JPRS トピックス & コラム

No.024

DNS を用いたサーバー証明書の発行制御 ～ CAAリソースレコードの概要と現状～

ドメイン名登録者が DNS を用いて認証局（以下、CA）に対しサーバー証明書の発行制御を行う「CAA リソースレコード」の概要と現状について解説します。

サーバー証明書の発行制御が行われるようになった背景

サーバー証明書はインターネットの安全な利用に欠かせない、重要な要素の一つです。しかし、近年、サーバー証明書の誤発行・不正発行によるセキュリティインシデントの発生が、しばしば問題になっています。

悪意を持つ者が誤発行・不正発行されたサーバー証明書を入手し、DNS情報や経路情報を偽って利用者を偽サイトに誘導する攻撃手法(*1)と組み合わせることで、例えば以下の状況を作り出すことが可能になります。

- ブラウザーには目的のサイトのURLが表示されている
- **誤発行・不正発行されたサーバー証明書により、アドレスバーに鍵マークが表示されている**
- にもかかわらず、アクセス先は偽サイトである

このように、サーバー証明書の誤発行・不正発行はサーバー証明書とそれを活用するSSL/TLSの信頼性を損ない、インターネットの安全な利用に対する重大な脅威となります。

DNS を用いたサーバー証明書発行制御の仕組み

サーバー証明書の誤発行・不正発行の検知の方法には、以下の二つがあります。

- 誤発行・不正発行を発行**前**に検知する
- 誤発行・不正発行を発行**後**に検知する

DNSを用いたサーバー証明書の発行制御は、前者の仕組みの一つです。ドメイン名登録者が発行制御のための情報を自身の権威DNS サーバーに**CAA**(*2)**リソースレコード**で設定し、申請を受け取ったCA がサーバー証明書の発行前にDNS 検索で内容を確認することで、誤発行・不正発行を検知します（図 1）。

なお、後者の仕組みの代表的なものとして、CT（Certificate Transparency）が挙げられます(*3)。

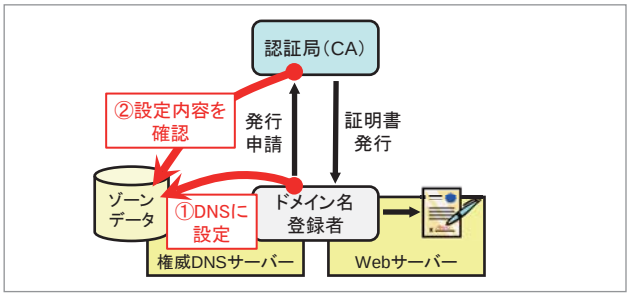

図 1 DNS を用いたサーバー証明書の発行制御

CAAリソースレコードの概要

CAAリソースレコードの仕様はRFC 6844として、2013年に標準化されています。CAAリソースレコードの記述例とその意味を、図2に示します。

＜設定例＞

```
① example.jp. IN CAA 0 issue "jprs.jp"
① example.jp. IN CAA 0 issue "ca.example.com"
② example.jp. IN CAA 0 issuewild ";"
③ example.jp. IN CAA 0 iodef "mailto:security@example.jp"
```

＜設定の意味＞

① example.jpの証明書発行を「jprs.jp」と「ca.example.com」に許可
② example.jpのワイルドカード証明書発行は不許可
③ 許可されていないCAが証明書の発行要求を受けた場合、<security@example.jp>に、電子メールを送ってほしい

図 2 CAA リソースレコードの設定例とその意味

サーバー証明書の発行を許可するCAをissue/issuewild属性タグを用いて、発行を許可しないCAに発行要求があった際の連絡先と連絡手段をiodef属性タグを用いて指定します。

CAAリソースレコードを利用する場合、DNSSECの利用が強く推奨されています。DNSSECを利用することでCAが受け取ったCAAリソースレコードが、ドメイン名登録者が設定したものであることを確認できます。

(*1) DNSキャッシュポイズニング、登録情報の不正書き換えによるドメイン名ハイジャック、BGP経路ハイジャックなど。
(*2) CAAはCertification Authority Authorization（認証局の許可）を意味しています。
(*3) CTについては本コラムでは解説しません。

CAAリソースレコードを用いた発行制御の判断の流れ

CAがCAAリソースレコードを用いてサーバー証明書の発行制御を判断する流れを、図3に示します。

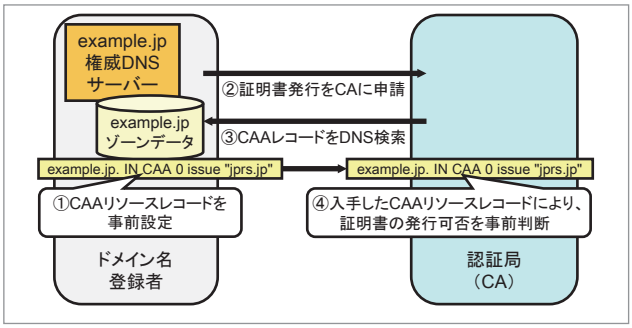

図 3 CAA リソースレコードによる判断の流れ

ドメイン名登録者はサーバー証明書の申請に先立ち、CAAリソースレコードを自身の権威DNSサーバーに設定します。申請を受け取ったCAはCAAリソースレコードをDNS検索し、設定されていた場合、その内容に従います (*4) (*5)。

CAAリソースレコードはサーバー証明書の利用時には検証されないため、発行後に設定内容を変更しても、使用中のサーバー証明書の有効性には影響しません (*6)。

DNS検索における注意点

CAがCAAリソースレコードをDNS検索する場合、DNSの階層構造をさかのぼる形でTLDまで順に検索し、最初に見つかったCAAリソースレコードを設定結果として用います。

例えば、www.example.jpにおけるCAAリソースレコードの検索手順は、表1のようになります。

① 「www.example.jp」のCAAを検索 ⇒見つかった場合検索終了、見つからない場合②へ
② 「example.jp」のCAAを検索 ⇒見つかった場合検索終了、見つからない場合③へ
③ 「jp」のCAAを検索 ⇒見つかった場合検索終了、見つからない場合④へ
④ 検索終了、CAAは設定されていなかったと判断

表 1 CAAリソースレコードの検索手順

そのため、CAAリソースレコードを設定していなかった場合、親ドメインのCAAリソースレコードにより予期しない形でサーバー証明書の発行が制限されてしまう場合があります (*7)。

ドメイン名の管理者が親ドメインのCAAレコードとは異なる設定を適用したい場合、サブドメインごとにCAAリソースレコードを設定する必要があります。

業界団体による必須化

業界団体であるCA/Browser Forum (*8)が、2017年9月8日以降のCAにおけるCAAリソースレコードを用いた発行制御の判断を必須化しました (*9)（JPRSのサーバー証明書発行サービスでは既に実装済みです）。

DNSソフトウェアにおけるサポート状況

主な権威DNSサーバーソフトウェアにおけるCAAリソースレコードのサポート状況を、以下に示します。

- BIND 9.9.6以降
- NSD 4.0.1以降
- PowerDNS Authoritative Server 4.0.0以降
- Knot DNS 2.2.0以降
- Windows DNS Windows Server 2016以降

DNSプロバイダーにおけるサポート状況

CAAリソースレコードの設定をサポートしている主なDNSプロバイダーを、以下に示します（2018年1月現在）。

- Amazon Route 53
- CloudFlare Global Managed DNS
- Dyn Managed DNS
- Google Cloud DNS
- Neustar UltraDNS
- さくらインターネット ドメインメニュー

IETFにおける問題点の指摘と改定作業

業界団体による必須化を受け、DNSソフトウェアやDNSプロバイダーにおいてCAAリソースレコードをサポートする動きが広がっています。その一方、RFC 6844の仕様上の問題点がIETFで指摘され (*10)、改定作業が進められています (*11)。

(*4) CAAリソースレコードの設定は任意であり、設定がない場合は従来通りの動作となります（発行制御なし）。
(*5) サーバー証明書の更新・再発行の際にも事前検証されます。
(*6) ドメイン名登録者が別のCAに乗り換える場合、CAAリソースレコードの内容を書き換えた後で、乗り換え先のCAに発行を申請します。
(*7) JPRSではjpやco.jpなどに、CAAリソースレコードを設定していません。
(*8) 電子証明書を使った通信の安全性やその利便性を向上させるためのガイドラインを策定している、会員制の任意団体です。
(*9) Ballot 187 - Make CAA Checking Mandatory - CAB Forum
https://cabforum.org/2017/03/08/ballot-187-make-caa-checking-mandatory/
(*10) 検証対象のドメイン名にCNAME/DNAMEが設定されている場合の、検索アルゴリズムの問題点が指摘されています。
(*11) DNS Certification Authority Authorization (CAA) Resource Record
https://tools.ietf.org/html/draft-hoffman-andrews-caa-simplification