コンピュータ・セキュリティー概略とアプリの作り方

目次

コンピュータ・セキュリティ－概略とアプリの作り方

はじめに

汎用機（メインフレームやレガシーとも呼ばれます）からオープンシステムへの移行が始まったとき、ネットワークからの侵入、データ破壊、情報漏洩、ウィルス etc.のセキュリティインシデントが問題になってきました[1]。背景として LAN(Ethernet)が装置間の相互接続や信頼性、可用性、利便性を主眼に技術開発が進められたことがあります。LAN がインターネットへと拡大すると利便性がそのまま脅威に変わりました。基本的なサービス・プロトコル（ファイル共有、名前解決、時刻同期、メール、遠隔ログイン 他）はインターネット時代になってからセキュリティ対策を施したバージョンが公開されたり別のプロトコルに置き換える選択肢が作られましたが脆弱なソフトウェアが導入時のまま残っていたり、ネットワークの中核で使われるルータや L3 スイッチが昔ながらの telnet で管理者としてログインできる設定で出荷されたりしています。

国際的なランサムウェアや DDoS 攻撃のニュースが増加している昨今ですが、日本国内では今でも性善説と大きなミスは犯さない前提で利便性抜群の Web を使ってシステムを作っています。しかし、日本人でも大ポカはするしそもそもシステムの製造を海外の見知らぬ人に任せてますよね...。

1. セキュリティ対策の階層

コンピュータを取り巻くセキュリティ対策はこれだけで OK というものはありません。脅威の対象毎に対策が必要で以下が代表的なものです。これらを背景にしたアプリケーション設計が必要です。

1.1. 施設、設備

コンピュータや周辺装置は電源を停止したり連結するケーブルを抜くだけで正常に動作しなくなります。業務継続に必要なデータの保管・バックアップ環境を含めて地震が少ない地域の住宅街に目立たないように施設を作ったり、重要度に応じた入場制限や設備への侵入監視装置の設置等が考えられます。また、DDoS 攻撃のような前触れなく急激に資源を消費させられるものは自前で攻撃に耐えられる設備を準備して監視や防御を行うことは困難なので、外部の不特定多数に公開するようなサイトはクラウド業者に管理を依頼するのが賢明です。

1.2. 管理・運用

セキュリティレベルをどの程度にするかは取り扱う情報の重要度で決まりますが、以下の観点が必要になります。

1.2.1. 個人認証とサーバ認証

不特定多数向けの情報発信を行うサイトを除き、権限のない要員や端末装置がネットワークに入れないようにしたり権限外の操作ができないように個人や装置の識別・認証が必要になります。また、フィッシングや DNS の書き換えよるなりすましサイトへの誘導に対しては、外部機関によるサーバ証明書をサーバ内に設置してクライアント側に確認（https を利用）してもらいます。

---

[1] JPCERT/CC セキュリティインシデント年表
https://www.jpcert.or.jp/magazine/chronology/index.html

コンピュータ・セキュリティ－概略とアプリの作り方

1.2.2.　認証とアクセス制御

社内のシステムであれば、操作が許されているか操作が許されているなら触れてよい情報の範囲、追加・更新・削除を許すか否かを判定し許可外の操作を抑止するのがアクセス制御です。組織の内部要員が悪意でデータ破壊することはあまりないと思いますが、故意か偶然か権限外の情報を見ようとしたりその操作ログを消そうとしたりすることは実際に発生します。また、退職する人間が営業秘密をライバル企業に持ち込んで訴訟になったというニュースも耳にします。

情報の参照／操作の許可の範囲は操作者の職務や職位等で決まります。日常業務で使うシステムであればあまり細かな分類は必要ないかもしれませんが、システム管理者／一般利用者の識別やアプリケーションの使用可否判定（表示メニューの切替）、操作ログの作成は障害対策としても必要です。

インターネットで不特定多数に公開しているシステムは世界中の悪意ある人間から挑戦を受けることになるので厳しいアクセス制御が必要です。クレジットカードの発行申請や商品販売のサイトではアカウントを持たない状態でアクセスを許可する（むしろ引き付ける）必要がある一方、既存会員の個人情報や購入履歴が漏れると大きな損害を生みます。では、個人情報や個人の使用履歴をサイトから消せばよいかというと使い勝手からしたらそうもいきません。

脅威や対策の主なものは以下です。

● 登録済アカウントの詐称

システムで扱う情報の重要度に応じて２段階認証やUSBトークン等の物理的な装置を利用します。

● ログイン不要の環境での情報窃取

会員登録画面があるサイトは会員情報にアクセスできる環境でありながら不特定多数にアクセスを許している環境になっているというとても危険な環境です。原則、利用者が入力した情報以外は再表示しないようにします。具体的な対策はサービスの権限設定とアプリケーションの実装で行います。

1.2.3.　アクセス制御とサービスの権限

サービス・アプリケーションには起動したユーザの権限（ファイルへのアクセス権等）が全て引き継がれます。Web アプリサーバの先駆け Tomcat はバージョン４辺りまではシステム管理者（Linux の場合 root）として起動するのがデフォルトだったので、侵入されるとシステム管理者の権限が全て奪われる事態になっていました。現在は（インストーラーによりますが）OS に対しては特権をもたないユーザ ID（Linux の uid、Windows のアカウント）で起動するように構成されます。

メーカー製のソフトウェアも同様に管理者権限で動作している可能性があるので、古いバージョンのサービスを使い続けたり上書きでインストールしている場合は起動状態を確認して OS に対する過剰な権限は外しておく必要があります。また、アプリケーションが問い合わせや構成変更（DDL の実行）に使う RDBMS の ID／パスワードはコードに直書きにせず別ファイルから参照するようにしておいた方が本番／テストの環境切替が簡単で安全です。

1.2.4.　データ保管

故意かミスかを問わず、人為的なデータ漏洩や破壊はしばしば発生します。発生事象を把握するため重要な資材については操作の成功／失敗を問わず操作記録を残します。また、自然災害に対しては遠隔地のサイトとミラーリングを行い、データを二重化／三重化して備える方法があります。

但し、ランサムウェアによる汚染はミラーリング先にも及ぶため事業継続の重要度によっては外部媒体へのバックアップと更新ジャーナルの時系列での運用管理も必要になります。

コンピュータ・セキュリティ－概略とアプリの作り方

1.2.5. ネットワーク管理

ネットワークは脅威の入り口になります。そして脅威は組織の内側にも外側にも存在します。

（1）LAN

未承認の装置が接続されるのを防ぐにはMACアドレスで接続を制限したりハブの空ポートを塞ぐ方法もありますが、MAC アドレスは簡単に書き換えができ VPN が使われる環境では物理的なハブへの接続可否では制御できません。対策としてサーバ証明書と個人証明書やトークン、生体認証を使った相互認証を行う方法がありますが、そこまで設備投資する価値は無いと判断するなら情報を管理しているサーバ側で個人認証とアクセス制御を行います。どの認証方法でも個人がウィルスを取り込むとその人の権限がウィルスに利用されてしまうのでアンチウィルスの対策も重要です。

（2）インターネット

ネットワークは一般的にファイアーウォールで外部からの攻撃から守りますが、内部から外部への接続を監視して制御している組織はあまりありません。プロキシで利用するサーバのポート番号を制限することは可能ですが、一般にプロキシが通過を許可する http（80 番ポート）や https（443 番ポート）を使って VPN を設定することもできるので危険性は残ります。メール添付のファイルを開かない、未認可のアプリをインストールしないといった利用者への教育、アンチウィルスの対策に加えて、重要な情報を扱うシステムでは侵入検知ソフトやパケットの内容を検証するアプリケーションレベル（L7）のファイアーウォールが必要になる場合があります。

（3）盗聴対策

ネットワークアダプター（NIC）は通常 自分の MAC アドレス宛のパケットだけを拾うようにしていますが、LAN上に流れているデータは同一LANに接続している全ての機器から見えています。

そこで情報保護のために https,ssh,scp 等による SSL/TLS の暗号化通信が推奨されていますが、Web アプリのテストツール（Fiddler,mitmproxy,Wireshark...）やアンチウィルスソフトはパケットを復号化して内容を見ることができ、更に書き換えることができるものもあります。

SSL/TLS の復号化は自側（ブラウザ等）と通信先のサーバの間に割り込み（プロキシとして設定）通過するパケットを捕捉して行います[2]。但し、サーバ／PC でリモートデスクトップによる接続が有効に（システムのプロパティ「このコンピューターへのリモート接続を許可する」にチェック）なっている場合は RDP を使ってこっそりこの盗聴に便利なツールを仕込まれる危険があります。

更に、Windows には WFP(Windows Filtering Platform)という API[3]があり、プロキシの設定なしで TCP/IP パケットのフィルタリング、検査と変更、接続の監視または承認、IPsec の規則と処理、および RPC フィルタリングが可能です。セキュリティを高めるための機構ですが、マルウェアに利用された事例があります。環境に問題が無さそうに見えても脅威が潜んでいる可能性は残ります。

---

[2] SSL/TLS の復号化には上位認証局の証明書（プロキシを認証する上位認証局の証明書またはルート証明書）を「信頼されたルート証明機関」等に登録する必要があり、ウィルスであってもルート証明書の登録許可を求める表示が出ます。ブラウザやインストーラの操作中に英文のダイアログで”CA”や“Trust”等の単語を見つけたら操作を中止して内容の確認が必要です

[3] Windows Filtering Platform Architecture Overview
https://learn.microsoft.com/en-us/windows-hardware/drivers/network/windows-filtering-platform-architecture-overview

コンピュータ・セキュリティ－概略とアプリの作り方

1.2.6. キャッシュ

ネットワークに関連した多くの情報がローカルに保存（キャッシュ）されます。

Windows であれば nbtstat コマンドや ipconfig コマンドでキャッシュされたホスト名、IP アドレスの対を確認できます。この組合せに虚偽の情報を登録する DNS キャッシュポイズニングという攻撃が有名です。各種対策が編み出されていますが新たな攻撃方法も見つかっています。DNS サーバを設置して外部公開しているドメインを自前で管理するであればインシデント情報に注意して DNS サーバを最新に保つ必要があります。また、キャッシュ DNS サーバは外部に公開しないようにします。

また、ブラウザを使うと（ブラウザによっては）https であってもアクセスした URL の表示内容やクッキー、資格情報（ID/パスワード 他）、フォーム入力データ、保存データが残されます。

Chrome ブラウザや Microsoft Edge のベースになっている Chromium はキャッシュを暗号化して保存しますが、この暗号化のパスワードを復元するというソフトウェアもフリーで公開[4]されています。

http のキャッシュはブラウザやプロキシに保存されますが、Web アプリの場合は動的に表示内容を書き換えたいので html のキャッシュは好ましくありません。

対策として http ヘッダに Cache-Control: no-store を指定して保存を抑止します。

また、フォームの自動入力は html の autocomplete 属性とブラウザの設定で保存有無が決まります。

＜nbtstat コマンド実行例＞

```
C:¥Users¥User>nbtstat -n

ローカル エリア接続* 11:
ノード IP アドレス: [192.168.11.2] スコープ ID: []

                NetBIOS ローカル ネーム テーブル

    名前                種類          状態
---------------------------------------------
hostname            一意          登録済
```

＜ipconfig /displayname コマンド実行例＞

```
C:¥Users¥User>ipconfig /displaydns

Windows IP 構成

    [redacted].com
    ----------------------------------------
    レコード名 . . . . . . . . : [redacted].com
    レコードの種類 . . . . . : 28
    Time To Live  . . . . . . : 1360
    データの長さ . . . . . . . : 16
    セクション . . . . . . . . : 回答
    AAAA レコード. . . . . . : 2[redacted]0:4110:86f::

    [redacted].com
    ----------------------------------------
    レコード名 . . . . . . . . : [redacted].com
    レコードの種類 . . . . . : 1
    Time To Live  . . . . . . : 1099
```

[4] NirSoft web site provides a unique collection of small and useful freeware utilities, all of them developed by Nir Sofer.


```
データの長さ . . . . . : 4
セクション . . . . . : 回答
A (ホスト) レコード. . . : 3[redacted]5.123
(以下略)
```

1.3. 外部から導入したサービス、ソフトウェア

セキュリティホールに関する情報は JPCERT/CC のメーリングリスト[5]等で受け取ることができます（英文で構わなければ US-CERT の同様のサービス[6]に元情報が掲載されていることが多いです)。

ネットワークに公開するサービスには幾つか種類があり、セキュリティホールが見つかったときの対策や注意点が若干異なります。

（1）システムポート

IANA(Internet Assigned Numbers Authorit)がシステムポート（ ~1023 番）として定義しているサービス（SSH、FTP、メール、DNS、HTTP/HTTPS、LDAP …）は最新版へのソフトウェア更新と不要ポートの閉鎖が対策になります。但し、HTTP/HTTPS で動的コンテンツを扱う場合は Web アプリの実装に関わるのでバージョンアップに関して次項のミドルウェアとしての注意が必要になります。また、DNS（名前サービス）はドメイン、サーバが増加するとともに管理が複雑になったりセキュリティ対策が難しくなってきたりで新しサービスアプリへの乗せ換えも検討対象になります。サービスアプリを変えた場合は環境定義や構成ファイルの新たな作成と動作確認が必要になります。

（2）ミドルウェア

セキュリティパッチの適用かバージョンアップを行います。

RDBMS のようにデータの継続が必要なものではデータのコンバージョンが必要になる場合があります。特にオープンソースの場合バージョン番号の先頭の番号（メジャーバージョン）が変わると機能が追加されるだけでなく、データファイルの構造が変わったり取り扱えるデータ型の変更や廃止になる機能もあるためアプリケーションの改修や運用コマンドの見直しも必要になることがあります。また、環境定義ファイルの設定項目やデフォルトが変わってバージョンアップ後に前の設定が無効になってしまうようなことも起こります。

（3）パッケージ製品

メーカが販売している製品でも App サーバや運用ツールを中心にオープンソースのソフトが根幹部分に使われているものがあります。特に問題になるのは使っているフレームワークに脆弱性が見つかった場合にメーカが手を加えていたり、メーカ独自の JVM 等が組み込まれていると解決策が提示されるまでに時間が掛かったり古いものだと対策不能の回答がされる場合があります。

汎用的な対策はありません。

※Windows サーバの場合は更新の反映に OS の再起動を伴うものが多いので、停止できないサーバの場合は多重化と切り替えの自動化が必要になります。

---

[5] JPCERT/CC メーリングリストの申し込み https://www.jpcert.or.jp/announce.html

[6] The Cybersecurity and Infrastructure Security Agency (CISA) Mailing Lists and Feeds
https://www.cisa.gov/uscert/mailing-lists-and-feeds

2. 攻撃者の段取りと貧者の対策

攻撃の対象と方法は幾つかあり不特定多数を狙う攻撃は目に見えず常時降りかかっていますが、狙いを付けられて集中的に行われるものでなければ高価な機材が無くても基本的な対策で対処できます。

2.1. 侵入

インターネットを使った攻撃の対象は JPNIC にドメイン名を登録しているサイトだけではありません。グローバル IP アドレスがあれば固定アドレスである必要さえありません。

《以下の手順は不正アクセスまたはそれを疑われる行為です。実際に試すのはやめてください》

例えば、IP アドレスがあれば番号体系から凡その設置場所が分かります。逆に地域を決めれば IP アドレスの範囲が決まるので、順番に ping を実行して応答を返すようなうっかりものを探します。

インターネットルータにはインターネットからリクエストが来ると内部ネットワークの特定のホストに転送する機能が付いていて、組織内からのみリモートで使うつもりのホスト（軽量の Linux を搭載した Web カメラや NAS を含む）を接続している場合があります。これらが ping に応答したらその IP アドレスの全てのポートに対して接続を試みます（ポートスキャン）。接続が成功する必要はありません。エラーを返してくるうっかりものが居たら、そのエラーメッセージからポートを開けているサービルの種類やバージョンが分かるので、セキュリティホールがあるバージョンのソフトウェアを見つけてゼロデイ攻撃や既知の攻撃手順を試みます。

ポートがエラーを返さない場合でも TELNET や FTP、SSH のサービスが上がっている場合は大抵標準のシステムポートを使っているので、ありがちな ID とパスワードで接続を試みます。例えば FTP はスクリプトから実行することも多いのでうっかり ftp/ftpuser のような簡単なアカウントにしがちです。ちなみに FTP はサブコマンドを介して OS コマンドを実行できます。

Web カメラや NAS（Network Attached Storage）が出荷時のデフォルトの管理者 ID／パスワードで（多くの場合そうですが）うっかり接続されていたら簡単に乗っ取られて DDoS の足場にされたりデータを暗号化されて身代金請求の人質にされる可能性があります。

ホストに入られると hosts ファイルや DNS キャッシュ、Windows なら NBT キャッシュから他のホストの存在が分かり、passwd、group ファイルからは管理者権限を持つ uid が分かります。攻撃者が管理者権限を持つアカウントに数分成り代わったら新しい管理者アカウントを作るか、活動していなさそうなアカウントに管理者権限を付加して最後にログを消します。更にマルウェアをインストールして定期的に攻撃者のホストに通信するようにしておけばグローバル IP アドレスが変わっても追跡できてしまいます。また、管理者のコマンドを使えばネットワーク構成を調べたりルート変更が可能なので接続先になっている顧客等の外部ネットワークも侵入の対象になります。

以上の手順はスクリプト化が可能でツールが出回っています。このように“うっかり”が幾つか重なると攻撃者がこっそり忍び込んでくる可能性があります。対策は「不特定多数に公開しているサーバでないならば外部ネットワークの ping に応答しない」、「不要なポート=サービスを閉じる」、「外部ネットワークから見える機器に設定されているデフォルトのパスワードは変更する」、「セキュリティアップデートをあてる」等のコストのかからない方法でも大分リスクを減らすことができます。

コンピュータ・セキュリティ－概略とアプリの作り方

2.2. フィッシング

主に個人の興味を引いて行動させる手法で、他の事業者を名乗って偽のリンクを添付したメールで罠を仕掛けた偽サイトに誘導したり、有名 EC サイトに出品して高評価を自演することで興味を持たせ攻撃サイトに誘って個人情報を入力させたりボットやランサムウェアをダウンロードさたりします。

攻撃者自身のサイトではなく、脆弱性を持つサイトへのリンクに攻撃用のリクエストパラメータを含ませてクロスサイトスクリプティング（XSS）を仕掛ける場合もあります。

対策はメール等で送られてきたリンクは「表示されているサイト／URL とリンクの URL が一致していることを確認する」「送られてきたリンクはクリックしない」「私用の PC でネットワークに接続しない」等の教育をすることです。また、扱う情報の重要度によってはプロキシを設置して信頼できるサイト以外はリクエストを送れないように制限するホワイトリスト方法が、開発用のネットワークでは怪しいリンクを発見する都度リスト化するブラックリスト方式が使い勝手が良いです。

また、XSS に関しては Web アプリの脆弱性が使われるのでサイト運営者の責任が問われます。

2.3. Web アプリのコードを想定した攻撃

攻撃対象のコードを想定して入力フォームに OS コマンドや SQL を実行させる文字列を入力する OS コマンドインジェクション、SQL インジェクション、HTTP リクエストの URL パラメータから内部処理を推測してパラメータを追加したり第三者のサーバを指定してリクエストを送信させる SSRF(Server Side Request Forgery)等があります。

これらはアプリケーションでの対応が必要です。

3. アプリケーション

以下、攻撃に晒されやすい Web アプリを使って運用環境と実装面からリスクを挙げます。

3.1. アプリケーションの運用から発生するリスク

悪意を持つ誰かは、以下のことをしようとするかもしれません。

・アプリのユーザ設定を利用した管理者権限の乗っ取り
・Web 資産の改変
・秘密情報の窃取

アプリケーションの脆弱性を利用してコマンドを実行されるとアプリケーション（Web コンテナ）のアクセス権限を利用されます。管理者権限で稼働させていた場合 外部からホスト（OS）に侵入されるのと同じ危険性が発生し、攻撃者が書き変えた資材でサービスを乗っ取ることも可能です。

内部からはビルドの途中で資材の一部のファイルをすり替えられるリスクがあります。Java の中規模のシステムであれば war に含まれるファイル数は千を超え個々の内容確認は困難ですが、実行ファイルが「成功」で作られたときもビルドの関連スクリプトと途中経過のログは確認が必要です。

開発中は RDBMS への接続用の ID／パスワードは固定のまま見える状態で放置しがちです。せめて DB 毎、参照用と更新用で変えた方が安全度はあがります。

運用時はログに書く情報も注意が必要です。開発時は入力データは全てログに出したいところですが運用時のパスワードや機密情報がログに残らないようにログ出力レベルを変更します。

3.2. アプリケーションの実装

アプリケーションがセキュリティー上で考えなければいけない点は、「パスワードや営業秘密に当たるような内容を“DEBUG”以上の出力レベルでログを作成しないようにする」といった運用と関係する部分を除けば脆弱性を除くことに尽きます。攻撃対象となる代表的なポイントを挙げます。

3.2.1. サービス側が他サーバに出すリクエスト（SSRF）

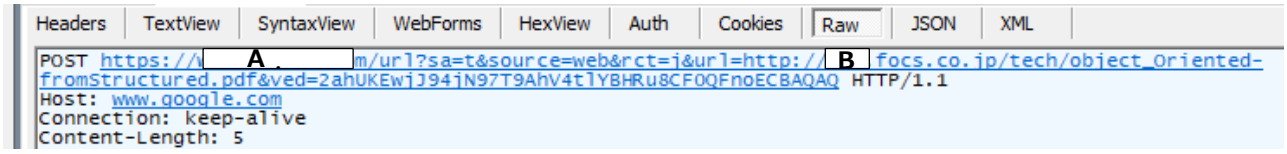
サービス側がクライアントから受け取ったリクエストパラメータに含まれる URL（第三者サーバ）にリクエスト出す処理形態の場合、第三者サーバでは“サービス”としてリクエストが処理されます。

クライアントは第三者サーバへのネットワーク経路やアクセス権を持っている必要がなく、curl 等のコマンドを使ってスクリプト化したりパラメータを自由に変えることができます。

以下の例は、検索サイトが返してくる検索結果のリンクです。

検索結果の一覧からリンクをクリックすると、次図のように A（検索サービス）に対して url パラメータで B サーバを指定した https リクエストが出ます

※図中の B をクライアントから到達できない URL に書き換えても A から見えれば処理が実行される



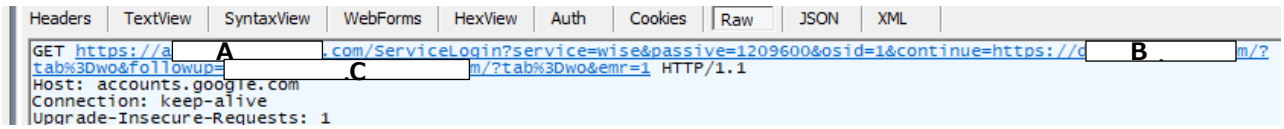
3.2.2. フィッシング（XSS）

下図は、次のユースケースで実際に作られたリクエストです。

① 未認証のまま B サービスを呼出す（そのときの画面で C が入力済）

② B サービスはログイン画面 A にリダイレクトする（パラメータに戻り先として B 書いておく）

③ A 画面で ID／パスワードで認証できたら①の画面に戻りサービスを継続する



この GET リクエストは HTML 形式のリンクにしてメールに書くことができ、しかもパラメータを隠すことができるので、以下の攻撃パターンが考えられます。

・攻撃用の偽サイト（例えば、攻撃対象が利用している銀行や EC サイト）を作る
・前図 B（戻り画面）に偽サイトの URL を指定する
・上記のリンクを書いたメールを攻撃対象に送付する
※攻撃者は偽サイトで待ち受けていれば、攻撃対象が取得した正規のセッションコード等も入手できます
（別のパターンとして）
・前図 C にスクリプトを記述し、リンクを攻撃対象にメールする
・A のサイトはリンクからリクエストパラメータを受け取り画面を生成します
※ A のサイトがリクエストパラメータの内容をそのまま画面に編集していたら（XSS の脆弱性）クライアント（攻撃対象）側でスクリプトが実行されて情報の窃取に繋がります

3.2.3. サイトの脆弱性を直接攻撃（OS コマンドインジェクション／SQL インジェクション）

アプリケーションが入力値をどのように使うか予測し、OS のコマンドや SQL の一部に使われると意味を持つ文字列を http リクエストパラメータに混ぜ込みます。

日本でも 2016 年に OS コマンドインジェクションを使った大規模な情報流失が起きています。

CVE-2016-1204 https://www.jpcert.or.jp/at/2016/at160019.html

2021 年にも類似製品で...

CVE-2021-20837 https://www.jpcert.or.jp/at/2021/at210047.html

OS コマンドインジェクションの解説サイト

https://www.ipa.go.jp/security/awareness/vendor/programmingv2/contents/501.html

脆弱性のあるサイトに対しては、この攻撃は画面の入力欄に 1 行のテキストを書き込む程度の簡単な操作で完結します。

3.2.4. 対策

既知のリスクに対してはフレームワークでほぼ対処ができます。但し、フレームワークを使っていても故意か誤りかにより脆弱になる場合があります。具体的には別資料に纏めます。

以上

