Vol.2018-MBL-89 No.6
Vol.2018-ITS-75 No.6
2018/11/15

標的型攻撃対策のためのブロックチェーン技術を用いたホワイトリスト方式防御システムの実現性に関する検討

中鉢 かける[1]　中村 嘉隆[1]　稲村 浩[1]

概要：近年，サイバー攻撃の一種である標的型攻撃が脅威を増し，その被害が増大しつつある．これに対し，標的型攻撃への対策を目的とした様々な手法が提案されているものの，攻撃への防御精度の高い手法であっても，既存業務の作業効率の低下などの悪影響がある場合は実際に使用されるまでに至らないという課題がある．また，既存業務へ悪影響を与えない場合でも，高精度の攻撃防御に必要な情報を単一組織のみが収集することには限界があるため，複数組織が連携して情報を共有する必要性がある．複数組織間での情報共有をセキュアに行うために有用な技術として近年注目されているブロックチェーン技術が挙げられる．本研究では，標的型攻撃に関するホワイトリストを複数組織間で連携して，ブロックチェーン技術によって共有する攻撃防御基盤の実現性に関数する検討を行い，その有用性を示す．

キーワード：ブロックチェーン，ホワイトリスト，標的型攻撃

Kakeru Nakabachi[1]　Yoshitaka Nakamura[1]　Hiroshi Inamura[1]

1. はじめに

近年，標的型攻撃の被害が増加し，脅威を増している [1]．標的型攻撃とは，攻撃対象をある特定の組織に絞ってマルウェアが添付されたメールなどを送りつけ，組織内の端末をマルウェア感染させて機密情報の窃取などを試みるサイバー攻撃の一種である．標的型攻撃の典型的な流れを図 1 に示す．攻撃者は標的組織に普段の業務と見分けがつかないようなマルウェア添付メールを送りつけ，標的組織内のユーザに開かせる．マルウェア添付メールを開封してしまった場合，組織内がマルウェア感染し，攻撃者のマルウェアの遠隔操作による機密情報の収集が行われる．最後に攻撃者サーバへの機密情報のアップロードが行われて窃取される．

近年の標的型攻撃の段階としてサイバーキルチェーン [2] という 7 つの段階があることが知られている．その中の「偵察」「武器化」「配送」「エクスプロイト」「インストール」「遠隔操作」「目的実行」のうち，実際に情報窃取を行う目的実行までのどの段階で攻撃を検知して防御や遮断ができるかが課題となっている．標的型攻撃の出口対策としては主にブラックリスト方式とホワイトリスト方式の 2 つが挙げられる．ブラックリスト方式を用いた場合は，ブラックリストに載っているアクセス先への通信のみを遮断するため，未知のマルウェアによるアクセスや攻撃者サーバの頻繁な変更に追従が困難であるという問題がある．ホワイトリスト方式を用いた場合，未知のマルウェアによるアクセスや，攻撃者サーバの頻繁な変更にも対処可能であるが，ホワイトリストに載っていないアクセスは全て遮断してしまうため，業務に必要な通信に関する利便性が低下してしまう問題がある．

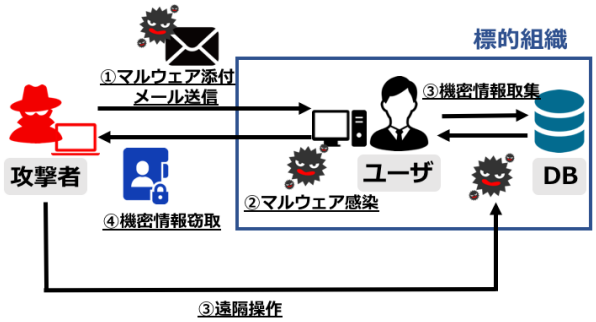

図 1　標的型攻撃の典型的な流れ

攻撃者サーバの頻繁な変更を用いるなど，近年巧妙化す

[1] 公立はこだて未来大学システム情報科学部
School of Systems Information Science, Future University Hakodate

る標的型攻撃にはホワイトリストによる防御が有効であると考えられる．このとき，通信を用いる通常業務が発生するたびにホワイトリストを拡張する必要があるが，単一組織では発生頻度の小さい業務も多く，ホワイトリストに登録されていない業務が発生するたびにホワイトリストへの登録作業を行うことは，通常業務への妨げになると考えられる．

そこで，複数の組織間で連携してホワイトリストを生成・共有する手法を検討する．その際 1 つの組織がホワイトリストの管理や更新をした場合，組織全体でシステムを運用するコストが偏ってしまったり，生成されたホワイトリストの内容を後で特別な権限を行使して改竄してしまうなどのセキュリティリスクが生じる可能がある．また，ホワイトリストに更新を行うロジックも透明性を持たない．

本研究では，近年注目されているブロックチェーン技術を用いて複数の組織が連携してホワイトリストを生成し，高い攻撃遮断精度と既存業務への悪影響を最小限に抑える標的型攻撃遮断システムの実現を目的とする．ブロックチェーン技術を用いることにより，複数組織間でシステムを運用する際のコストを均一にさせたり，共有しているホワイトリストを更新する際のロジックを全組織間で統一し，特定の組織が特別な権限を持って更新してしまうリスクを排除できる．

2. ブロックチェーン技術

2.1 ブロックチェーン技術の概要

ブロックチェーン技術とは，Satoshi Nakamoto 氏によって考案された Bitcoin[3] において銀行などの第三者機関を排除し，ネットワーク参加者同士で直接送金を可能にした分散型台帳と呼ばれる分散システム技術の一種である．現在，主に仮想通貨の実装に用いられている．近年はブロックチェーン技術における資源の移動記録を改竄が困難な台帳に全て記録するという特徴を用い，資源のトレーサビリティを保証するサービス [4] や，第三者機関を排除することができる事による従来型分散システムのコスト削減など，注目を浴びている．

ブロックチェーン上では基本的にユーザが他のユーザに送金を行う処理が 1 つのトランザクションとして発行される．ブロックチェーンネットワーク上で発行された大量のトランザクションは 1 つのブロックという単位にまとめて保存され，ネットワーク参加者全員に共有される．ブロックとチェーンの基本的な関係を図 2 に示す．各ブロックが大量のトランザクションに加え，前のブロックのハッシュ値を含みチェーン状に時系列的に生成されていく．この構造により，1 つのブロックを改竄しようとするとそれ以降のブロックも全て改竄する必要があるため改竄が困難であるという特徴を持っている．Bitcoin では，互いに信頼のない参加者同士でネットワークが構築されるため，全員が正

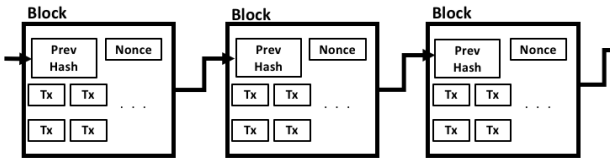

図 2 ブロックとチェーンの関係

しい合意をするために PoW（Proof of Work）と呼ばれる合意形成アルゴリズムが用いられる．PoW ではブロックを生成するために大量のマシンパワーを消費してあるハッシュ値よりも小さくなるナンス（Nonce）と呼ばれる値を発見できるまでハッシュ計算を繰り返すマイニングという作業が合意形成として行われる．

2.1.1 ブロックチェーンのネットワーク構成の種類

現在では数多くのブロックチェーンプラットフォームの開発が活発に進んでいる．ブロックチェーンのネットワーク構成は，パブリック型，プライベート/コンソーシアム型の 2 つに分けることができる．

(1) パブリック型：パブリック型のブロックチェーンは，ネットワークへのノードの参加や離脱が自由であり，互いに信頼のないノードも含んで構成されるという特徴を持っている．そのため，悪意のあるノードが参加する可能性がある．このような状況下でも参加ノード間で正しいデータをブロックチェーンに保存するための合意形成アルゴリズムとして PoW が用いられている．

(2) プライベート/コンソーシアム型：プライベート型のネットワークで構成されるブロックチェーンは互いに信頼のある限定的なノードで構成される．そのため，PoW のような合意形成アルゴリズムは必要なく，トランザクションの実行速度が早いという特徴がある．特定の単一組織のみで構成されるブロックチェーンネットワークをプライベート型と呼び，特定の複数組織で構成されるブロックチェーンネットワークをコンソーシアム型と呼ぶ．

2.1.2 ブロックチェーンプラットフォームの種類

ブロックチェーンプラットフォームとして分散システム開発用やビジネス用途など数々のものが存在し，多くが OSS（Open Source Software）として公開されている．現在，仮想通貨で最も主要なブロックチェーンプラットフォームである Bitcoin も OSS である．

また，分散システム開発向けブロックチェーンプラットフォームとして Ethereum[5] が登場し，現在も開発が続けられている．Ethereum はパブリック型ブロックチェーンに分類される．また，Bitcoin とは異なる合意形成アルゴリズムが用いられており，プライベート型プラットフォームの分散システムの開発も可能になっている．また Ethereum の特徴として，ブロックチェーン上にトランザクションを

保存する以外にスマートコントラクトという業務などで決まっている契約をプログラムとしてブロックチェーン上に保存し，その契約を自動実行できるという機能がある．

その他にも The Linux Foundation[6] が主催し，Hyperledger Project[7] によって開発されている Fabric[8] がある．Fabric はコンソーシアム型に分類され，ビジネス環境でもシステム構築のプラットフォームとして開発できるように機密性と拡張容易性の機能をサポートしている．Fabric も Ethereum と同じようにネットワーク参加者が同意したビジネスロジックをスマートコントラクト（Fabric ではチェーンコードと呼ばれる）を実行することで State DB と呼ばれる KVS（key-value store）のデータベースを操作する．

2.2 本研究で扱うブロックチェーンプラットフォームの選定

近年 Fabric を用いたクラウドソースドラーニングシステムの開発 [9] や Fabric の性能評価 [10] に関する研究は盛んに進められている．Fabric を用いた応用事例としてロンドンのスタートアップ企業 Everledger 社 [4] によるダイアモンドやアートなどの取引履歴を保存することによって，資産の品質保証を行うシステムが有名である．このように Fabric は活発に開発や研究，応用がされており，今後様々な分野で主流となって適用されることが予想されることから本研究では Fabric を用いて提案システムの実装を行う．

2.2.1 Hyperledger Fabric

Hyperledger Fabric（Fabric）では，State DB と呼ばれるブロックチェーン上の最新の状態が保存されている KVS のデータベースを操作するためにトランザクションを発行する．トランザクションはチェーンコードと呼ばれる Ethereum のスマートコントラクトの一種であるようなブロックチェーンネットワークに参加しているノードが同意した不変なビジネスロジックが記載されたプログラムを実行する．トランザクションには State DB の参照を行う Query トランザクションと，更新を行う Invoke トランザクションの 2 種類存在する．Fabric では，このチェーンコードを呼び出すためにトランザクションが発行される．

2.2.2 Fabric のアーキテクチャ

例として，組織 4 で構成された Fabric の基本的なコンポーネントと構成を図 3 に示す [11][12].

Fabric は主に以下のコンポーネントから構成される．

- Peer（ピア）
  組織内の Fabric 上でのノードを表す．それぞれのピアは分散台帳とチェーンコードを保持している．ピアはエンドーサーとしての役割を持ち，ユーザから発行されたトランザクションを仮実行して結果が正しいか確認することでエンドースメント（同意）を行ってトランザクションに署名し，ユーザに返却する．また，ピアはコミッターとしての役割を持っており，自身の分散台帳に含まれるブロックチェーンと State DB の更新も行う．
- Chaincode（チェーンコード）
  トランザクションによって呼び出されるスマートコントラクトである．チェーンコードによって State DB へ値の参照・更新やブロックチェーン内の過去の State DB への更新履歴を参照することも可能である．
- State DB
  トランザクションが呼び出したチェーンコードにより更新された最新の状態が保存される．Fabric を用いて Bitcoin などのようなシステムを実装した際は各ユーザの最新の残高などが保存される．
- Fabric Client SDK
  ユーザに用意された SDK(Software Development Kit) であり，Fabric のピアにチェーンコードをインストールしたりチェーンコードを実行するなどの機能を利用するための API である．
- Orderer（オーダラー）
  ピアによってエンドースメントされたトランザクションを受け取り，ブロックチェーンと State DB に結果を反映する際の順序の制御を行い，ブロックを生成してピアにブロードキャストするノードである．
- Blockchain（ブロックチェーン）
  Fabric のブロックチェーン上にはチェーンコードによって State DB に更新された値の履歴が全て書き込まれている．チェーンコードにより State DB に更新された履歴を全て取得することも可能である．
- MSP(Membership Service Provider)
  CA（Certification Authority）や外部の CA と連携をしてユーザに証明書を発行する役割を担う．ユーザはこの証明書を用いてトランザクションを認証し，ピアはエンドースメントする．

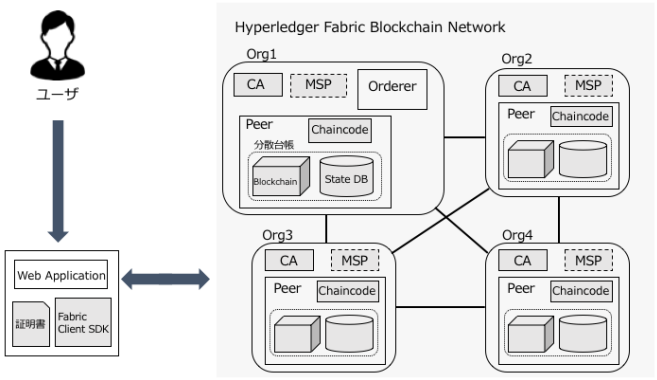

図 3 Fabric の基本的な構成

3. 関連研究

標的型攻撃防御のための研究は既に数多く行われている．特に，自律進化型防御システム（AED:Autonomous Evolution of Defence）に関する研究が活発にされている [13][14][15]．これらの既存研究では，基本的に組織内のユーザが悪意のある危険な悪性サイトにアクセスしようとしていると判断された場合に，マルウェアには突破が困難である CAPTCHA[16] 認証を用いた追加認証をユーザに要求している．そうすることで，人間による意図的な外部サイトへのアクセスを許可する．

角田らは，提案した分析装置を用いて算出されたサイトの悪性度に基づき，安全性の判断が困難な外部サイトをグレーリストに振り分け，このグレーリストに含まれるサイトに対して人手による追加認証を行う手法を提案している [13]．これにより，悪性サイトへのアクセスを遮断することが可能となっている．しかし，この手法は悪性度の算出ロジックが静的であるため，未知のマルウェアへの追従が困難である．

仲小路らは，予めマルウェア動的解析システムによる判断結果をグレーリストとして登録させた上で，危険または，安全と判断された外部サイトに共通する属性を学習し，未知のサイトへのアクセス発生時に，サイトの安全性を予測し，対処可能にする不審属性学習機能を含む AED を提案している [14]．これにより追加認証の必要なサイトを限定することで，ユーザの負担の小さい防御を可能としている．しかし，本来遮断すべき悪性サイトについても，50%の割合で検知漏れが発生し，アクセスできてしまう問題がある．

重本らは，ホワイトリストのみを用い，ホワイトリストに登録されていない外部サイトへのアクセスに対してのみ CAPTCHA による追加認証をユーザに要求することで，マルウェアによるアクセスの完全な遮断を実現している [15]．ホワイトリストのみを用いることで，ホワイトリストに登録されていない外部サイトは全て遮断するため，未知のマルウェアによるアクセスも遮断可能である．また，ユーザへの追加認証の要求率を下げるため，組織内である一定数のユーザが特定の外部サイトに対して追加認証を成功していた場合に限り，このサイトをホワイトリストに登録して提案システムによる業務への悪影響を削減している．しかしこの方式では，システムを利用するユーザ数を 1000 人程度確保できないとユーザへの追加認証要求率が高くなってしまい，既存業務にも影響を及ぼす．

4. アプローチ

既存のブラックリストやグレーリストを用いた手法では，既知のマルウェアによる悪性通信しか遮断できず，日々変化するマルウェアに追従するのは困難である．またマルウェア動的解析システムによる判断結果から未知のマルウェアによる悪性通信を予測する手法は，予測精度に問題がある．そこで本研究では，悪性サイトへのアクセス遮断精度が高く未知のマルウェアにも対処可能な重本らが提案する AED をもとに，システムの利用ユーザ数の問題を解決するための手法を検討する．

まず，システムを複数組織が共同で利用することで，複数組織間の連携により，全体で 1 つのホワイトリストを生成・共有する手法を考える．複数組織間でホワイトリストを生成・共有する場合は，特定の組織が特別な権限を持ってホワイトリストを改竄しないこと，複数組織間でシステムを運用するコストを均一にすること，ホワイトリストを生成する際のロジックに透明性を持たせることが条件として挙げられる．そこで，ホワイトリストをブロックチェーン技術によって生成する手法を提案する．

5. 提案手法

5.1 システム構成

提案システムを図 4 に示す．提案するブロックチェーンを用いたホワイトリスト方式防御システムでは，ユーザからの外部サイトへのアクセスをプロキシサーバである Squid[17] を経由して ICAP[18] サーバを用いて Fabric との連携を実現する．ユーザによるアクセス情報を Fabric 内のチェーンコード（Chaincode）である追加認証判定により処理を行いユーザへ CAPTCHA による追加認証を要求するかどうか判定する．CAPTCHA 認証の結果は Fabric 内で共有され，これを元に複数組織間でホワイトリストを生成・共有する．

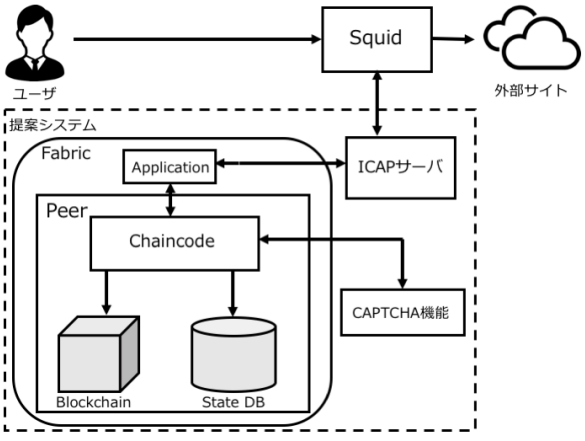

図 4 提案システム

各構成要素は以下の通りである．

- ICAP サーバ
  Squid から送られてくるユーザのリクエスト情報を ICAP による通信で受け取り，Fabric 内の Application と連携させる役割を担う．

- Application
  ICAP サーバから届いたユーザのリクエスト情報を受け取り，Fabric Client SDK により作成された当該 Application によりトランザクションの発行を行う．
- Chaincode（チェーンコード）
  Application から発行されたトランザクションをもとに State DB の参照，追加認証結果の反映などのチェーンコードを実行する．
- State DB
  State DB には，キーとしてホワイトリストとなるドメイン名，それに紐づく値として追加認証成功回数，ユーザ IP，組織 ID が保存される．
- Blockchain（ブロックチェーン）
  チェーンコードによる State DB へ更新された際のトランザクション履歴が保存される．
- CAPTCHA 機能
  チェーンコードにより追加認証が必要と判定された場合にユーザに CAPTCHA 認証画面を要求する機能を担う．

5.2 チェーンコード

提案システムで動作するチェーンコードの行うトランザクション処理は以下の通りである．State DB へ参照を行う Query トランザクションと，更新を行う Invoke トランザクションに分類される．

A)Query トランザクション

- getValue
  キーであるドメイン名を与えられた時にそれに紐づく値を参照する．
- addAuthJudge
  キーであるドメイン名，ユーザのリクエストに含まれる IP などを与えられた時に追加認証判定を実行してユーザへ追加認証を行うかどうかを判定する．

B)Invoke トランザクション

- initLedger
  Fabric のブロックチェーンネットワークが立ち上がった際に State DB の初期化を行う．
- createList
  キーであるドメイン名が State DB 内に存在しなかった場合，新たに State DB に当該ドメイン名をキーとして登録する．
- addUser
  リクエストを送信したユーザが過去にリクエストを行なったドメインに対して追加認証を行っておらず，追加認証判定により追加認証を要求された結果，成功した場合に当該ユーザの IP アドレスと組織 ID が登録される．同時に，追加認証成功数を 1 増加させる．

追加認証判定については重本らのホワイトリスト型 AED と同様の方法 [15] を用いる．チェーンコードと追加認証判定のフローの関係を図 5 に示す．ユーザからのリクエストが Fabric 内に到着すると，まずそのドメイン名をキーとする値が State DB 内に存在するか否かを調べる．存在すると判断された場合，そのドメイン名に対する追加認証成功数が定めた閾値以上である場合にのみ，ドメイン名が State DB 内のホワイトリストに登録されていると判定してアクセスを許可する．閾値に達していなかった場合，リクエストを送信したユーザ IP が過去に当該ドメインに対し，追加認証を突破しているか否かを調べる．

また，HTML ファイルが外部のサーバに保存されている CSS ファイルを参照していた場合，保存先のサーバがホワイトリストに登録されていないとアクセスが遮断されてしまい，レイアウトが崩れる問題が発生するため，ユーザからのリクエスト情報に Referer ヘッダが含まれていた場合はアクセスを許可する判定フローとした．

追加認証成功数の閾値に対しては，訓練の結果単一組織内でも 50%近いユーザが標的型攻撃メールを開封してしまうデータが得られている [19] ため，慎重に設定する必要がある．この問題に対しては今後検討を行っていく必要がある．

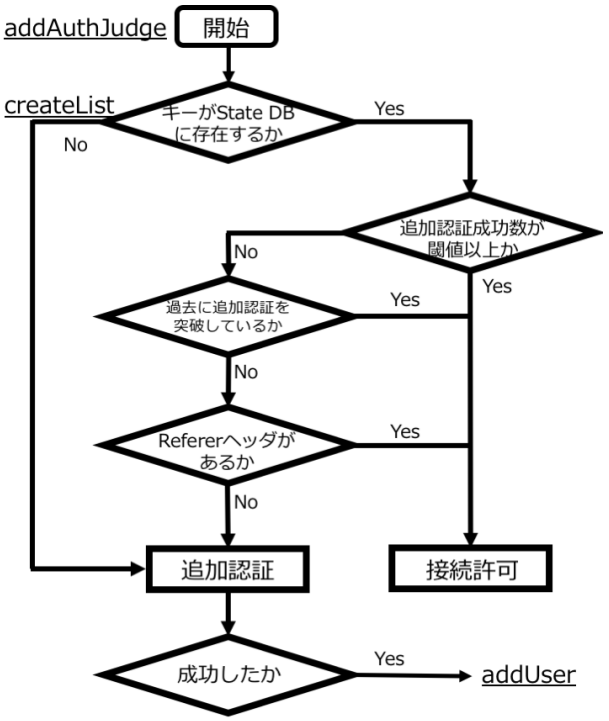

図 5 チェーンコードと追加認証判定フローの関係

5.3 State DB のデータ構造

ホワイトリストとして扱う State DB のデータ構造を図 6 のように設計した．Fabric の State DB は Apache CouchDB[20] をサポートしているため，本研究でも用いて

JSON フォーマットのドキュメントをネイティブで保存できるよう設計した．【success_num】はユーザが追加認証に成功した総数を表す．【ip】は追加認証を行なったユーザの IP アドレスで，追加認証判定のフローで過去にリクエストを送ってきたユーザが過去に追加認証を成功しているかどうか判定するために用いる．【org】はそのユーザが属する組織 ID を表す．【ip】と【org】を含む【users】は追加認証が成功する度随時追加されていく．

```
[
  {
    "Key": "fun.ac.jp",
    "Value": {
      "domain": "fun.ac.jp",
      "success_num": 1,
      "users": [
        {
          "ip": "123.456.789",
          "org": "org1"
        },
      ]
    }
  },
]
```

図 6 State DB の構造

6. プロトタイプシステムの実装

Query，Invoke トランザクションの平均実行速度（s/tx）を計測するため，プロトタイプシステムの実装を行なった．プロトタイプではユーザからのリクエストを Squid で受信し ICAP サーバと連携する部分を除き，Fabric 内のみでトランザクションを発行し，チェーンコードを実行することで State DB の参照・更新する実装としている．プロトタイプシステムの構成を図 7 に示す．Fabric は Vagrant[21] と Virtual Box[22] を用いて仮想マシン上に Docker[23] コンテナによりブロックチェーンネットワークを構築している．複数の Docker コンテナの管理には複数コンテナ管理ツール Docker Compose を用い，Application の実装には Node.js を，チェーンコードの実装には Go を用いた．また，State DB には Fabric でサポートされている Apache CouchDB を使用した．

7. 基礎評価実験及び考察

7.1 実験方法

提案システムの基礎評価実験として，実装したチェーンコードの平均実行速度の計測を行なった．Node.js で実装した Application を通して Query/Invoke トランザクションの 2 種類に分けて発行して実行した．トランザクションを単純に 100 回連続で発行して Fabric に負荷をかけるパターンに加え，複数ユーザの同時使用を想定し，並列数

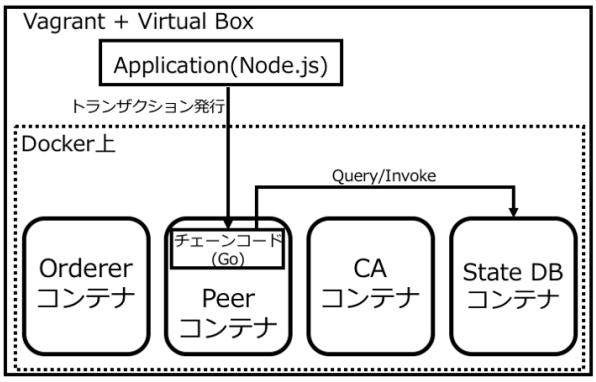

図 7 プロトタイプシステムの構成

n(n =1, 2, 3, 4, 5) でトランザクションを発行することで Fabric に負荷をかけるパターンの実験も行なった．

Query トランザクションは，図 5 の追加認証判定を実行するチェーンコード addAuthJudge により，State DB を参照し Referer ヘッダがあるかどうかまで実行させアクセスを許可するフローとして発行した．

Invoke トランザクションは，チェーンコード addAuthJudge の結果，addUser を呼び出して追加認証が成功したと仮定し，State DB に追加認証結果の更新を行うフローとして発行した．

7.2 実験結果

評価実験によって得られた Query トランザクション，Invoke トランザクションの並列数とその時の平均実行速度を表 1 に示す．

State DB 内を参照するだけの Query トランザクション平均実行速度は 0.002〜0.008 秒程度であることがわかった．一方で Invoke トランザクションでは平均実行速度は 0.2〜0.6 秒程度であることがわかった．また，チェーンコードのロジック自体は正常に動作しており，State DB への更新・参照が正しく行われていることがわかった．

Query トランザクションについては並列数によって平均実行速度にばらつきがあり，Invoke トランザクションについては，並列数が上がるにつれて平均実行速度も上がる傾向にあることがわかった．

また，Application と Peer コンテナ間の通信インターフェースに gRPC[24] を用いたが，Query トランザクションを並列数 6 以上で発行した場合 gRPC による通信エラーが発生し正常に処理が行われなかった．

Invoke トランザクションにおいては gRPC によるエラーは確認されなかったが，並列数 6 以上で発行した場合，正常に完了しないトランザクションが発生し，正常に State DB に更新がされない場合があることを確認した．

表 1 実験結果

	Query(s/tx)	Invoke(s/tx)
並列数 1	0.00405627	0.27217875
並列数 2	0.00231275	0.37106416
並列数 3	0.008093733	0.45254005
並列数 4	0.00794445	0.535991188
並列数 5	0.00412464	0.630434658

7.3 考察

Query トランザクションについては，十分実用的に扱える平均実行速度が得られたが，並列数 6 以上で実行した場合には正常にトランザクションが発行されていない場合があることから，4 章で述べた利用ユーザ数を想定した場合正常に動作しない状況も考えられるため，ブロックチェーンネットワークの構築と動作環境については Peer の数，Orderer の数などについての検討を行う必要がある．

また，佐藤らによる Fabric の性能評価 [10] と比較すると Query トランザクションについては実装や実験環境が異なるものの，同程度の実行速度が得られたが，Invoke トランザクションについては並列数をあげることにより増加傾向にある．Invoke トランザクションは Query トランザクションに比べ，Peer にトランザクションが送信されてから検証されユーザに返却された後，Orderer にそのトランザクションが送られてブロックにまとめるという処理が増加しているため Query トランザクションよりも実行速度の増加は予想できるが，既存の性能評価と比べると予想以上の実行速度の増加が得られたため，その原因を解明するためにも，ネットワーク構築や動作環境についての検討が必要である．

また，Squid がユーザからのリクエストを受信して ICAP サーバを用いて Fabric と連携させる処理を除いて Fabric 上のみでの平均実行速度を計測したが，連携させた場合でも実行速度は既存業務へ影響を与えるレベルの低下が見られないため，十分実用可能と考えられる．

8. おわりに

本研究では，近年被害が増加する標的型攻撃の防御のために，ブロックチェーン技術を用いて複数組織間で連携してシステムユーザ数を増やすことで既存の手法よりもユーザへの利便性を高めるシステムの提案を行った．提案システムに対し，基礎評価実験を行ったところ，Query トランザクションに関しては，実用レベル範囲内と考えられる実行速度が得られた．また，Invoke トランザクションは Query トランザクションに比べ Fabric の仕様上，Orderer との通信が増えるなどの影響で実行速度が増加することがわかった．Query トランザクションでは並列数によって正常な処理が行われない，Invoke トランザクションではボトルネック部分が存在しているなどの問題が発生したため，構築するネットワーク及び動作環境を再検討する必要がある．

提案システムは，プロキシサーバである Squid と ICAP サーバを経由して Fabric と連携させることでユーザからのアクセス制御を行うが，実験結果より，既存のホワイトリストのように本システムを使用する場合は十分に実用可能な実行速度が得られると予想される．しかし，追加認証をユーザが行った場合，全てのプロセスの実行完了までに時間がかる可能性がある．これについては，追加認証の成功時にはユーザに対して迅速に外部サイトへのアクセスを許可し State DB への更新処理を非同期で行うことで解決できると考えられる．しかし，更新には今回得られたような実行時間がかかるため最新の State DB の状態を次のユーザが参照できるまでにはある程度時間がかかってしまう．この問題に対する対策は今後検討していく必要がある．

参考文献

[1] IPA: 情報セキュリティ 10 大脅威 (2018), 入手先 ⟨https://www.ipa.go.jp/security/vuln/10threats2018.html⟩ (参照 2018-05).
[2] Eric M. Hutchins et al.: Intelligence-Driven Computer Network Defence Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains, ICIW2011, 2011.3
[3] Satoshi Nakamoto: Bitcoin: A Peer-to-Peer Electronic Cash System(2009), 入手先 ⟨https://bitcoin.org/bitcoin.pdf⟩ (参照 2018-05)
[4] Everledger, 入手先 ⟨https://www.everledger.io/⟩ (参照 2018-10)
[5] Ethereum ホワイトペーパー, 入手先 ⟨https://github.com/ethereum/wiki/wiki/White-Paper⟩ (参照 2018-10)
[6] The Linux Foundation ホームページ, 入手先 ⟨https://www.linuxfoundation.jp/⟩ (参照 2018-10)
[7] Hyperledger Project ホームページ, 入手先 ⟨https://www.hyperledger.org/projects/fabric⟩ (参照 2018-10)
[8] Hyperledger Fabric ホームページ, 入手先 ⟨https://www.hyperledger.org/⟩ (参照 2018-10)
[9] 掘寿美 他: 学習経済に基づくクラウドソースラーニングの提案, 情報処理学会研究報告, Vol.2018-CLE-25, No.3, pp.1-8(2017)
[10] 佐藤竜也 他: ブロックチェーン基盤 Hyperledger Fabric の性能評価と課題整理, 情報処理学会研究報告, Vol.2017-IOT-36, No.30, pp.1-8(2017)
[11] Hyperledger Fabric 公式ドキュメント, 入手先 ⟨https://hyperledger-fabric.readthedocs.io/en/latest/⟩ (参照 2018-08)
[12] 早川勝監修: ブロックチェーンの革新技術 Hyperledger Fabric によるアプリケーション開発, リックテレコム (2018)
[13] 角田朋 他: グレーリストを用いたホワイトリスト/ブラックリストの自動生成によるマルウェア感染検知方法の検討, 情報処理学会研究報告, Vol.2014-SPT-10, No.16, pp.1-7(2014)
[14] 仲小路博史 他: 人間行動を用いた自律進化型防御システムの提案, 暗号と情報セキュリティシンポジウム 2016(SCIS2016), pp.1-8(2016)
[15] 重本倫宏 他: ホワイトリストを用いた自律進化型防御システムの開発, 情報処理学会論文誌, Vol.59, No.3, pp.1050-1060(2018)

[16] CAPTCHA, 入手先 ⟨http://www.captcha.net/⟩ (参照 2018-10)
[17] Squid, 入手先 ⟨http://www.squid-cache.org/⟩ (参照 2018-10)
[18] Internet Engineering Task Force: Internet Content Adaptation Protocol(ICAP), 入手先 ⟨https://tools.ietf.org/html/rfc3507⟩ (参照 2018-10)
[19] たぷるとぽちっと: 開封率は約 50%！本当に自分は大丈夫？〜「標的型メール攻撃訓練」の結果〜, 入手先 ⟨https://www.ines-solutions.com/article/2017/11/28/125⟩ (参照 2018-10)
[20] Apache CouchDB ホームページ, 入手先 ⟨http://couchdb.apache.org/⟩ (参照 2018-10)
[21] Vagrant, 入手先 ⟨https://www.vagrantup.com/⟩ (参照 2018-10)
[22] Oracle VM VirtualBox, 入手先 ⟨https://www.virtualbox.org/⟩ (参照 2018-10)
[23] Docker, 入手先 ⟨https://www.docker.com/⟩ (参照 2018-10)
[24] gRPC, 入手先 ⟨https://grpc.io/⟩ (参照 2018-10)