde se almacenan los hash. El inconveniente con este último es que si la aplicación o el secreto es comprometido, todos los hash pueden ser comprometidos puesto que comparten el mismo pepper. [29]

Finalmente se resalta que al añadir un valor adicional, sea mediante pepper o salt, a la contraseña, estamos cambiando por completo el hash asociado a la misma. De esta manera, las tablas pre-calculadas encontradas en internet son invalidadas puesto que aquellas fueron probablemente generadas únicamente aplicando la función hash por defecto.

FASE DE ANÁLISIS Y DISEÑO

Una vez identificada la necesidad a cubrir y el sistema a implementar, se debe ofrecer y satisfacer necesidades básicas de un sistema de gestión de credenciales, igualmente se deben considerar las limitantes de las tecnologías utilizadas y factores de usabilidad y seguridad teniendo en cuenta el tipo de arquitectura elegida.

Requisitos

Requisitos de la aplicación

Teniendo en cuenta que se busca ofrecer un servicio de gestión de credenciales descentralizado, se deben ofrecer las capacidades básicas del mismo, las cuales incluyen:

- Contar con un formulario de login
- Obtener un secreto de la red descentralizada
- Almacenar un secreto en la red descentralizada

Requisitos de Seguridad

Con el objetivo de ofrecer un sistema seguro y transparente en su funcionamiento, se debe tener presente que :

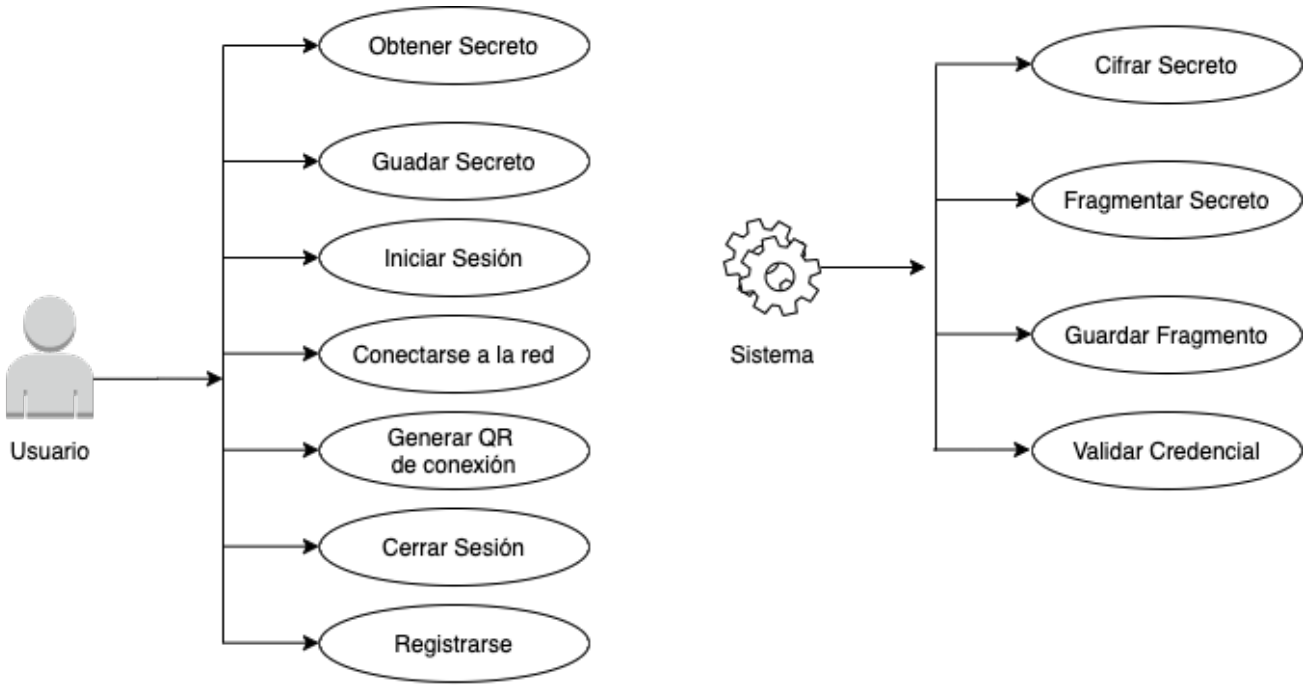
- La seguridad del sistema debe ser de conocimiento público, apalancada en algoritmos y esquemas de seguridad estándar de la industria
- Con el objetivo de ofrecer confidencialidad, todo secreto debe ser cifrado antes de ser fragmentando
- Con el objetivo de que un nodo nunca tenga en su totalidad el secreto de un usuario, se debe fragmentar cada secreto antes de ser distribuido y almacenado en la red, de esta manera un nodo únicamente almacenará el fragmento del usuario
- Los nodos deben ofrecer el servicio sin exponerse a ataques dirigidos contra el mismo, para ello se debe apalancar en la red TOR y onion Services
- Se deben contar con mecanismos que garanticen que un nodo puede determinar si la información que almacena no ha sido alterada
- La información debe ser transmitida por canales y protocolos de comunicación seguros
- El consumo de los servicios en localhost deben utilizar mecanismos de sesión

Requisitos de Usabilidad

Teniendo en cuenta la complejidad que implica tratar con sistemas descentralizado, se debe ofrecer un mecanismo que le facilite a los usuarios el ingreso a la red de nodos. Para ello :

- Se debe permitir configurar nodos “semilla”, los cuales serán de conocimiento público, estarán siempre activos y facilitarán ingresar a la red de nodos y descubrir otros nodos. Se debe tener presente que estos nodos no son diferentes a los demás, simplemente son de conocimiento público y no representan un esquema centralizado, puesto que para ingresar a la red de nodos pueden usarse otros nodos de la misma red.
- Se debe ofrecer un mecanismo que permita compartir una imagen QR con la información de conexión entre nodos, de esta manera el usuario que desee crear una red de confianza entre amigos y familiares se le facilitará realizar dicha tarea.
- Se debe contemplar un mecanismo sencillo para la instalación y despliegue del sistema, como el de distribución de imágenes Docker
- Se deben considerar mecanismos para mitigar la latencia existente al utilizar la red de Tor

Casos de Uso



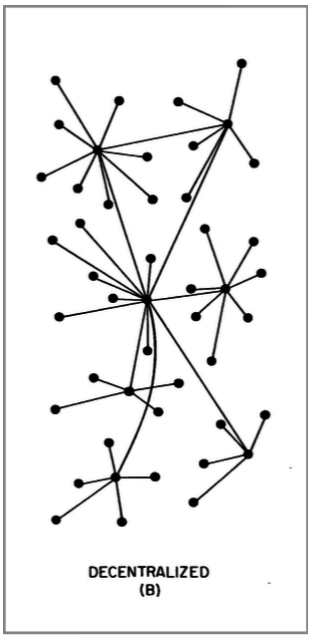
Es así como se espera que el usuario pueda:

- Elegir si desea conectarse a una red de nodos existente o crear su propia red, para ello el sistema se apalancará en la lectura y generación de códigos QR los cuales facilitaran el ingreso a la red. Se resalta igualmente la existencia de nodos semilla, los cuales serán de conocimiento público y permitirán facilitar el ingreso a una red por defecto.
- Registrarse como usuario en la red a la cual pertenece e iniciar sesión en la plataforma
- Obtener los secretos del usuario
- Almacenar un secreto nuevo
- Cerrar Sesión

Igualmente se resalta que el sistema debe poder ofrecer las capacidades para :

- Una vez se indique que se debe almacenar un nuevo secreto, este debe ser cifrado y fragmentado
- Cada uno de los fragmentos debe ser almacenado y replicado en varios nodos

Arquitectura General de dVault



Como se ha comentado a lo largo del documento, el sistema se apoyará en una arquitectura descentralizada, el cual se caracteriza por ser un sistema de información sin una autoridad central, donde existen múltiples nodos con esta responsabilidad, aportando de forma general al sistema mayor grado de tolerancia a fallos, redundancia de los datos y un gobierno descentralizado.

Se espera entonces que cada nodo cumpla funciones tanto de servidor como de cliente, ofreciendo así acceso a la información almacenados en este y consumiendo los servicios de otros nodos para acceder a la información que este no posea.

Se debe tener presente que dados los requerimientos de seguridad, el sistema se apalancará en la red TOR para permitir que cada nodo pueda ofrecer sus servicios sin estar directamente expuesto a internet y de forma anónima, para ello cada nodo ofrecerá sus capacidades como onion services y podrá ser consumido tanto desde la red TOR como a través de la red loopback.

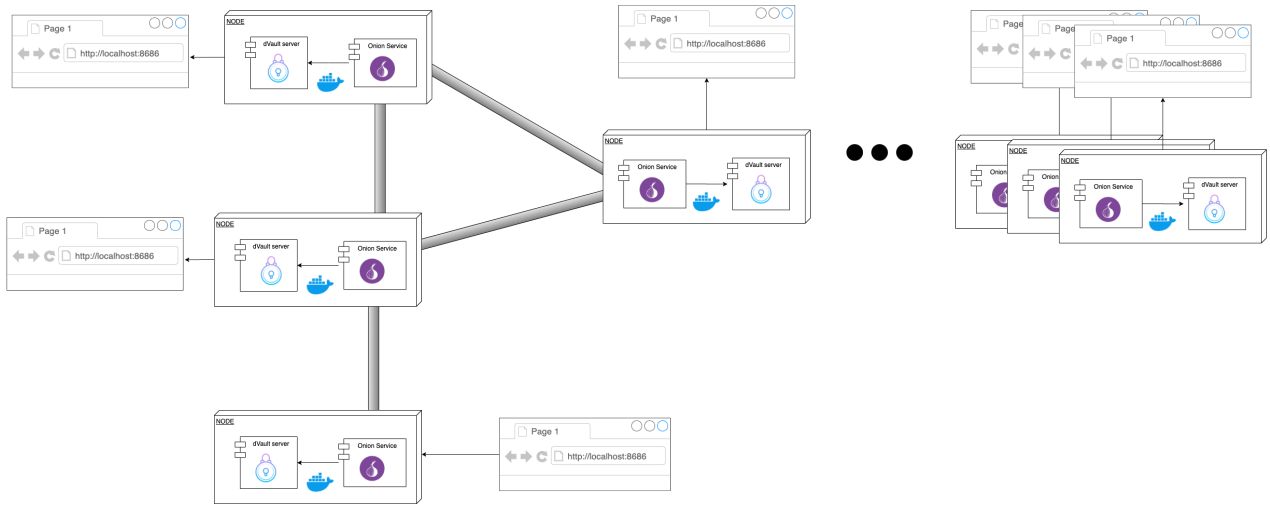


Diagrama de Arquitectura

En el diagrama anterior se visualiza la forma en el cual estarán conectados los diferentes nodos que son parte de la red. Se destaca que la interacción del usuario con el sistema será a través del navegador consumiendo directamente los servicios a través de la red de Loopback, una vez el nodo determine que necesita interactuar con

otros nodos de la red, esta interacción se realizará a través de la red Tor, consumiendo el servicio onion del nodo destino y llegando al mismo mediante una tabla hash distribuida DHT.

¿Por qué esta arquitectura?

Se resalta que la elección de este tipo de arquitectura descentralizada busca solventar los problemas inherentes a la centralización de datos muy sensibles como credenciales y secretos, apalancándose en los siguientes hechos:

- Código Open Source por lo cual los mecanismos de seguridad son públicos y conocidos
- Al no tener control directo sobre el funcionamiento del sistema, no es posible limitar el ingreso a la plataforma a un usuario en específico dado un requerimiento o intención de entidades externas
- No es posible otorgar la información ni comportamiento de los usuarios a agencias de seguridad ni terceros pues no se tiene centralizada la información ni se puede acceder a ella por los mecanismos de cifrado implementados
- Al usar Docker, este facilita la instalación, despliegue, actualización y distribución del sistema

¿Por qué usar Tor?

- Tor evita que los nodos conozcan la ubicación directa de otros nodos que están solicitando la información almacenada en este, de esta manera no es posible relacionar una IP con la información almacenada en un nodo.
- Tor evita ataques dirigidos contra un nodo en específico al evitar que se reciban peticiones desde la red cercana al host, de esta manera se debe conocer el servicio onion de un host en específico para poder llegar a este a través de la red Tor.

A la pregunta, ¿ Cómo saber qué nodo está disponible y qué nodo tiene los datos requeridos ?. Para ello nos apoyaremos en una DHT “Distribuited Hash Table” mediante el algoritmo Kademlia, quien ya resuelve estas necesidades.

Estructura de Datos & Algoritmos

Distribuited Hash Table

Con el objetivo de contar con un sistema de almacenamiento de datos cuya arquitectura esté acorde a la arquitectura elegida para nuestro sistema descentralizado, el sistema se apalancará en una estructura de **Tabla Hash Distribuida - DHT,** el cual permitirá almacenar información del tipo key - value, para ello se utilizará el algoritmo **Kademlia** bajo la implementación de la librería para Nodejs llamada **Kadence**.

La forma en el cual al algoritmo kademlia funciona es mediante la distribución de información en los nodos de computo que hacen parte de la red kademlia, para ello cada nodo contiene 2 tablas, una para almacenar la información en formato key - value y otra para enrutar las peticiones a el nodo adecuado dada una búsqueda o almacenamiento de información en la red.

Se entiende como nodo adecuado para una búsqueda o almacenamiento de un valor, aquel que una vez realizada una operación de distancia entre el hash de la llave y el identificador del nodo, se determine el nodo más cercano al recurso.

010 ⊕ 100 = 110 ➔ 6

110 ⊕ 100 = 010 ➔ 3

De esta manera cada nodo de la red cuenta con un identificador único de 160 bits el cual va a ser comparado con el hash de la llave del recurso que también es del mismo tamaño, y mediante una operación XOR se determina el nodo adecuado para almacenar o redireccionar la búsqueda de un recurso dada cierta distancia entre el recurso y el identificador del nodo.

Se debe tener presente que la tabla de enrutamiento no requiere conocer de todos los nodos de la red, sino que mediante una lista de buckets se almacenan solo un pequeño conjunto de los nodos en cada bucket, siendo actualizados constantemente por cada búsqueda en la red.

Nodo 011		
Bucket 1	{111, 80.100.1.27, 3458} {110, 80.109.1.18, 3454}	{100, 80.105.1.36, 3457} {101, 80.91.14.22, 3460}
Bucket 2	{001, 88.107.1.35, 3444}	{000, 83.102.1.59, 3494}
Bucket 3	{010, 88.107.1.35, 3444}	

Se resalta que la comunicación entre nodos es mediante el protocolo UDP, sin embargo la implementación de Kadence en nodejs permite utilizar diferentes plugins, algunos de ellos orientados a habilitar otros mecanismos de comunicación entre nodos, incluyendo HTTP y Onion Services [23] [24].

Hash & Cifrado

Teniendo en cuenta que parte de la información almacenada en el DHT será información confidencial como contraseñas y metadata de correlación, el sistema usará una función de derivación de clave conocido como PBKDF2 en escenarios específicos del sistema, algoritmo quien ha demostrado protección contra ataques de fuerza bruta, sin embargo se debe tener presente que este no es utilizado para el mecanismo de login del sistema, esto con el objetivo de evitar almacenar el hash en la DHT y evitando así almacenar un valor relacionado directamente con la contraseña de acceso a la plataforma.

En cuanto al mecanismo de cifrado se ha optado por implementar un mecanismo de cifrado simétrico bajo la especificación AES 256, cuyo objetivo en el sistema será cifrar la información almacenada por el usuario en el DHT, y será este último en el que se apalancará el login del sistema, de tal manera que el resultado de iniciar sesión recaerá en la capacidad para descifrar un valor.

Se ha optado por usar un cifrado simétrico en vez de uno asimétrico por 2 razones:

- El cifrado asimétrico no permite cifrar grandes volúmenes de datos
- Se busca que el sistema pueda ser accedido con algo que el usuario conoce y no con algo que el usuario posee, en el caso del cifrado asimétrico seria un par de llaves (Pública y Privada).

Stack Tecnológico

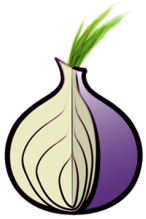
Para la implementación y despliegue del sistema se utilizará:

Nodejs como entorno de ejecución para javascript del lado del servidor, esto teniendo en cuenta la amplia comunidad, la variedad de librerías, la sencillez del lenguaje y su modelo de concurrencia mono-hilo asíncrono **no bloqueante** .

Docker como plataforma para facilitar el despliegue, distribución e instalación del sistema

Kadence como librería nodejs quien implementa el algoritmo Kademlia y permite habilitar el uso de DHT - Distribuited Hash Table

TOR como red de anonimato la cual nos permitirá exponer de servicios del sistema a través de ella

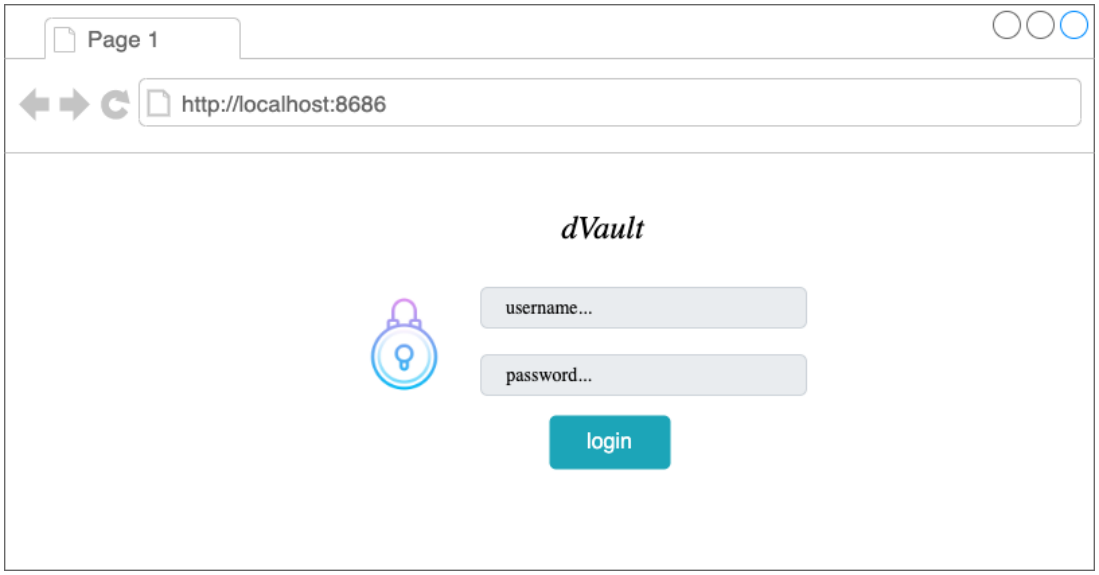


Angular como framework de desarrollo Web

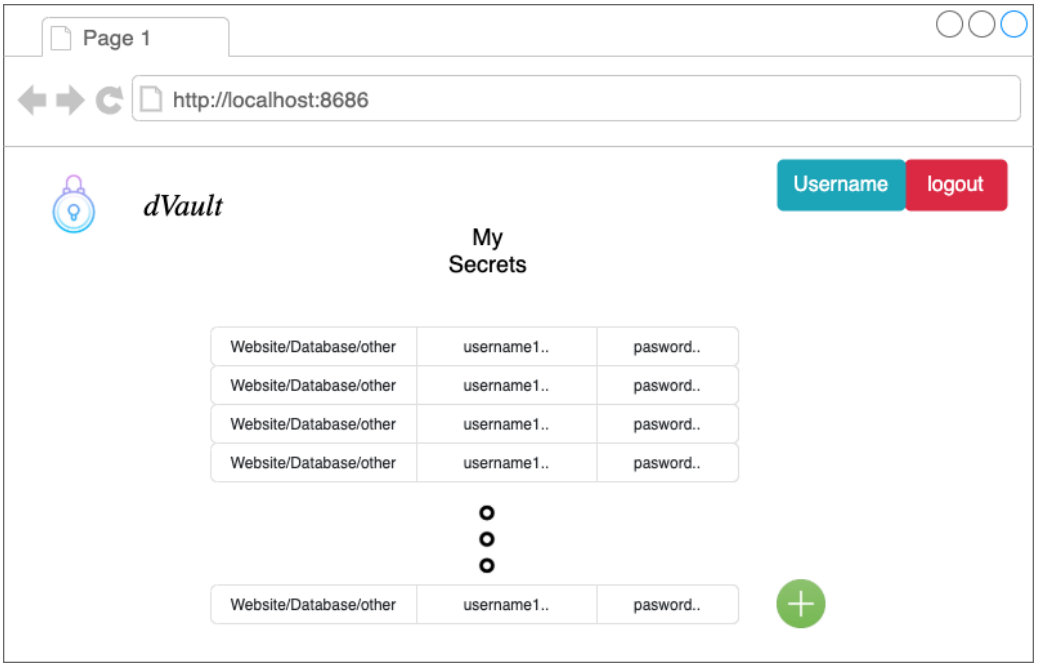
Express como framework nodejs de exposición de servicios REST

Diseño de Interfaces gráficas

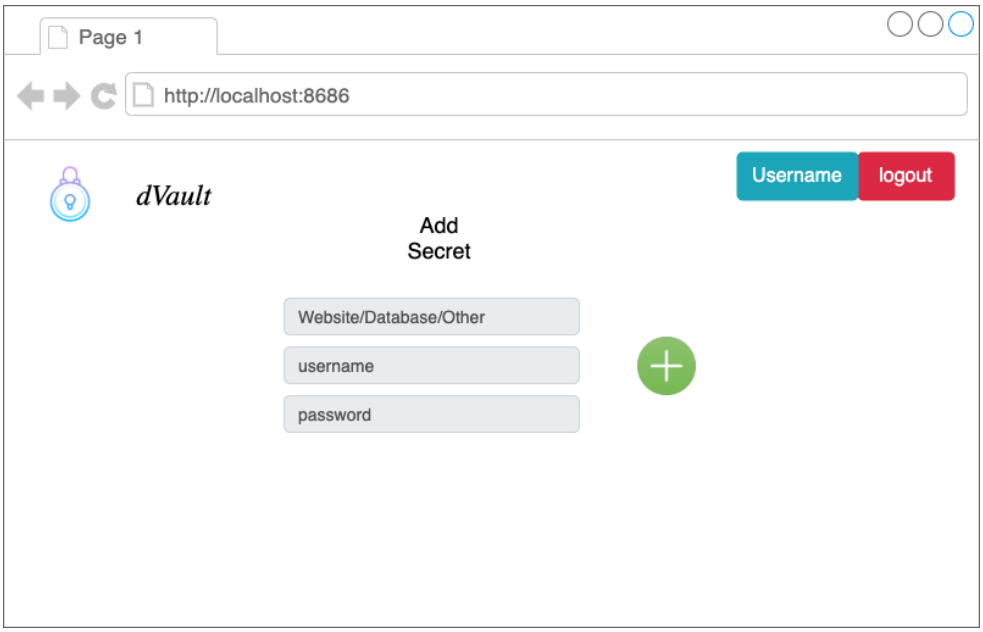
Login Page



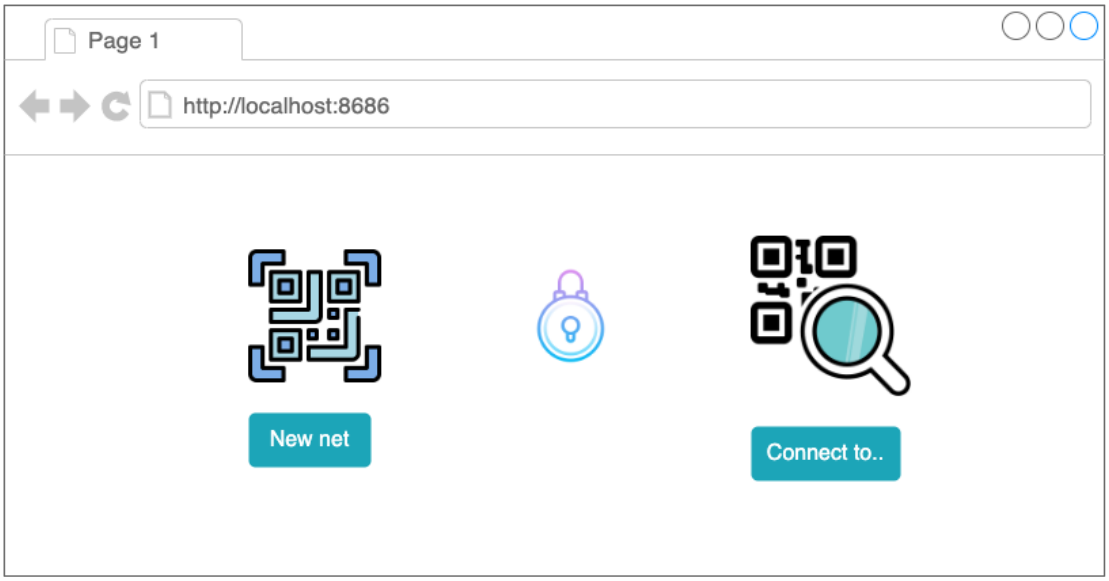
Secrets Page



Add Secret Page



Network Page



IMPLEMENTACIÓN

A continuación se presenta el proceso de implementación de la plataforma el cual he denominado **dVault**, el cual parte de un juego de palabras entre la letra d de **distributed** y **vault** de baúl de secretos.

Este proceso de codificación estuvo caracterizado por la solución de algunos retos técnicos no contemplados al inicio de la planeación del mismo y apalancados por las tecnologías elegidas como la red de anonimato Tor y el hecho de utilizar una tabla hash distribuida DHT como mecanismo de almacenamiento, a continuación se detallan.

Retos abordados

Dentro de las dificultades no contempladas al inicio de la planeación se tuvieron:

Latencia por el uso de la red Tor

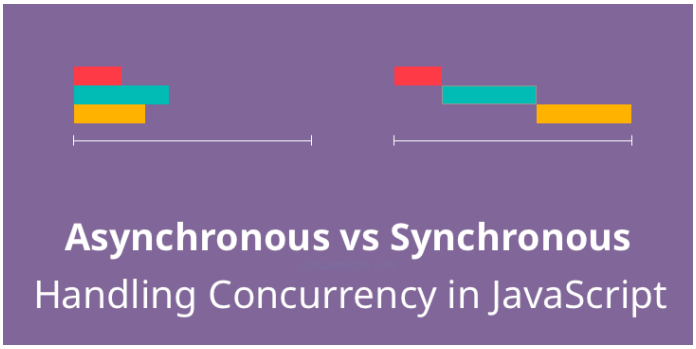
Se debe tener presente que al utilizar la red de anonimato Tor, por la arquitectura y seguridad introducida por la misma, peticiones de red que por lo regular toman menos del segundo en ejecutarse, ahora pueden tomar de 2 a 5 segundos, incluso según la operación realizar se alcanzan tiempos entre 40 y 50 segundos dependiendo de la región del mundo desde donde se ejecuten y la cantidad de operaciones internas a ejecutar, por ejemplo el hecho de consultar los diferentes fragmentos de un secreto.

Para la mayoría de las operaciones, una latencia de 2 a 5 segundos era tolerable, sin embargo la operación de consulta de secretos, por la forma en la cual fue concebida e implementada, la latencia percibida por el usuario se convertía en la latencia acumulada de consultar cada secreto.

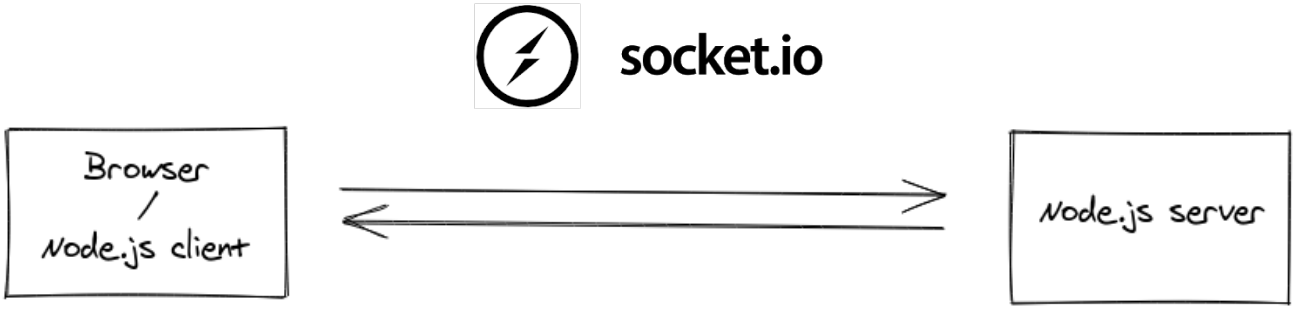
Es así como toma bastante importancia la latencia introducida por Tor en la usabilidad del sistema, por lo cual se debían idear mecanismos que mejoraran la percepción de fluidez del sistema.

Solución

La primera solución y la más sencilla fue aprovechar las capacidades del lenguaje Nodejs para ejecutar peticiones de forma asíncrona, de esta manera pasábamos de tener una sumatoria de latencias a percibir únicamente la mayor latencia de las operaciones ejecutadas, de esta manera se vio beneficiada la operación de obtener los diferentes fragmentos de un secreto, pues el hecho de obtenerlos en paralelo mejoró considerablemente la fluidez del sistema.



El segundo mecanismo que mejoró la percepción de fluidez del sistema fue el hecho de implementar Sockets y la emisión de eventos desde el backend hacia el front end, de esta manera se vio beneficiado la consulta de secretos del usuario, pues teniendo en cuenta que consultar un secreto tomaba alrededor de 2 segundos, no era factible implementar un servicio que se demorara 10, 20 o más segundos para obtener el listado completo de secretos.



Fue así como el usar Sockets.IO permitió emitir eventos desde el servidor nodejs al navegador del usuario para que este desplegara la información del secreto encontrado, de esta manera pasamos de contar un un servicio que tomaba 20 segundos para mostrar la lista completa de secretos a un servicio que emite cada 2 segundos los secretos encontrados.

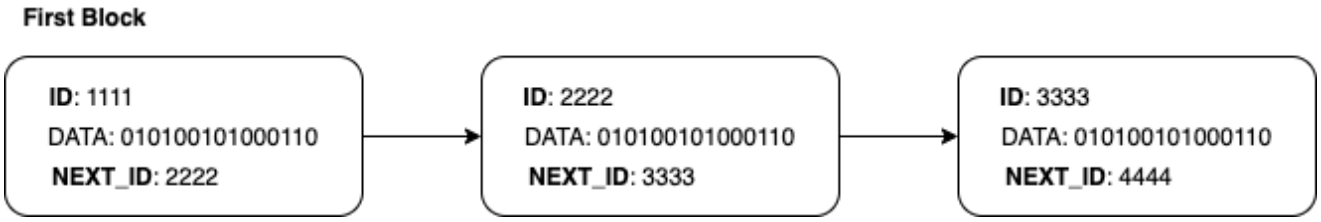
Implicaciones de usar una DHT

Al utilizar una tabla hash distribuida como mecanismo de almacenamiento de información, se debe tener presente que algunas de las tareas que se realizan frecuentemente como actualizar y eliminar un valor, no son fácilmente replicables en este sistema, tareas de correlación de información, como obtener los secretos de un usuario, no son tareas sencillas pues no es factible contar con un documento editadle que indique la ubicación o identificador de los diferentes secretos del usuario.

Solución

Teniendo en cuenta que contamos con mecanismo de almacenamiento que debe ser tratado como un mecanismo inmutable de almacenamiento de información, se optó por implementar un sistema similar al de listas enlazadas donde se cuenta con un conjunto de bloques enlazados.

Una vez registrado un usuario, se almacena el primer manifiesto en la DHT (Primer bloque), con un identificador conocido que será explicado en el mecanismo de login. El manifiesto cuenta con una metadata importante que hace posible encadenar el conjunto de bloques pertenecientes al usuario.



Es así como cada bloque cuenta con un identificador, un contenido cifrado y un identificador del siguiente bloque, de tal manera que para llegar a un bloque, se debe buscar en la DHT por el identificador del mismo, para descifrarlo se debe usar el identificador del anterior bloque (Excepto para el primer bloque que será explicado en el siguiente apartado) y el elemento que permite relacionar los bloques es el “NEXT_ID”.

NEXT_ID es un valor generado cada vez que se genera un nuevo bloque e indica el identificador que deberá tener el siguiente bloque, facilitando así su búsqueda e identificación. Para ello se una un identificador UUID v4 de tal manera que sea muy poco factible que se tenga el mismo identificador para 2 bloques diferentes.

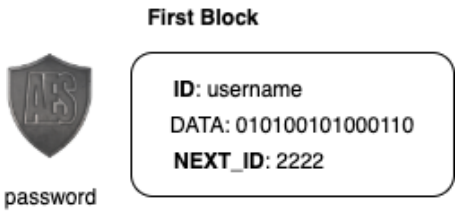
El contenido cifrado del bloque es el listado de identificadores de los fragmentos pertenecientes a un secreto en específico, de esta manera se distribuye el contenido de un secreto en la red y se vela porque un nodo no contiene todos los fragmentos de un secreto.

Se resalta que el contenido almacenado por un nodo siempre está cifrado, y únicamente se tiene en texto plano el identificador y el hash del contenido, estos últimos utilizados para identificar el fragmento o manifiesto y velar porque el contenido no ha sido modificado.

Login

Teniendo en cuenta que para registrar el primer bloque no se puede utilizar un identificador aleatorio, sino que este debe estar relacionado a algo que el usuario conozca y recuerde, se ha optado por usar el nombre de usuario como identificador del primer bloque.

De esta manera tanto al registrarse como al momento de realizar un login, se procede a utilizar el nombre de usuario como identificador del primer bloque y se usa la contraseña del usuario como la llave secreta de cifrado simétrico de este primer bloque.



De esta manera el proceso de login implica tener la capacidad de identificar el bloque adecuado mediante el conocimiento del nombre de usuario, y la capacidad para poder descifrar este bloque mediante la contraseña del usuario.

Se ha optado por este mecanismo de login puesto que:

- No era factible implementar un sistema de almacenamiento mutable y distribuido bajo el esquema de tabla hash distribuida
- Almacenar el hash de la contraseña en la DHT, le permitiría a cualquier individuo conectado a la red conocer el hash de la contraseña de un usuario, de esta manera aún cuando el hash no es la contraseña en sí, sí le permite al atacante obtener un valor cercano y relacionado a la contraseña
- Optar por usar un sistema de login externo implicaría garantizar que el sistema externo es seguro, confiable y descentralizado, de tal manera que no exista un punto de denegación de acceso al sistema por una solicitud de un gobierno, agencia externa o la misma plataforma en si.

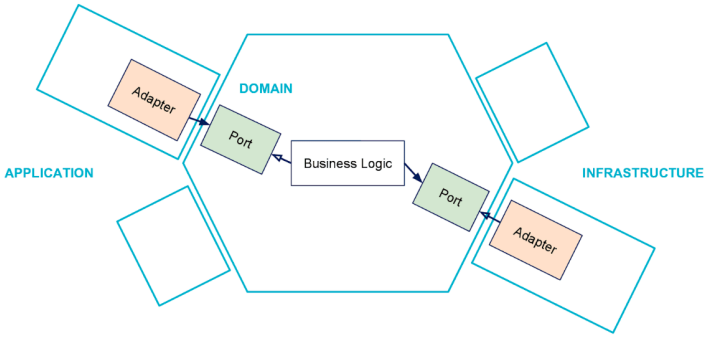
Fragmentación y replicación de los datos

Teniendo en cuenta que uno de los requisitos de la plataforma, es el de velar por el que un secreto no se encuentre en su totalidad almacenado en un nodo, se deben realizar tareas de fragmentación de los datos, antes de ser almacenados y distribuidos en la tabla hash distribuida. Para ello, el contenido del secreto JSON es primero cifrado con AES 256, utilizando como clave el identificador del bloque anterior, excepto para el primer bloque que es la contraseña del usuario.

Posteriormente se obtiene un string correspondiente al cifrado del secreto, el cual es dividido en 70 bytes por defecto, donde cada uno de los fragmentos es distribuido y replicado en la red mediante el algoritmo kademlia, enviando con este el hash del fragmento, esto para garantizar que el fragmento no ha sido modificado.

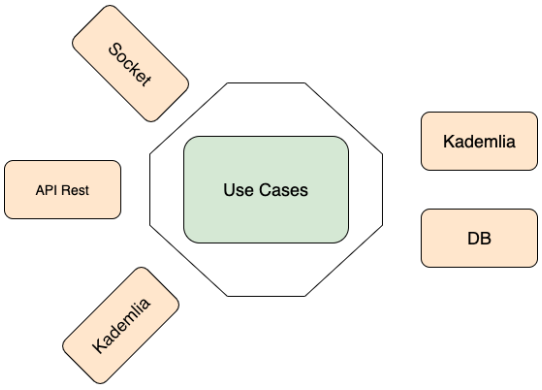
Arquitectura de Software

La implementación estuvo guiada por una separación de responsabilidades lógica basada en arquitectura hexagonal aplicada al lenguaje de Javascript, se debe tener presente que en su aplicación formal se deben aplicar conceptos como inyección e inversión de dependencias, uso de interfaces y otros elementos fácilmente encontrados en lenguajes orientados a objetos pero que no son fácilmente aplicables a Javascript.



Dentro de los beneficios de trabajar bajo una arquitectura hexagonal, se resalta la posibilidad de identificar fácilmente las entradas y salidas del sistemas, y la relación entre estas y las reglas del negocio y casos de uso del mismo.

A continuación se presenta la arquitectura hexagonal de dVault:



Mecanismos de acceso al sistema

API Rest: Mecanismo de interacción para el navegador del usuario, este mecanismo permite realizar los siguientes casos de uso:

- Login
- Registro del usuario
- Almacenar un Secreto
- Eliminar un Secreto
- Conectarse a un nodo de la red

Socket: Mecanismo de interacción para el navegador del usuario, este mecanismo permite realizar el siguiente caso de uso:

- Obtener el conjunto de secretos del usuario de forma fluida y periódica

Kademlia: Mecanismo de interacción entre nodos para el sistema dVault, este mecanismo permite realizar los siguientes casos de uso:

- Almacenar un valor en un nodo externo
- Obtener un valor de un nodo externo

Mecanismos de salida del sistema

Kademlia: Mecanismo de interacción entre nodos para el sistema dVault, este mecanismo permite realizar los siguientes casos de uso:

- Almacenar un valor en un nodo externo
- Obtener un valor de un nodo externo

BD: Mecanismo para almacenar información en el nodo de forma persistente, se almacena:

- Manifiesto
- Fragmentos de secretos

Recursos

A continuación se presentan los repositorios de Github y Docker hub donde se puede encontrar el código fuente utilizado y la respectiva imagen Docker del sistema.

Repositorio de Github

https://github.com/andmagom/dVault

Imagen docker

https://hub.docker.com/r/andmagom/dvault

Instalación del sistema

De acuerdo a los requerimientos de usabilidad, el modo de despliegue, distribución e instalación del sistema será a través de imágenes Docker y ejecución de contenedores Docker. Igualmente se resalta que al ser un proyecto Open Source, existe la posibilidad que el usuario puedan clonar el repositorio, instalar las dependencias y ejecutar el servicio directamente.

La instalación se puede realizar de 2 formas:

NodeJs

Se procede a clonar el repositorio de Github y se instalan las dependencias respectivas, para ello se debe tener NodeJs en su versión 10.

```
$  git clone https://github.com/andmagom/dVault.git
$  cd angular
$  npm build
$  cp -r dist/dvault ../../public
$  cd ..
$  npm install
$  node index.js
```

Docker

El mecanismo recomendado y sencillo de instalación es mediante el uso de la plataforma Docker. Para ello se ejecutan los siguientes comandos docker :

```
$  docker pull andmagom/dvault
$  docker run -v dvault-data:/usr/src/app/data -p 8686:8686
    andmagom/dvault
```

Se resalta del comando anterior, que este crea un volumen llamado dvault-data donde se almacenará los datos del nodo enviados a través de la red kademlia, los cuales pueden ser tanto manifiestos como fragmentos de secretos.

Parámetros del sistema

dVault permite cambiar su funcionamiento con ciertas variables de entorno que pueden ser configuradas. Se presenta a continuación las variables permitidas:

- **SECRET_SPLIT_SIZE_BYTES**: Cantidad de bytes en el cual dividir el secreto. **default**: 70
- **NUMBER_REPLICATIONS**: Número de veces en el cual se replicarán los datos de forma adicional al ya realizado por kademlia. **default**: 4
- **COOKIE_SECRET**: Secreto para cifrar la información almacenada en el navegador. **default**: dVault
- **TOR_FOLDER**: Ruta donde se almacenará la configuración de Tor. **default**: /usr/src/app/data/hidden_service
- **KADENCE_REFRESH**: Tiempo en el cual cada nodo realizará una validación de la tabla de enrutamiento. **default**: 300000 ms

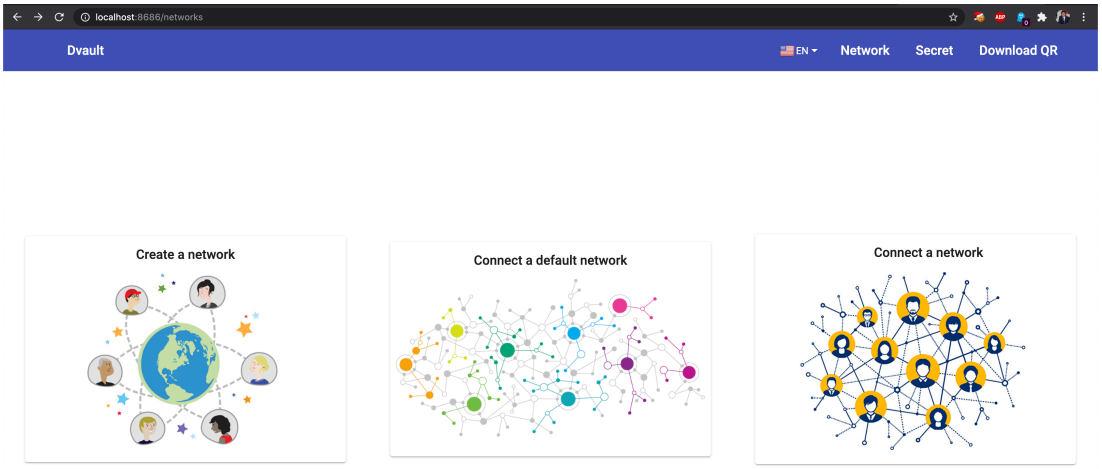
Funcionamiento de dVault

Una vez el aplicativo se encuentra en ejecución, este adquiere automáticamente una identidad kademlia y se crea el servicio onion Tor respectivo para ese nodo, el cual le permitirá comunicarse de forma anónima y segura con otros nodos de la red. Cuando se inicia el servicio por primera vez, se aprecia al final del log los valores anteriormente comentados.

```
info: querying tor for socks proxy port {"service":"kadence","timestamp":"2020-11-24T15:46:53.158Z"}
info: Node Identity: 6d37f587315815617501 93d8b6a4753adbb7cdf3 {"service":"kadence","timestamp":"2020-11-24T15:46:53.165Z"}
info: node listening on local port 1337 and exposed at undefined//rxndkbo5liuy44rhoioz2mrlrrbk5uhv5xt6ezapdsas33j6qle2ytad.onion:443 {"service":"kaden
ce","timestamp":"2020-11-24T15:46:53.165Z"}
```

Procedemos a ingresar a la página web por defecto bajo la url http://localhost:8686:

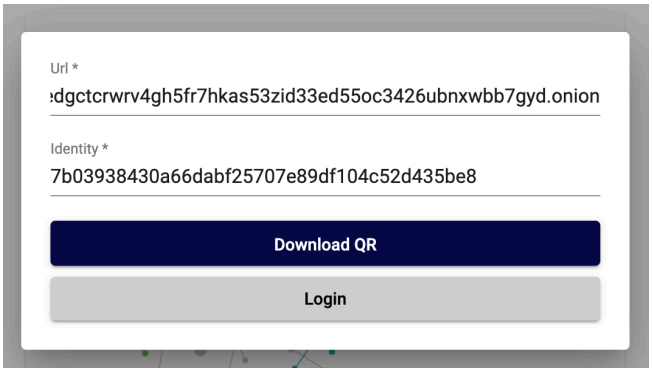
Página principal



Como se aprecia anteriormente, esta comunicación es únicamente a través de la red localhost, nunca sale del nodo si no es por medio de la red Tor, el cual lo hace a través de un canal de comunicación cifrado y anónimo.

El usuario tiene la posibilidad de crear una red kademlia, de conectarse a una existente o de conectarse a un nodo semilla por defecto. Según la elección elegida así mismo se presenta un popup que le permite descargar los datos de conexión para el nodo actual, el cual puede compartir a otros usuarios para crear una red de nodos, o puede ingresar los datos de conexión para conectarse a una red kademlia a partir de un nodo existente o simplemente conectarse a una red por defecto.

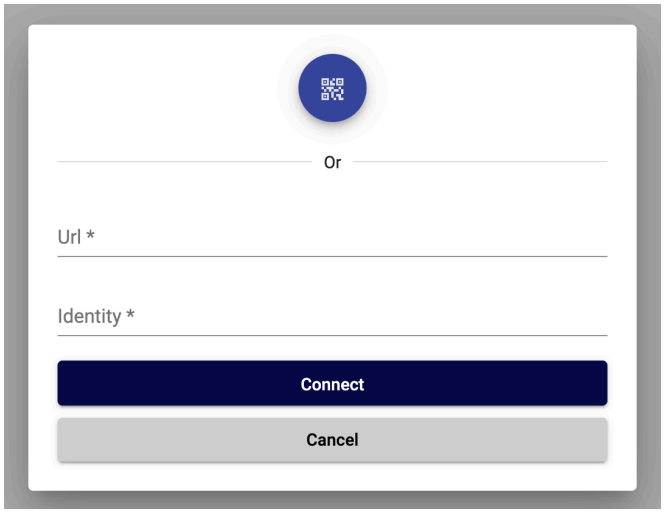
Popup al crear una red



Este popup le permite al usuario:

- Obtener los datos de conexión, identidad y url del servicio Tor del nodo actual, los cuales pueden ser compartidos con otros usuarios
- Descargar la información de conexión en un formato QR
- Ir a la página de login de dvault. **Nota**: Se debe tener presente que si en la red kademlia solo existe un nodo (el nodo actual), el sistema no se comporta de forma ideal y no se puede garantizar que la información se pueda acceder, pues no existen otros nodos con los cuales se pueda replicar la información.

Popup al conectarse una red



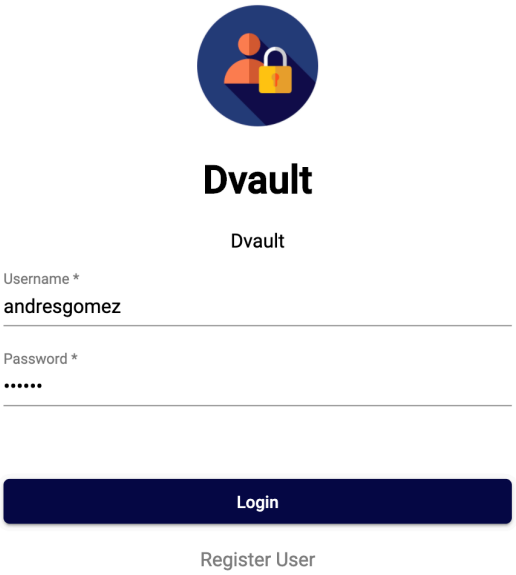
Este popup le permite al usuario ingresar los datos de conexión de un nodo kademlia, para ello se debe ingresar la identidad y la url del servicio Tor de este. Se resalta que este valor puede ser obtenido del popup anterior o ingresado mediante una imagen QR, esto con el objetivo de facilitar la usabilidad del sistema.

Una vez el nodo del usuario se conecta al nodo externo indicado, el algoritmo kademlia intenta conectarse a otros nodos conocidos por el nodo externo, de esta manera se crea una tabla de enrutamiento de nodos cercanos para el nodo del usuario.

Página de login y Registro

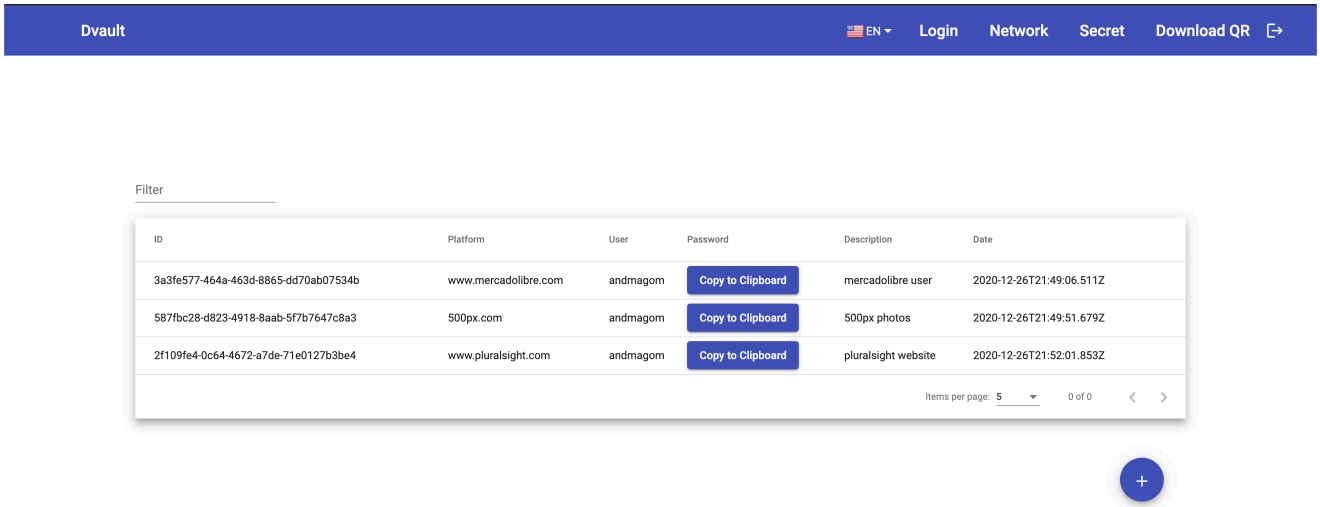
Mediante esta página se pueden realizar el login en dVault y el registro de un nuevo usuario en el sistema. Se debe tener presente que:

- El mecanismo de registro crea el primer bloque en la DHT con un identificador relacionado al nombre de usuario y cifrando el bloque con la contraseña del usuario.
- El mecanismo de login implica buscar el bloque con el identificador relacionado al nombre de usuario y la capacidad de poder descifrar ese primer bloque mediante la contraseña del usuario.



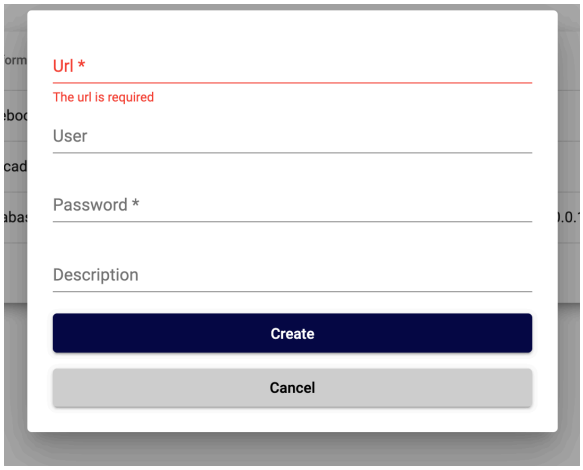
Página de secretos

Una vez se ingresan correctamente las credenciales, el usuario es automáticamente redirigido a la página de secretos, en el cual se cargan de forma asíncrona los secretos una vez son encontrados en la DHT, para ello internamente se abre un canal mediante sockets donde el backend emite eventos cada cierto tiempo.



Popup Secretos

Una vez el usuario decide agregar un nuevo secreto, a este se le despliega un popup solicitándole la información respectiva del secreto a almacenar.



CONCLUSIONES

En relación con el objetivo inicial, el cual contemplaba implementar una plataforma alternativa para la gestión de credenciales de forma descentralizada y apalancada sobre la red Tor, se considera que el fruto de este trabajo, que he denominado dVault, a permitido ofrecer una versión preliminar que satisface las necesidades básicas de este tipo de plataformas.

Se resalta que dVault permite:

- Almacenar información de forma descentralizada y distribuida.
- Interacción entre nodos de forma anónima mediante la utilización de la red Tor y del uso de servicios onion.
- Distribución y replicación de información cifrada entre nodos, fragmentando cada dato y distribuyéndola entre los nodos, velando que un nodo no posea la totalidad de los fragmentos para un dato.

Igualmente no resta recordar que a lo largo del trabajo, se abordaron los restantes objetivos específicos que permitieron elaborar esta plataforma, los cuales involucraban:

- Entender el funcionamiento de arquitecturas centralizadas y descentralizadas desde una perspectiva técnica como funcional, resaltando las implicaciones al nivel de privacidad/disponibilidad de los datos y las limitantes de cada arquitectura.
- Analizar el funcionamiento de la red Tor, resaltando sus beneficios y consideraciones al implementar aplicaciones bajo esta red, exponiendo el funcionamiento de los servicios onion, los cuales son fundamentales en dVault para poder que los nodos interactúen entre si de forma anónima.
- Elegir un mecanismos de cifrado adecuado para el sistema, contemplado tanto algoritmos simétricos como asimétricos.

Finalmente se resalta que dVault es una versión preliminar, y que como todo sistema, debe atravesar primero por un proceso de madurez antes de ser utilizado en un entorno productivo, que permita aplicar mejoras a la plataforma en forma general, desde resolución de posibles errores, identificación de problemas de seguridad, mejoras de rendimiento o de usabilidad. Es por ello que se recomienda usar la plataforma únicamente como referencia de implementación o para pruebas de concepto, mientras el tiempo y la comunidad permiten ofrecer una plataforma más madura.

Lineas de trabajo futuro

Durante la elaboración del trabajo y de la plataforma dVault, surgieron diferentes cuestiones e implicaciones de importancia sobre la elección de las tecnologías y arquitecturas utilizadas. De las diferentes implicaciones que se contemplaron, hubo una que resaltó por su impacto negativo para un sistema de gestión de credenciales descentralizado de poderse materializar y es por ello que esta debería ser parte de trabajos futuros.

Se debe tener presente que al utilizar una arquitectura descentralizada, esta se debe apoyar en mecanismos de replicación de información para poder garantizar el acceso y disponibilidad de los datos. Datos que deben ser almacenados en los nodos pertenecientes a la red, y que pueden accedidos por los propietarios de los nodos, es por ello que mecanismos de cifrado son fundamentales al utilizar estas arquitecturas descentralizadas.

Teniendo en cuenta que la información tratada en un sistema de gestión de credenciales es totalmente crítica y que los mecanismos de seguridad deben poder garantizar su seguridad no solo en la actualidad sino para el futuro cercano, **se deben considerar las implicaciones de almacenar información a largo plazo en sistemas inmutables y descentralizados, tomando principal interés la evolución de las capacidades de computo cuántico, que podrían eventualmente invalidar cualquier mecanismo de cifrado** y por lo tanto vulnerar la privacidad de los usuarios que depositaron información crítica como secretos y credenciales en estos sistemas descentralizados.

BIBLIOGRAFÍA

[1]
Electronic Privacy Information Center, «Equifax Data Breach», *Equifax Data Breach*, sep. 25, 2020. https://www.epic.org/privacy/data-breach/equifax/.

[2]
Jeffrey A. Meldman, Centralized Information Systems and the Legal Right to Privacy, 52 Marq. L. Rev. 335 (1969). Available at: http://scholarship.law.marquette.edu/mulr/vol52/iss3/1

[3]
«encrypt-my-data», *Github*, sep. 26, 2020. https://github.com/STYJ/encrypt-my-data.

[4]
«Savana - Decentralized 1password alternative», *Savana - Decentralized 1password alternative*. https://github.com/savana-app/savana.

[5]
«BlockVault», *BlockVault*, sep. 27, 2020. https://blockvault.site.

[6]
Suzanne Frey. "Notifying Administrators about Unhashed Password Storage." Notifying administrators about unhashed password storage, May 21, 2019. https://cloud.google.com/blog/products/g-suite/notifying-administrators-about-unhashed-password-storage.

[7]
Facebook, "Keeping Passwords Secure," *Keeping Passwords Secure*. https://about.fb.com/news/2019/03/keeping-passwords-secure/ (accessed Oct. 17, 2020).

[8]
Parag Agrawal, "Keeping your account secure," *Keeping your account secure*. https://blog.twitter.com/official/en_us/topics/company/2018/keeping-your-account-secure.html.

[9]
Forbes, "Apple Helps FBI Track Down George Floyd Protester Accused Of Firebombing Cop Cars," *Apple Helps FBI Track Down George Floyd Protester Accused Of Firebombing Cop Cars*. https://www.forbes.com/sites/thomasbrewster/2020/09/16/apple-helps-fbi-track-down-george-floyd-protester-accused-of-firebombing-cop-cars/#1900bc4f5901 (accessed Oct. 17, 2020).

[10]
Google Cloud, "Exponential growth in DDoS attack volumes," *Exponential growth in DDoS attack volumes*. https://cloud.google.com/blog/products/identity-security/identifying-and-protecting-against-the-largest-ddos-attacks (accessed Oct. 17, 2020).

[11]
Bloomberg, "Amazon's Alexa Team Can Access Users' Home Addresses," *Amazon's Alexa Team Can Access Users' Home Addresses*. https://www.bloomberg.com/news/articles/2019-04-24/amazon-s-alexa-reviewers-can-access-customers-home-addresses (accessed Oct. 17, 2020).

[12]
informationisbeautiful, "World's Biggest Data Breaches & Hacks," *World's Biggest Data Breaches & Hacks*. https://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/ (accessed Oct. 17, 2020).

[13]
theintercept, "Facebook and Twitter Cross a Line Far More Dangerous Than What They Censor," *Facebook and Twitter Cross a Line Far More Dangerous Than What They Censor*. https://theintercept.com/2020/10/15/facebook-and-twitter-cross-a-line-far-more-dangerous-than-what-they-censor/ (accessed Oct. 17, 2020).

[14]
BBC, "Megaupload file-sharing site shut down," *Megaupload file-sharing site shut down*. https://www.bbc.com/news/technology-16642369 (accessed Oct. 17, 2020).

[15]
"Internet censorship in the Arab Spring." https://en.wikipedia.org/wiki/Internet_censorship_in_the_Arab_Spring (accessed Oct. 17, 2020).

[16]
BBVA, "BBVA teams up with MIT to improve card fraud detection." https://www.bbva.com/en/bbva-teams-up-with-mit-to-improve-card-fraud-detection/ (accessed Oct. 17, 2020).

[17]
IMF, "The Truth about the Dark Web," *The Truth about the Dark Web*. https://www.imf.org/external/pubs/ft/fandd/2019/09/the-truth-about-the-dark-web-kumar.htm (accessed Oct. 18, 2020).

[18]
statista, "Hours of video uploaded to YouTube every minute as of May 2019," *Hours of video uploaded to YouTube every minute as of May 2019*. https://www.statista.com/statistics/259477/hours-of-video-uploaded-to-youtube-every-minute/#:~:text=As%20of%20May%202019%2C%20more,for%20online%20video%20has%20grown.

[19]
Who Is Satoshi Nakamoto?, Lemieux, Pierre.Regulation; Washington Vol. 36, Iss. 3, (Fall 2013): 14-15.

[20]

dailymail.co.uk, “China’s disappeared: At least one is dead and the rest haven’t been heard from in months, so why isn’t the world asking what happened to the brave souls who dared to speak up about the coronavirus outbreak after Beijing lied to the world*?* https://www.dailymail.co.uk/news/article-8233203/Chinas-disappeared-happened-dared-speak-coronavirus.html (accessed Oct. 20, 2020).

[21]
theguardian.com, “NSA Files Decoded.” https://www.theguardian.com/world/interactive/2013/nov/01/snowden-nsa-files-surveillance-revelations-decoded#section/1 (accessed Oct. 20, 2020).

[22]
Torproject, “Which Tor node knows what?,” *Which Tor node knows what?* https://trac.torproject.org/projects/tor/wiki/doc/TorFAQ#Overviewastable (accessed Oct. 21, 2020).

[23]
“Plugins,” *https://gitlab.com/deadcanaries/kadence/-/blob/master/doc/plugins.md*. .

[24]
“Transport Adapters.” https://gitlab.com/deadcanaries/kadence/-/blob/master/doc/transport-adapters.md.

[25]
“Key stretching,” Key stretching. [Online]. Available: https://en.wikipedia.org/wiki Key_stretching. (accessed Oct. 21, 2020).

[26]
“PBKDF2,” https://en.wikipedia.org/wiki/PBKDF2. (accessed Oct. 21, 2020).

[27]
“Salt (cryptography),” Salt (cryptography). [Online]. Available: https://en.wikipedia.org/wiki/Salt_(cryptography). (accessed Oct. 21, 2020).

[28]
“Pepper (cryptography),” Pepper (cryptography). [Online]. Available: https://en.wikipedia.org/wiki/Pepper_(cryptography). (accessed Oct. 21, 2020).

[29]
“Password Hashing: add salt + pepper or is salt enough?,” Password Hashing: add salt + pepper or is salt enough? [Online]. Available: https://security.stackexchange.com/questions/3272/passwordhashing-add-salt-pepper-or-is-salt-enough. (accessed Oct. 21, 2020).