セキュリティ勉強会

実践編

2017/08/30

目次

例のアレ

損害賠償請求事件
東京地方裁判所判決／平成２３年（ワ）第３２０６０号
平成２６年１月２３日
ウェブサイトによる商品の受注システムを利用した顧客のクレジットカード情報が流出した事故につき、システムの設計、製作、保守等の受託業者の債務不履行に基づく謝罪・問合せ等の顧客対応費用、売上損失等の損害賠償責任が肯定された事例

主　　文

１　被告は，原告に対し，２２６２万３６９７円及びこれに対する平成２３年１０月１５日から支払済みまで年６分の割合による金員を支払え。
２　原告のその余の請求を棄却する。
３　訴訟費用はこれを５分し，その４を原告の負担とし，その余を被告の負担とする。
４　この判決は，第１項に限り仮に執行することができる。

http://www.softic.or.jp/semi/2014/5_141113/op.pdf

セキュリティ対策はやって当たり前

SQLインジェクションが原因でカード情報が流出した事件です。

判決のポイントは、

そこで検討するに，証拠（甲１４，２５，２９）によれば，経済産業省は，平成１８年２月２０日，「個人情報保護法に基づく個人データの安全管理措置の徹底に係る注意喚起」と題する文書において，ＳＱＬインジェクション攻撃によってデータベース内の大量の個人データが流出する事案が相次いで発生していることから，独立行政法人情報処理推進機構（以下「ＩＰＡ」という。）が紹介するＳＱＬインジェクション対策の措置を重点的に実施することを求める旨の注意喚起をしていたこと，ＩＰＡは，平成１９年４月，「大企業・中堅企業の情報システムのセキュリティ対策～脅威と対策」と題する文書において，ウェブアプリケーションに対する代表的な攻撃手法としてＳＱＬインジェクション攻撃を挙げ，ＳＱＬ文の組み立てにバインド機構を使用し，又はＳＱＬ文を構成する全ての変数に対しエスケープ処理を行うこと等により，ＳＱＬインジェクション対策をすることが必要である旨を明示していたことが認められ，これらの事実に照らすと，被告は，平成２１年２月４日の本件システム発注契約締結時点において，本件データベースから顧客の個人情報が漏洩することを防止するために，ＳＱＬインジェクション対策として，バインド機構の使用又はエスケープ処理を施したプログラムを提供すべき債務を負っていたということができる。

セキュリティ対策はやって当たり前

SQLインジェクションが原因でカード情報が流出した事件です。

判決のポイントは、

そこで検討するに，証拠（甲１４，２５，２９）によれば，経済産業省は，平成１８年２月２０日，「個人情報保護法に基づく個人データの安全管理措置の徹底に係る注意喚起」と題する文書において，ＳＱＬインジェクション攻撃によってデータベース内の大量の個人データが流出する事案が相次いで発生していることから，独立行政法人情報処理推進機構（以下「ＩＰＡ」という。）が紹介するＳＱＬインジェクション対策の措置を重点的に実施することを求める旨の注意喚起をしていたこと，ＩＰＡは，平成１９年４月，「大企業・中堅企業の情報システムのセキュリティ対策～脅威と対策」と題する文書において，ウェブアプリケーションに対する代表的な攻撃手法としてＳＱＬインジェクション攻撃を挙げ，ＳＱＬ文の組み立てにバインド機構を使用し，又はＳＱＬ文を構成する全ての変数に対しエスケープ処理を行うこと等により，ＳＱＬインジェクション対策をすることが必要である旨を明示していたことが認められ，これらの事実に照らすと，被告は，平成２１年２月４日の本件システム発注契約締結時点において，本件データベースから顧客の個人情報が漏洩することを防止するために，ＳＱＬインジェクション対策として，バインド機構の使用又はエスケープ処理を施したプログラムを提供すべき債務を負っていたということができる。

セキュリティ対策はやって当たり前

SQLインジェクションが原因でカード情報が流出した事件です。

判決のポイントは、

そこで検討するに，証拠（甲１４，２５，２９）によれば，経済産業省は，平成１８年２月２０日，「個人情報保護法に基づく個人データの安全管理措置の徹底に係る注意喚起」と題する文書において，ＳＱＬインジェクション攻撃によってデータベース内の大量の個人データが流出する事案が相次いで発生していることから，独立行政法人情報処理推進機構（以下「ＩＰＡ」という。）が紹介するＳＱＬインジェクション対策の措置を重点的に実施することを求める旨の注意喚起をしていたこと，ＩＰＡは，平成１９年４月，「大企業・中堅企業の情報システムのセキュリティ対策～脅威と対策」と題する文書において，ウェブアプリケーションに対する代表的な攻撃手法としてＳＱＬインジェクション攻撃を挙げ，ＳＱＬ文の組み立てにバインド機構を使用し，又はＳＱＬ文を構成する全ての変数に対しエスケープ処理を行うこと等により，ＳＱＬインジェクション対策をすることが必要である旨を明示していたことが認められ，これらの事実に照らすと，被告は，平成２１年２月４日の本件システム発注契約締結時点において，本件データベースから顧客の個人情報が漏洩することを防止するために，ＳＱＬインジェクション対策として，バインド機構の使用又はエスケープ処理を施したプログラムを提供すべき債務を負っていたということができる。

SQLインジェクション対策を施したプログラムを提供すべき債務

同時に、

「望ましい」は必須ではない

証拠（甲２４の１・２，甲２５）によれば，厚生労働省及び経済産業省が平成１９年３月３０日に改正した「個人情報の保護に関する法律についての経済産業分野を対象とするガイドライン」（同日厚生労働省・経済産業省告示第１号）では，クレジットカード情報等（クレジットカード情報を含む個人情報）について特に講じることが望ましい安全管理措置として，利用目的の達成に必要最小限の範囲の保存期間を設定し，保存場所を限定し，保存期間経過後適切かつ速やかに破棄することを例示し，ＩＰＡは，同年４月，前記「大企業・中堅企業の情報システムのセキュリティ対策～脅威と対策」と題する文書において，データベース内に格納されている重要なデータや個人情報については暗号化することが望ましいと明示していたことが認められる。しかし，上記告示等は，いずれも上記対策を講じることが「望ましい」と指摘するものにすぎないし，上記ＩＰＡの文書においては，データベース内のデータ全てに対して暗号化の処理を行うとサーバー自体の負荷になることがあるので，特定のカラムだけを暗号化するなどの考慮が必要であるとも指摘されている（甲２５）ように，暗号化の設定内容等は暗号化の程度によって異なり，それによって被告の作業量や代金も増減すると考えられることに照らすと，契約で特別に合意していなくとも，当然に，被告がクレジットカード情報を本件サーバー及びログに保存せず，若しくは保存しても削除する設定とし，又はクレジットカード情報を暗号化して保存すべき債務を負っていたとは認められない。

「望ましい」は必須ではない

証拠（甲２４の１・２，甲２５）によれば，厚生労働省及び経済産業省が平成１９年３月３０日に改正した「個人情報の保護に関する法律についての経済産業分野を対象とするガイドライン」（同日厚生労働省・経済産業省告示第１号）では，クレジットカード情報等（クレジットカード情報を含む個人情報）について特に講じることが望ましい安全管理措置として，利用目的の達成に必要最小限の範囲の保存期間を設定し，保存場所を限定し，保存期間経過後適切かつ速やかに破棄することを例示し，ＩＰＡは，同年４月，前記「大企業・中堅企業の情報システムのセキュリティ対策～脅威と対策」と題する文書において，データベース内に格納されている重要なデータや個人情報については暗号化することが望ましいと明示していたことが認められる。しかし，上記告示等は，いずれも上記対策を講じることが「望ましい」と指摘するものにすぎないし，上記ＩＰＡの文書においては，データベース内のデータ全てに対して暗号化の処理を行うとサーバー自体の負荷になることがあるので，特定のカラムだけを暗号化するなどの考慮が必要であるとも指摘されている（甲２５）ように，暗号化の設定内容等は暗号化の程度によって異なり，それによって被告の作業量や代金も増減すると考えられることに照らすと，契約で特別に合意していなくとも，当然に，被告がクレジットカード情報を本件サーバー及びログに保存せず，若しくは保存しても削除する設定とし，又はクレジットカード情報を暗号化して保存すべき債務を負っていたとは認められない。

「望ましい」は必須ではない

証拠（甲２４の１・２，甲２５）によれば，厚生労働省及び経済産業省が平成１９年３月３０日に改正した「個人情報の保護に関する法律についての経済産業分野を対象とするガイドライン」（同日厚生労働省・経済産業省告示第１号）では，クレジットカード情報等（クレジットカード情報を含む個人情報）について特に講じることが望ましい安全管理措置として，利用目的の達成に必要最小限の範囲の保存期間を設定し，保存場所を限定し，保存期間経過後適切かつ速やかに破棄することを例示し，ＩＰＡは，同年４月，前記「大企業・中堅企業の情報システムのセキュリティ対策～脅威と対策」と題する文書において，データベース内に格納されている重要なデータや個人情報については暗号化することが望ましいと明示していたことが認められる。しかし，上記告示等は，いずれも上記対策を講じることが「望ましい」と指摘するものにすぎないし，上記ＩＰＡの文書においては，データベース内のデータ全てに対して暗号化の処理を行うとサーバー自体の負荷になることがあるので，特定のカラムだけを暗号化するなどの考慮が必要であるとも指摘されている（甲２５）ように，暗号化の設定内容等は暗号化の程度によって異なり，それによって被告の作業量や代金も増減すると考えられることに照らすと，契約で特別に合意していなくとも，当然に，被告がクレジットカード情報を本件サーバー及びログに保存せず，若しくは保存しても削除する設定とし，又はクレジットカード情報を暗号化して保存すべき債務を負っていたとは認められない。

さて、何がやる必要の
ある対策なのか

そこで検討するに，証拠（甲１４，２５，２９）によれば，経済産業省は，平成１８年２月２０日，「個人情報保護法に基づく個人データの安全管理措置の徹底に係る注意喚起」と題する文書において，ＳＱＬインジェクション攻撃によってデータベース内の大量の個人データが流出する事案が相次いで発生していることから，独立行政法人情報処理推進機構（以下「ＩＰＡ」という。）が紹介するＳＱＬインジェクション対策の措置を重点的に実施することを求める旨の注意喚起をしていたこと，ＩＰＡは，平成１９年４月，「大企業・中堅企業の情報システムのセキュリティ対策～脅威と対策」と題する文書において，ウェブアプリケーションに対する代表的な攻撃手法としてＳＱＬインジェクション攻撃を挙げ，ＳＱＬ文の組み立てにバインド機構を使用し，又はＳＱＬ文を構成する全ての変数に対しエスケープ処理を行うこと等により，ＳＱＬインジェクション対策をすることが必要である旨を明示していたことが認められ，これらの事実に照らすと，被告は，平成２１年２月４日の本件システム発注契約締結時点において，本件データベースから顧客の個人情報が漏洩することを防止するために，ＳＱＬインジェクション対策として，バインド機構の使用又はエスケープ処理を施したプログラムを提供すべき債務を負っていたということができる。

そこで検討するに，証拠（甲１４，２５，２９）によれば，経済産業省は，平成１８年２月２０日，「個人情報保護法に基づく個人データの安全管理措置の徹底に係る注意喚起」と題する文書において，ＳＱＬインジェクション攻撃によってデータベース内の大量の個人データが流出する事案が相次いで発生していることから，独立行政法人情報処理推進機構（以下「ＩＰＡ」という。）が紹介するＳＱＬインジェクション対策の措置を重点的に実施することを求める旨の注意喚起をしていたこと，ＩＰＡは，平成１９年４月，「大企業・中堅企業の情報システムのセキュリティ対策～脅威と対策」と題する文書において，ウェブアプリケーションに対する代表的な攻撃手法としてＳＱＬインジェクション攻撃を挙げ，ＳＱＬ文の組み立てにバインド機構を使用し，又はＳＱＬ文を構成する全ての変数に対しエスケープ処理を行うこと等により，ＳＱＬインジェクション対策をすることが必要である旨を明示していたことが認められ，これらの事実に照らすと，被告は，平成２１年２月４日の本件システム発注契約締結時点において，本件データベースから顧客の個人情報が漏洩することを防止するために，ＳＱＬインジェクション対策として，バインド機構の使用又はエスケープ処理を施したプログラムを提供すべき債務を負っていたということができる。

そこで検討するに，証拠（甲１４，２５，２９）によれば，経済産業省は，平成１８年２月２０日，「個人情報保護法に基づく個人データの安全管理措置の徹底に係る注意喚起」と題する文書において，ＳＱＬインジェクション攻撃によってデータベース内の大量の個人データが流出する事案が相次いで発生していることから，独立行政法人情報処理推進機構（以下「ＩＰＡ」という。）が紹介するＳＱＬインジェクション対策の措置を重点的に実施することを求める旨の注意喚起をしていたこと，ＩＰＡは，平成１９年４月，「大企業・中堅企業の情報システムのセキュリティ対策～脅威と対策」と題する文書において，ウェブアプリケーションに対する代表的な攻撃手法としてＳＱＬインジェクション攻撃を挙げ，ＳＱＬ文の組み立てにバインド機構を使用し，又はＳＱＬ文を構成する全ての変数に対しエスケープ処理を行うこと等により，ＳＱＬインジェクション対策をすることが必要である旨を明示していたことが認められ，これらの事実に照らすと，被告は，平成２１年２月４日の本件システム発注契約締結時点において，本件データベースから顧客の個人情報が漏洩することを防止するために，ＳＱＬインジェクション対策として，バインド機構の使用又はエスケープ処理を施したプログラムを提供すべき債務を負っていたということができる。

復習しましょう。

裁判で負けないための努力

「裁判対策」としては、下記のようなドキュメントに書いてあることは、「やって当たり前」という扱いを受けます。

裁判で負けないための努力

「裁判対策」としては、下記のようなドキュメントに書いてあることは、「やって当たり前」という扱いを受けます。

法律・政令・省令

裁判で負けないための努力

「裁判対策」としては、下記のようなドキュメントに書いてあることは、「やって当たり前」という扱いを受けます。

法律・政令・省令
経済産業省のガイドライン

裁判で負けないための努力

「裁判対策」としては、下記のようなドキュメントに書いてあることは、「やって当たり前」という扱いを受けます。

法律・政令・省令
経済産業省のガイドライン
IPAのガイドライン

裁判で負けないための努力

「裁判対策」としては、下記のようなドキュメントに書いてあることは、「やって当たり前」という扱いを受けます。

法律・政令・省令
経済産業省のガイドライン
IPAのガイドライン
業界団体のガイドライン

Webシステムが受ける攻撃とは

情報セキュリティ10大脅威

表 2.1 情報セキュリティ 10 大脅威 2017「個人」および「組織」向けの脅威の順位

「個人」向け脅威	順位	「組織」向け脅威
インターネットバンキングや クレジットカード情報の不正利用	1	標的型攻撃による情報流出
ランサムウェアによる被害	2	ランサムウェアによる被害
スマートフォンやスマートフォンアプリを 狙った攻撃	3	ウェブサービスからの個人情報の窃取
ウェブサービスへの不正ログイン	4	サービス妨害攻撃によるサービスの停止
ワンクリック請求等の不当請求	5	内部不正による情報漏えいと それに伴う業務停止
ウェブサービスからの個人情報の窃取	6	ウェブサイトの改ざん
ネット上の誹謗・中傷	7	ウェブサービスへの不正ログイン
情報モラル欠如に伴う犯罪の低年齢化	8	IoT 機器の脆弱性の顕在化
インターネット上のサービスを悪用した攻撃	9	攻撃のビジネス化 （アンダーグラウンドサービス）
IoT 機器の不適切な管理	10	インターネットバンキングや クレジットカード情報の不正利用

情報セキュリティ10大脅威

表 2.1 情報セキュリティ 10 大脅威 2017「個人」および「組織」向けの脅威の順位

「個人」向け脅威	順位	「組織」向け脅威
インターネットバンキングや クレジットカード情報の不正利用	1	標的型攻撃による情報流出
ランサムウェアによる被害	2	ランサムウェアによる被害
スマートフォンやスマートフォンアプリを 狙った攻撃	3	ウェブサービスからの個人情報の窃取
ウェブサービスへの不正ログイン	4	サービス妨害攻撃によるサービスの停止
ワンクリック請求等の不当請求	5	内部不正による情報漏えいと それに伴う業務停止
ウェブサービスからの個人情報の窃取	6	ウェブサイトの改ざん
ネット上の誹謗・中傷	7	ウェブサービスへの不正ログイン
情報モラル欠如に伴う犯罪の低年齢化	8	IoT 機器の脆弱性の顕在化
インターネット上のサービスを悪用した攻撃	9	攻撃のビジネス化 （アンダーグラウンドサービス）
IoT 機器の不適切な管理	10	インターネットバンキングや クレジットカード情報の不正利用

OWASP Top10

OWASP (The Open Web Application Security Project) が発表する、重要な脅威をまとめたドキュメント。リスクの高さや攻撃シナリオもまとまっているので、非常に参考になる。

https://www.owasp.org/images/7/79/OWASP_Top_10_2013_JPN.pdf

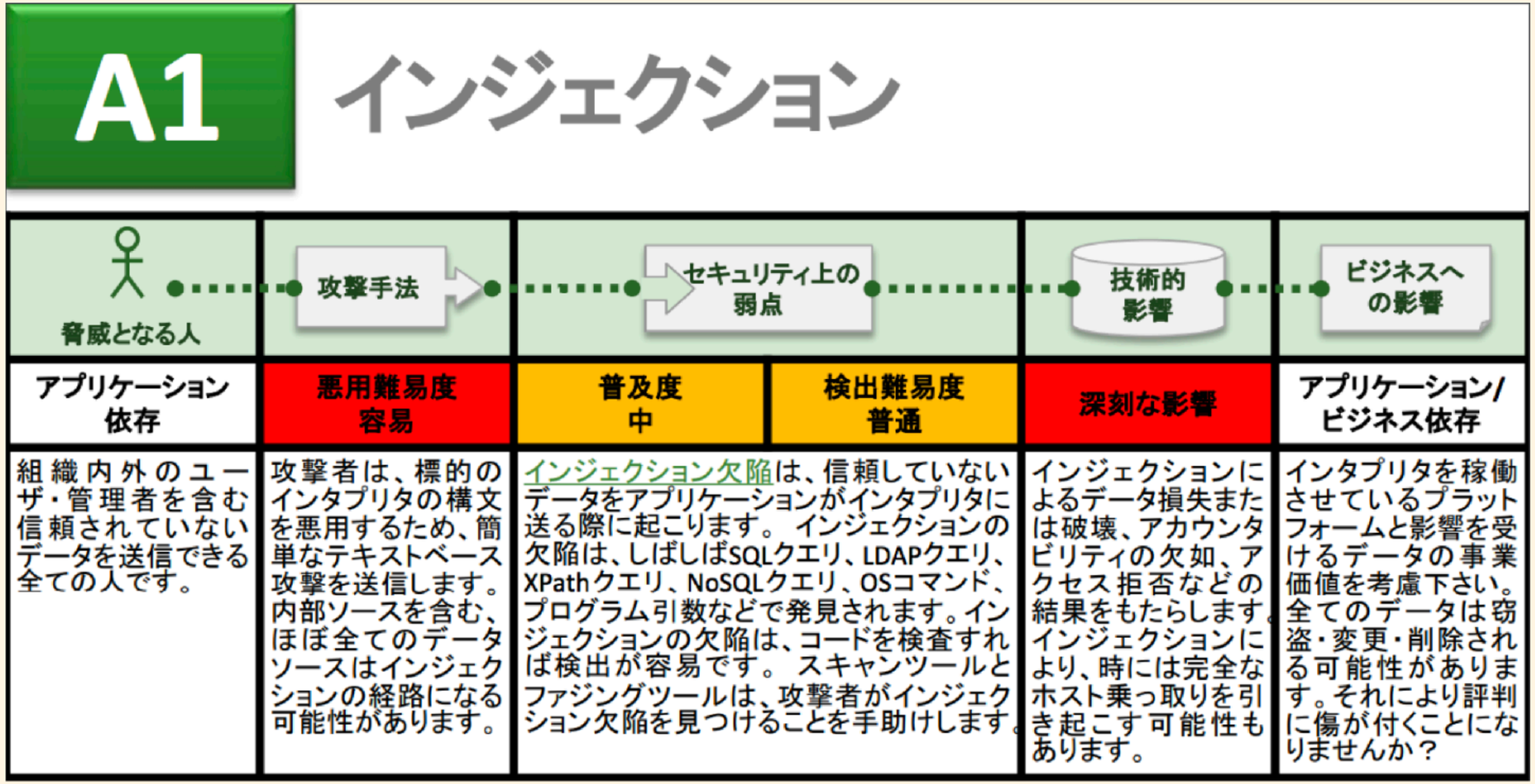
OWASP Top10 (2013)

A1 – インジェクション	SQLとOS、LDAPなどのインジェクション欠陥は、信頼されていないデータがコマンド又はクエリの一部としてインタプリタに送信される際に発生します。攻撃者の悪意のあるデータは、意図しないコマンドの実行や適切な権限のないデータアクセスをさせるように、インタプリタを騙すことができます。
A2 –認証とセッション管理の不備	認証とセッション管理に関連するアプリケーション機能は、しばしば正しく実装されていません。そのため、攻撃者はパスワード又は鍵、セッショントークンを漏洩させたり、他の実装の不備を悪用してなりすますことができます。
A3 – クロスサイトスクリプティング (XSS)	XSS欠陥は、アプリケーションが信頼されていないデータを受け取る時、適切な検証やエスケープをせずに、ウェブブラウザに送信する際に発生します。XSSにより、攻撃者は被害者のブラウザでスクリプトを実行でき、ユーザセッションのハイジャックやウェブサイトの改竄、またユーザを悪意のあるサイトにリダイレクトすることが可能です。
A4 –安全でないオブジェクト直接参照	オブジェクトの直接参照は、開発者がファイル又はディレクトリ、データベースのキーなど、内部に実装されているオブジェクトへの参照を公開する際に発生します。アクセス制御チェックや他の保護が無ければ、攻撃者はこれらの参照を用い、アクセス権限のないデータへアクセスすることができます。
A5 – セキュリティ設定のミス	適切なセキュリティには、アプリケーション、フレームワーク、アプリケーションサーバ、Webサーバ、データベースサーバ、およびプラットフォームに対して、セキュアな設定を定義して反映させる事が必要です。製品のデフォルト設定は安全ではない場合が多いため、セキュアな設定が定義・実装・維持されないといけません。また、ソフトウェアを最新の状態に保つ必要があります。

OWASP Top10 (2013)

A6 – 機密データの露出	多くのWebアプリケーションは、クレジットカードや納税者番号、認証情報などの機密データを適切に保護していません。攻撃者は、これらの脆弱に保護されているデータを窃取・改変して、クレジットカード詐欺や個人情報盗難などの犯罪を犯します。機密データは、保存時や転送時に関わらず、特別な保護をされるべきです。また、ブラウザとの間で取り交わされる際に、特別な注意も払うべきです。
A7 – 機能レベルアクセス制御の欠落	ほとんどのWebアプリケーションは、機能をUIで表示する前に、機能レベルのアクセス権限を検証しています。但し、各機能がアクセスされる際に、アプリケーションはサーバ上で、同じアクセス制御チェックを実行しないといけません。リクエストが検証されていない場合、適切な承認もされていないため、攻撃者はリクエストを偽造して、狙った機能にアクセスできます。
A8 – クロスサイトリクエストフォージェリ (CSRF)	CSRF攻撃は、脆弱性のあるWebアプリケーションに対し、ログオンしている被害者のブラウザから、偽造されたHTTPリクエストを送信させます。そのリクエストには、被害者のセッションクッキーや他の自動的に組み込まれた認証情報が含まれています。これにより、攻撃者は脆弱性のあるアプリケーションに、ユーザからの正当なリクエストとして認識されるリクエストを被害者のブラウザに生成させることができます。
A9 – 既知の脆弱性を持つコンポーネントの使用	ライブラリ、フレームワーク、および他のソフトウェアモジュールなどのコンポーネントは、ほとんど常にフル権限で実行されます。脆弱なコンポーネントが悪用される場合、深刻なデータ損失やサーバ乗っ取りまでに至る攻撃ができます。既知の脆弱性を持つコンポーネントを使用するアプリケーションは、アプリケーションの防御を弱体化し、様々な攻撃と影響も可能になります。
A10 – 未検証のリダイレクトとフォーワード	Webアプリケーションは、頻繁にユーザを他の画面やウェブサイトへリダイレクト・フォーワードしますが、信頼されていないデータを用いて、転送先画面を決定しています。適切な検証がないと、攻撃者は被害者をフィッシングサイトやマルウェアサイトへリダイレクトできたり、フォーワードで閲覧権限のない画面へアクセスできます。

OWASP Top10 (2013)



OWASP Top10 (2013)

脆弱性有無の確認

アプリケーションにインジェクション欠陥があるかどうかを確認する最善の方法は、使用されている全てのインタプリタが信頼出来ないデータをコマンドやクエリから区別しているかどうかを確認することです。SQLを発行する場合に、全てのプリペアドステートメントやストアドプロシージャにバインド変数を使用し、動的なクエリを避けることを意味します。

コードレビューは、アプリケーションが安全にインタプリタを使用しているかどうかを確認する迅速かつ正確な方法です。コード分析ツールを用いて、セキュリティアナリストはアプリケーションのインタプリタの使用を検出し、データフローをトレースできます。ペネトレーションテスターは、脆弱性を確認する巧妙なエクスプロイトによりこれらの問題を検証できます。

アプリケーションに対する動的自動化スキャンは、悪用できるインジェクション欠陥の存在を明らかにします。スキャナーは、インタプリタに到達できない事があり、攻撃の成否の判断が困難な場合かあります。不十分なエラー処理は、インジェクション欠陥を簡単に検出されるようにします。

防止方法

インジェクション攻撃を防止するには、コマンドとクエリから信頼出来ないデータを常に区別することが必要です。

1. 推奨されるオプションは、インタープリタを全く用いない安全なAPIを利用するか、パラメータ化されたインターフェースを用いる事です。ただ、ストアードプロシージャなどの、パラメータ化していてもインジェクション攻撃が可能なAPIには注意して下さい。
2. パラメータ化されたAPIが利用出来ない場合、インタプリタにて定められたエスケープ構文を用いて特殊文字のエスケープ処理を慎重に実施すべきです。OWASP's ESAPIはこれらの定番のエスケープを多く提供します。
3. 多くのアプリケーションが特殊文字の入力を必要とするため、"ホワイトリスト"による入力検証も推奨しますが、完全な防御ではありません。特殊文字が必要な場合、上述の1.と2.を用いることで、安全に使用出来ます。OWASP's ESAPIは、ホワイトリストの入力検証ルーチンの拡張可能なライブラリがあります。

攻撃シナリオの例

シナリオ #1: あるアプリケーションは信頼出来ないデータを用いて以下の脆弱なSQL呼び出しを生成します。

```
String query = "SELECT * FROM accounts WHERE custID='" + request.getParameter("id") + "'";
```

シナリオ #2: 同様に、あるアプリケーションのフレームワークに対する盲目的な信頼もまた、脆弱なクエリになりえます。（例えば、Hibernateクエリ言語（HQL））

```
Query HQLQuery = session.createQuery("FROM accounts WHERE custID='" + request.getParameter("id") + "'");
```

両方も攻撃者がブラウザで、パラメータ'id'を' or '1'='1で送信します。

```
http://example.com/app/accountView?id=' or '1'='1
```

両方のクエリの意味が変えられ、アカウントテーブルにあるレコードは全て返されます。データの改ざん、ストアドプロシージャの呼び出しなど、より危険な攻撃も可能です。

参考資料

OWASP

- OWASP SQL Injection Prevention Cheat Sheet
- OWASP Query Parameterization Cheat Sheet
- OWASP Command Injection Article
- OWASP XML eXternal Entity (XXE) Reference Article
- ASVS: Output Encoding/EscAPIng Requirements (V6)
- OWASP Testing Guide: Chapter on SQL Injection Testing

その他

- CWE Entry 77 on Command Injection
- CWE Entry 89 on SQL Injection
- CWE Entry 564 on Hibernate Injection

やるべき対策とは

安全なウェブサイトの作り方

http://www.ipa.go.jp/security/vuln/websecurity.html

安全なウェブサイトの作り方

開発時に実施すべきセキュリティ対策についてまとめられています。

安全なウェブサイトの作り方　チェックリスト

■ ウェブアプリケーションのセキュリティ実装 チェックリスト（1/3）

No	脆弱性の種類	対策の性質	チェック	実施項目	解説
1	SQLインジェクション	根本的解決	※ □ 対応済 □ 未対策 □ 対応不要	□ SQL文の組み立ては全てプレースホルダで実装する。	1-(i)-a
				□ SQL文の構成を文字列連結により行う場合は、アプリケーションの変数をSQL文のリテラルとして正しく構成する。	1-(i)-b
		根本的解決	□ 対応済 □ 未対策 □ 対応不要	ウェブアプリケーションに渡されるパラメータにSQL文を直接指定しない。	1-(ii)
		保険的対策	□ 対応済 □ 未対策 □ 対応不要	エラーメッセージをそのままブラウザに表示しない。	1-(iii)
		保険的対策	□ 対応済 □ 未対策 □ 対応不要	データベースアカウントに適切な権限を与える。	1-(iv)
2	OSコマンド・インジェクション	根本的解決	□ 対応済 □ 未対策 □ 対応不要	□ シェルを起動できる言語機能の利用を避ける。	2-(i)
		保険的対策	□ 対応済 □ 未対策 □ 対応不要	□ シェルを起動できる言語機能を利用する場合は、その引数を構成する全ての変数に対してチェックを行い、あらかじめ許可した処理のみを実行する。	2-(ii)

脆弱性の見つけ方

ウェブ健康診断仕様

ウェブアプリの脆弱性検査の方法についてまとめられています。

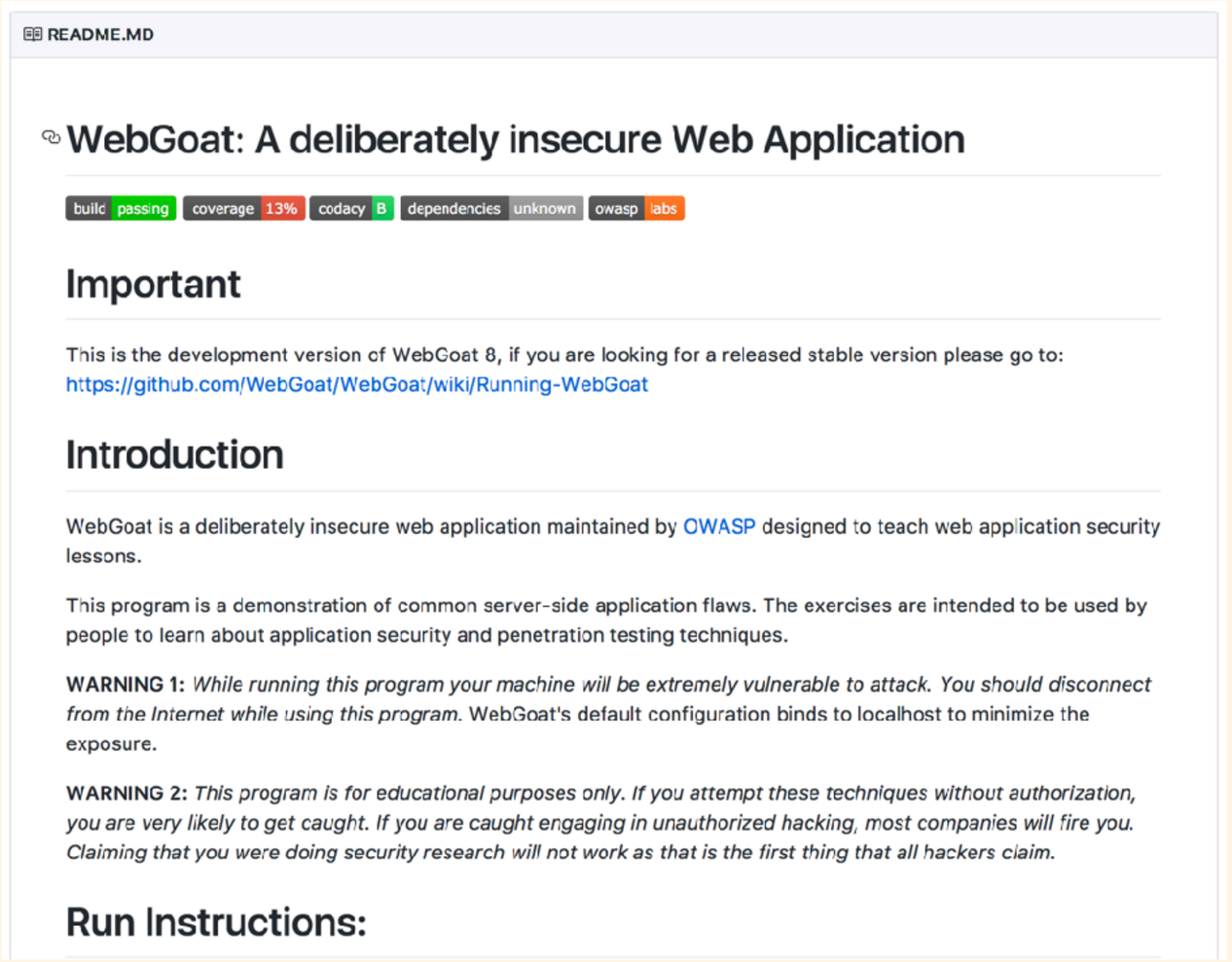
脆弱なサイトで脆弱性を見つけてみよう

WebGoat

https://github.com/WebGoat/WebGoat

お手軽に脆弱性を試せる
Webアプリケーション。

とっても脆弱！



README.MD

WebGoat: A deliberately insecure Web Application

build passing coverage 13% codacy B dependencies unknown owasp labs

Important

This is the development version of WebGoat 8, if you are looking for a released stable version please go to:
https://github.com/WebGoat/WebGoat/wiki/Running-WebGoat

Introduction

WebGoat is a deliberately insecure web application maintained by OWASP designed to teach web application security lessons.

This program is a demonstration of common server-side application flaws. The exercises are intended to be used by people to learn about application security and penetration testing techniques.

WARNING 1: *While running this program your machine will be extremely vulnerable to attack. You should disconnect from the Internet while using this program.* WebGoat's default configuration binds to localhost to minimize the exposure.

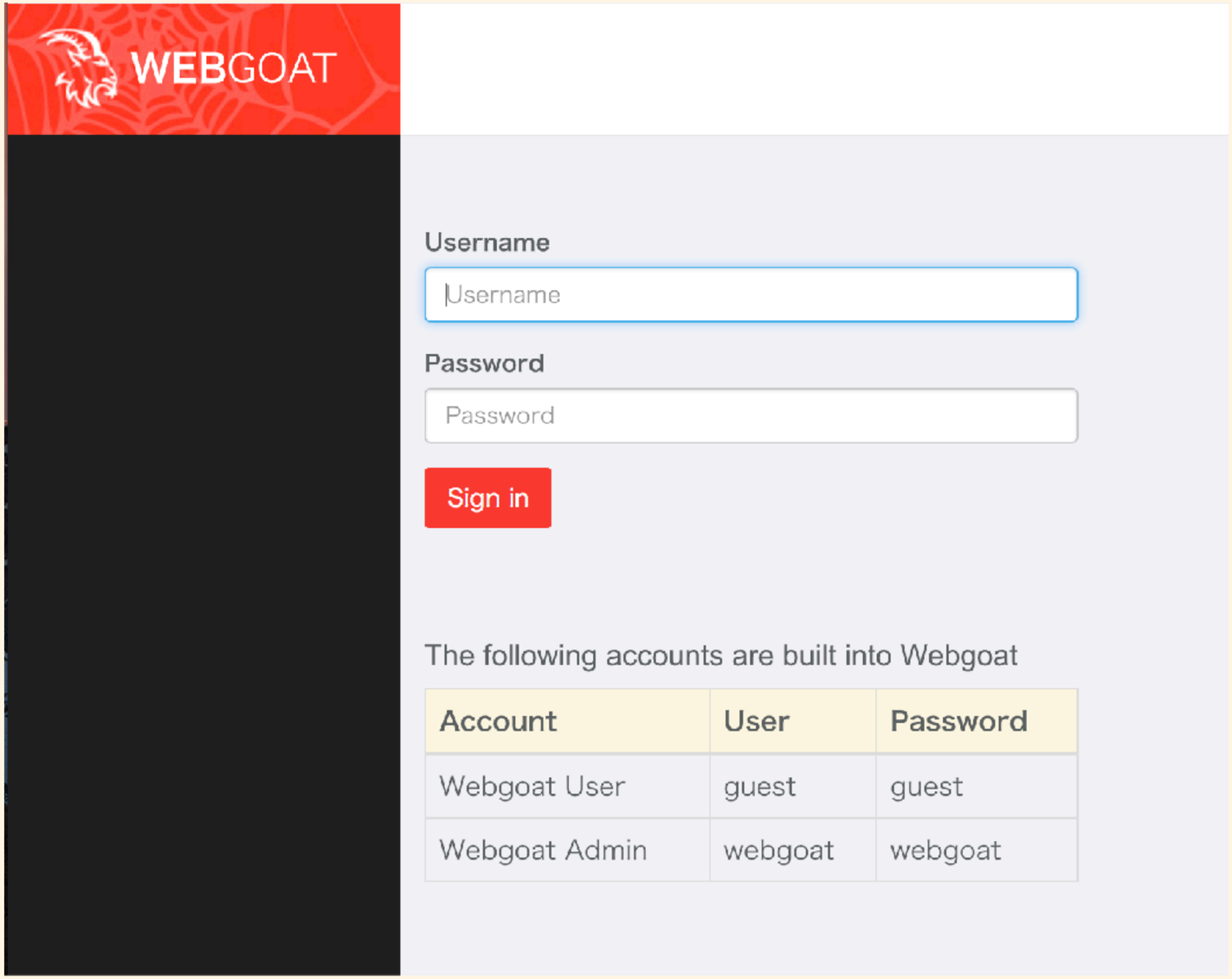
WARNING 2: *This program is for educational purposes only. If you attempt these techniques without authorization, you are very likely to get caught. If you are caught engaging in unauthorized hacking, most companies will fire you. Claiming that you were doing security research will not work as that is the first thing that all hackers claim.*

Run Instructions:

脆弱すぎて外に立てるのは
憚られるので

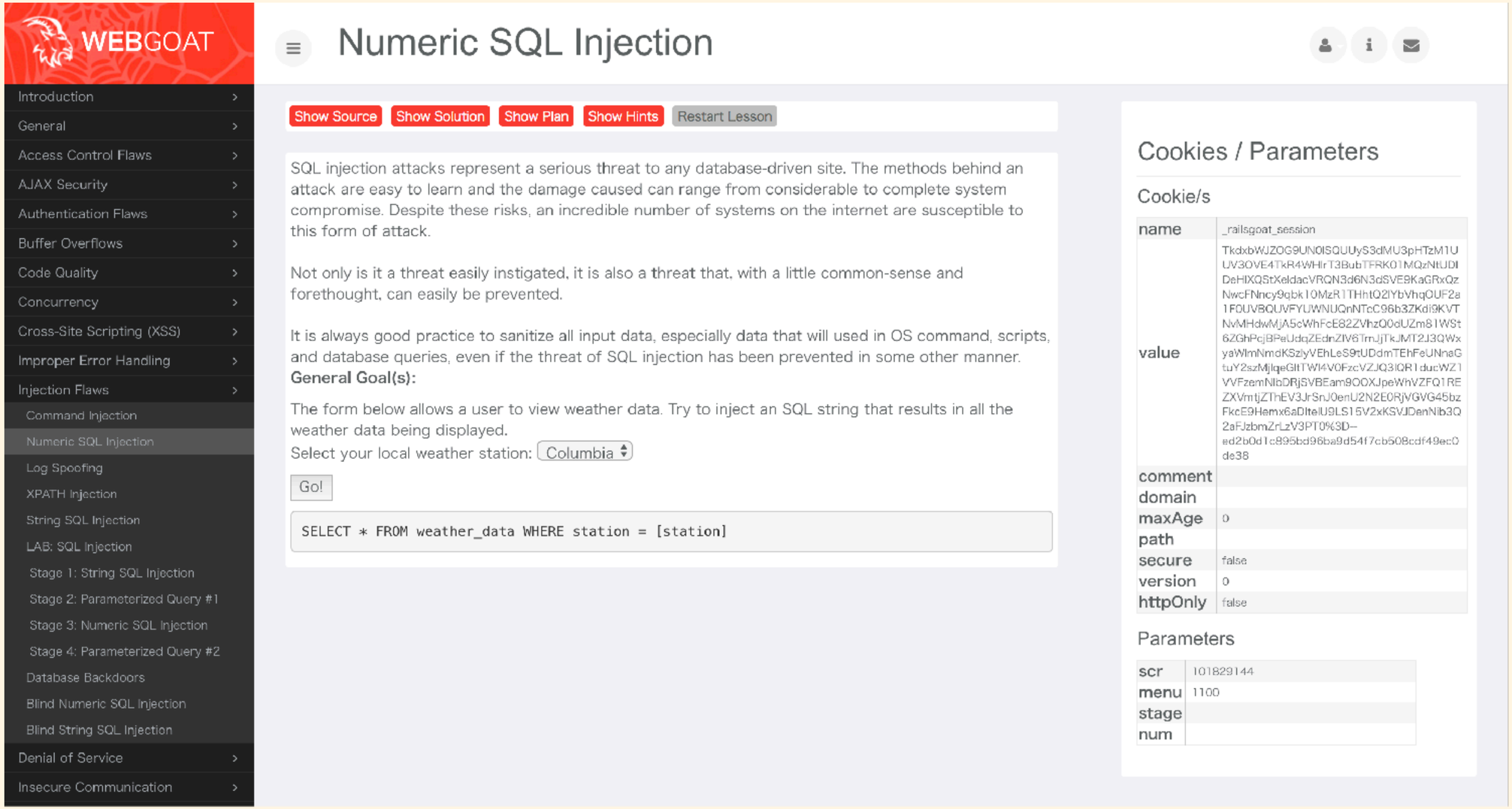
オススメはDockerで！

docker run -p 8080:8080 webgoat/webgoat-7.1



http://localhost:8080/WebGoat/

guest / guestでログインしたら



メニューから試したい脆弱性を探して遊ぼう！

脆弱性チェックツール

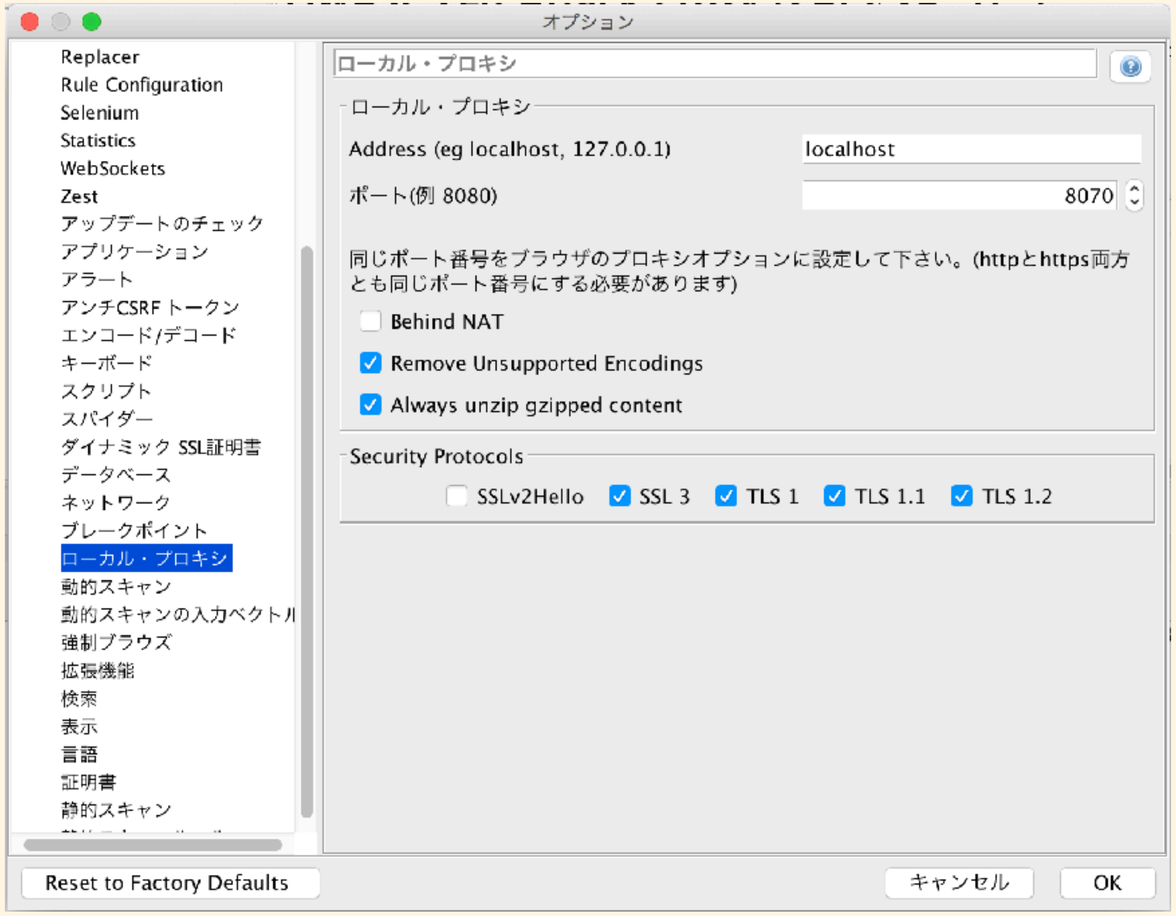
OWASP ZAP

OWASPが提供するツール。

https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

OWASP ZAPの使い方

ローカルプロキシ編（1）

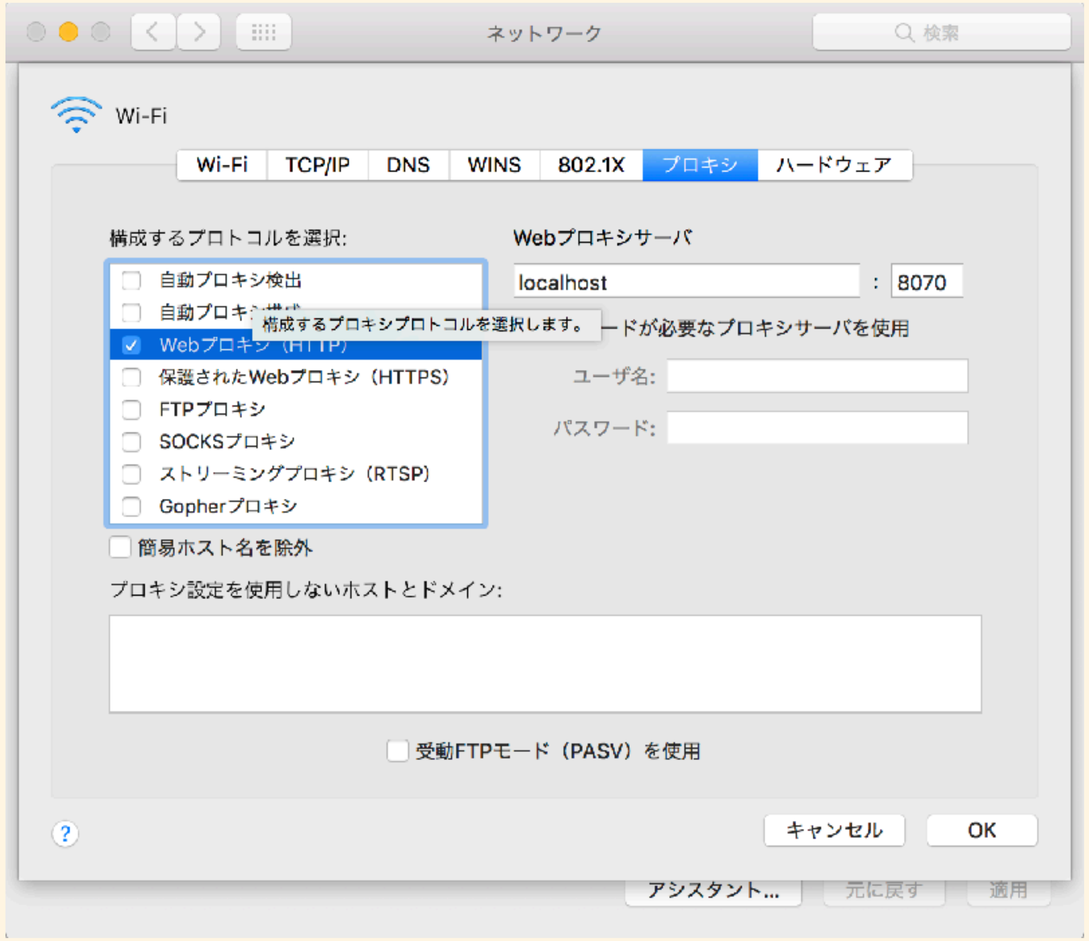


ZAPのオプションから、「ローカル・プロキシ」を設定する。

先程のWebGoatが8080で動いていると思うので、重ならないポートで動かしましょう！（例では8070）

OWASP ZAPの使い方

ローカルプロキシ編（2）

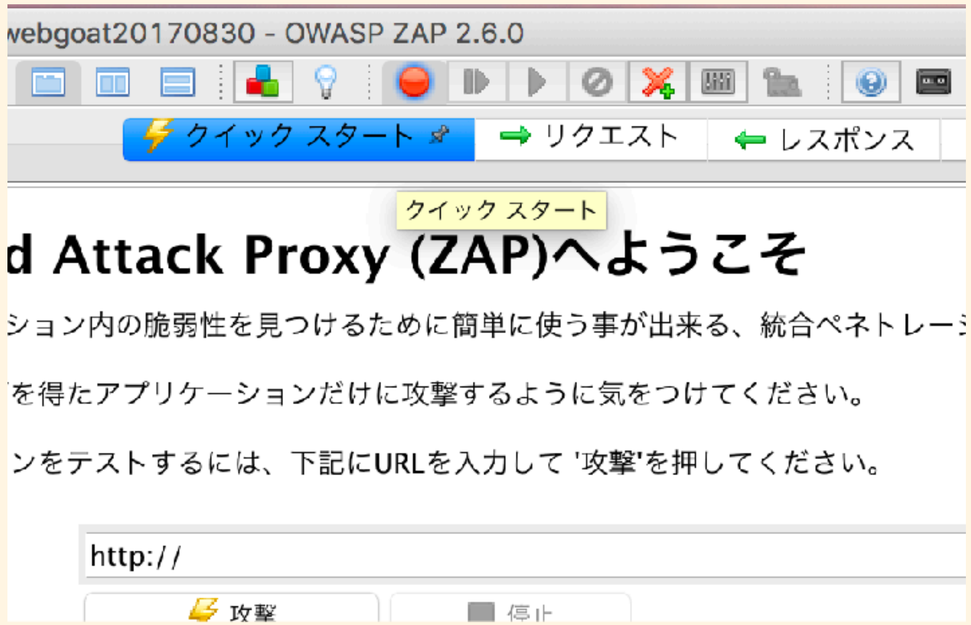


ネットワーク設定からWebプロキシを設定。先程のZAPで設定したポートを指定して下さい。

OWASP ZAPの使い方

ローカルプロキシ編（3）

ブレーク設定をON



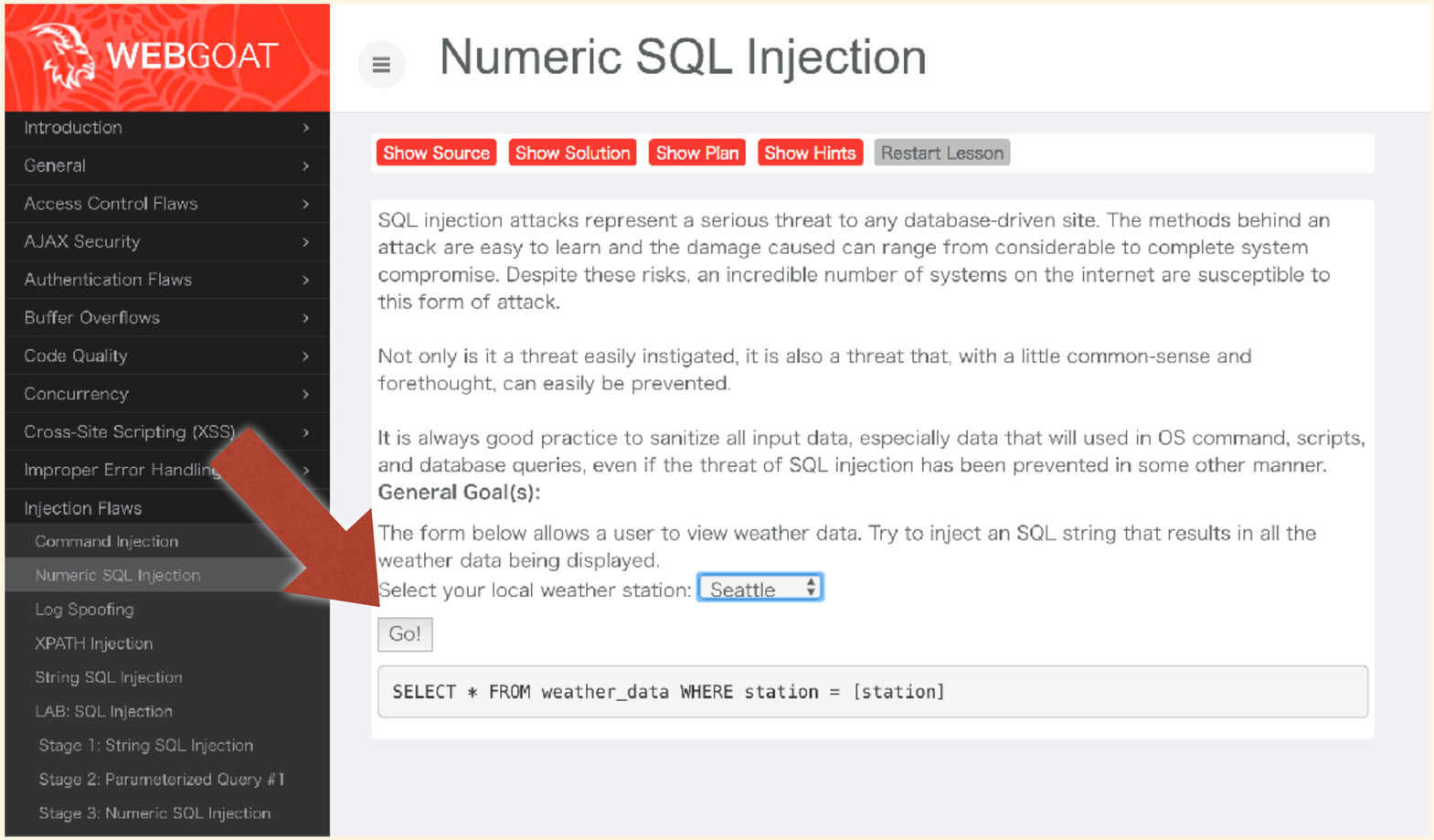
OWASP ZAPの使い方

ローカルプロキシ編（3）

ブレーク設定をON

OWASP ZAPの使い方

ローカルプロキシ編（４）



ブラウザから
アクセスを実行

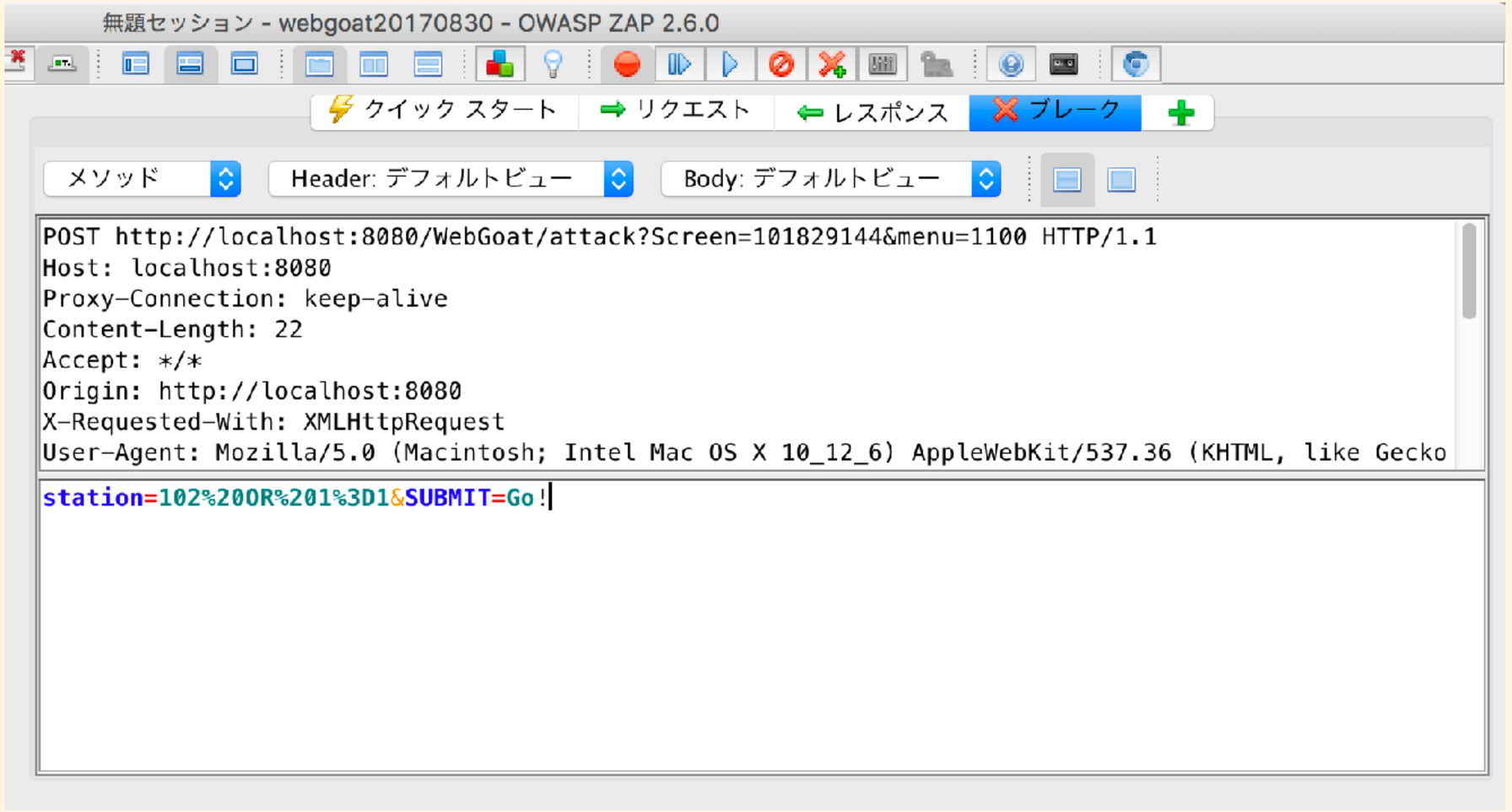
OWASP ZAPの使い方

ローカルプロキシ編（5）

アクセスを捕獲！

OWASP ZAPの使い方

ローカルプロキシ編（6）



OWASP ZAPの使い方

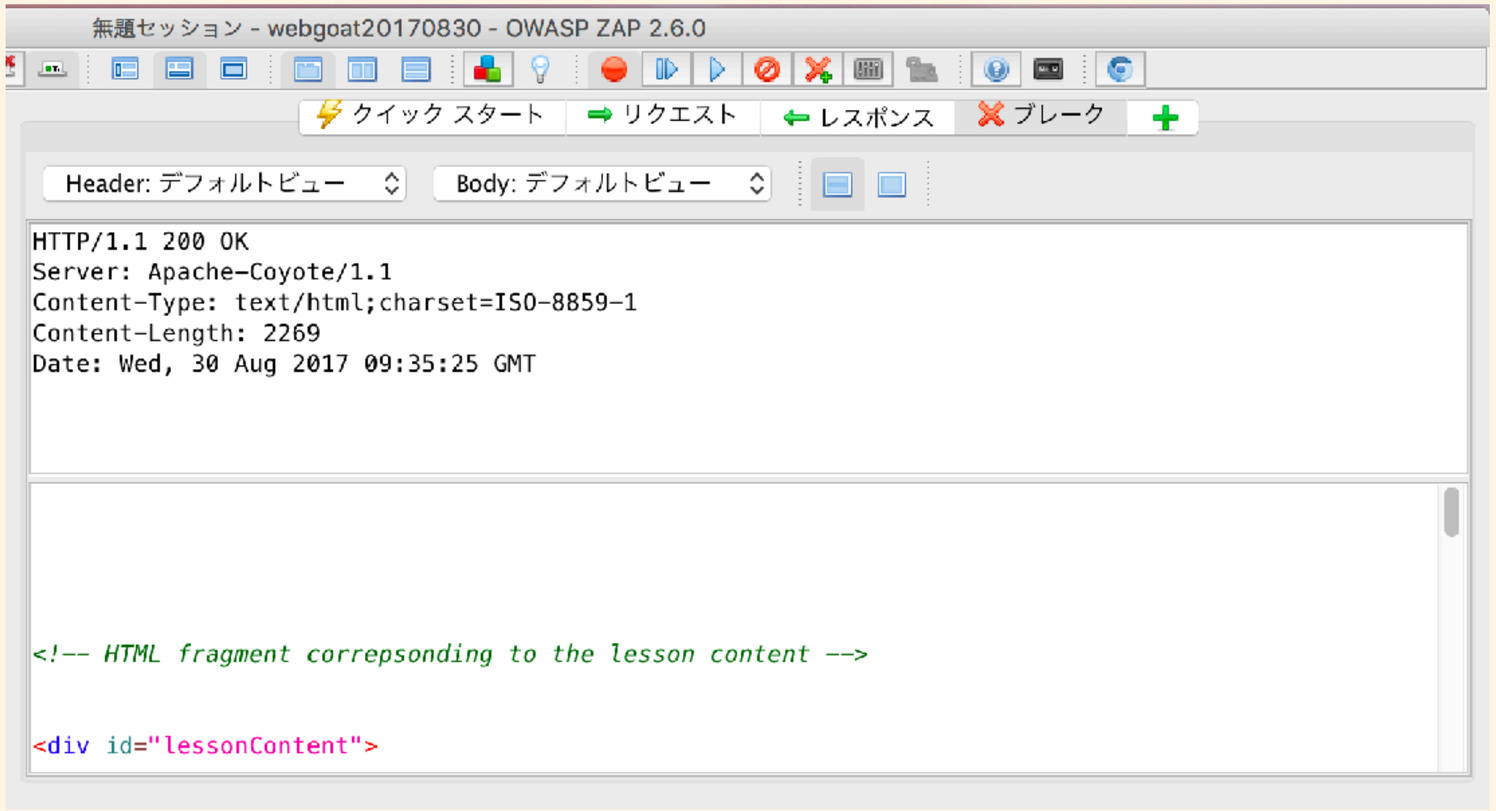
ローカルプロキシ編（6）

OWASP ZAPの使い方

ローカルプロキシ編（6）

OWASP ZAPの使い方

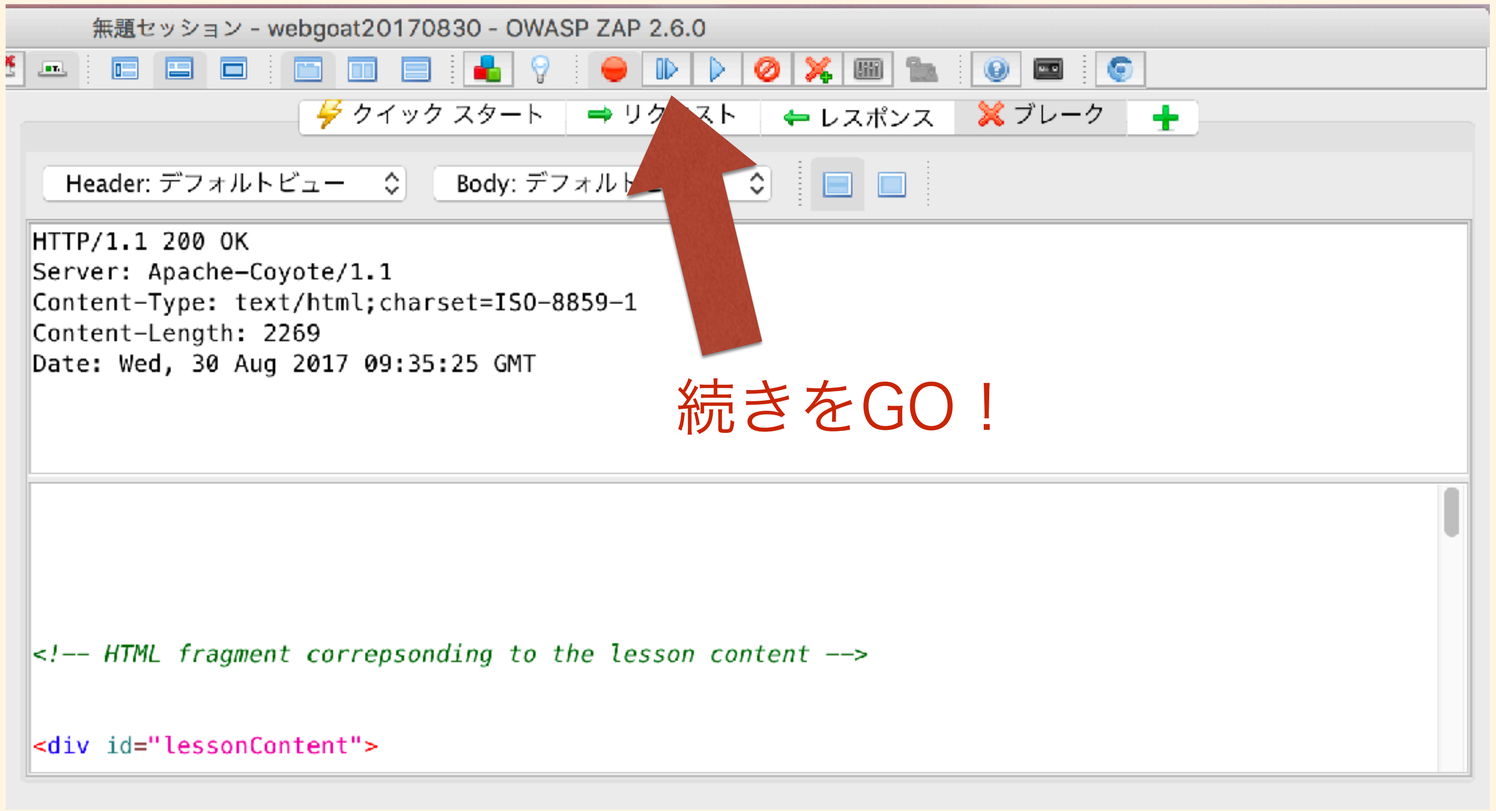
ローカルプロキシ編（7）



レスポンスが返ってくる！

OWASP ZAPの使い方

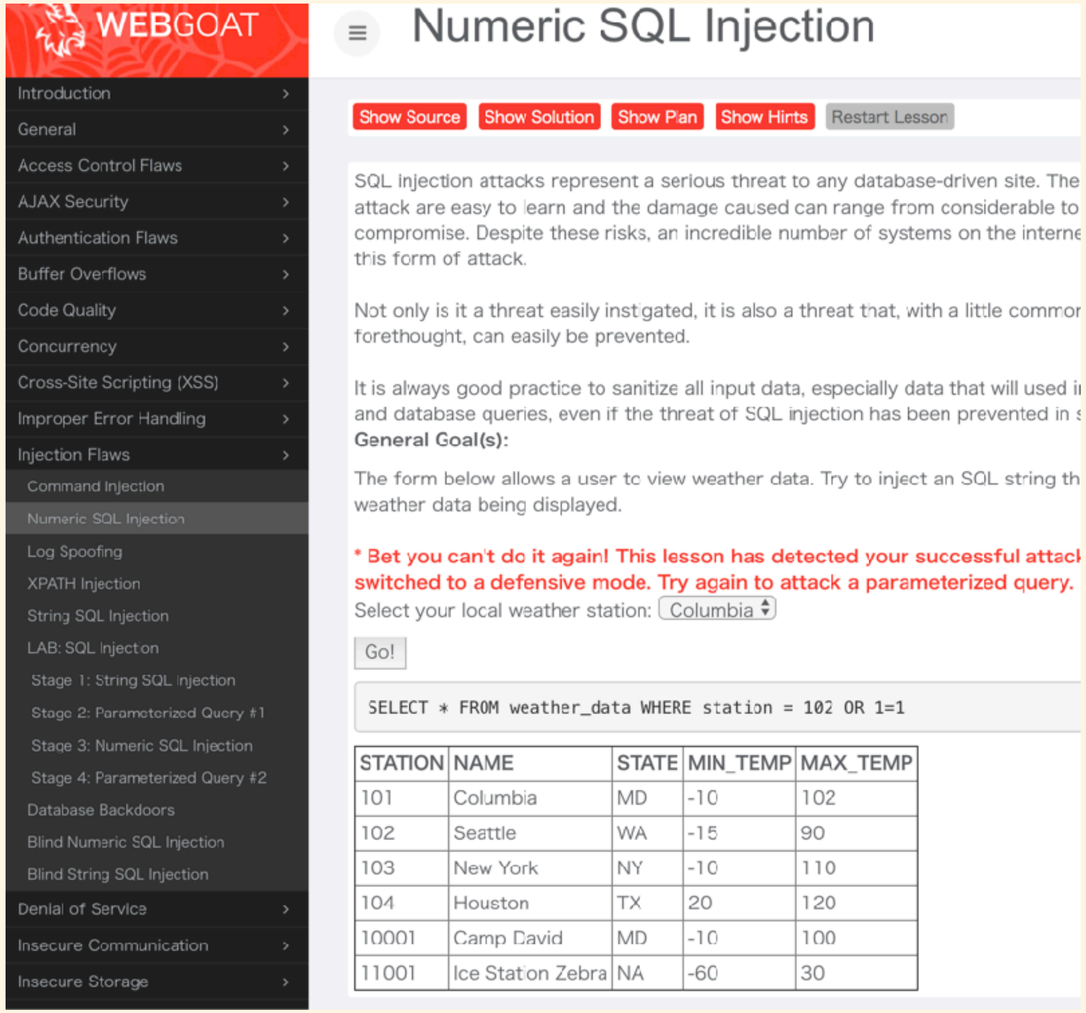
ローカルプロキシ編（7）



レスポンスが返ってくる！

OWASP ZAPの使い方

ローカルプロキシ編（8）



ブラウザにレスポンスが表示される！

お手軽にいろんな脆弱性をツールで
チェックしてくれたらいいのに！

OWASP ZAPの使い方

セキュリティスキャン編（1）

OWASP ZAPの使い方

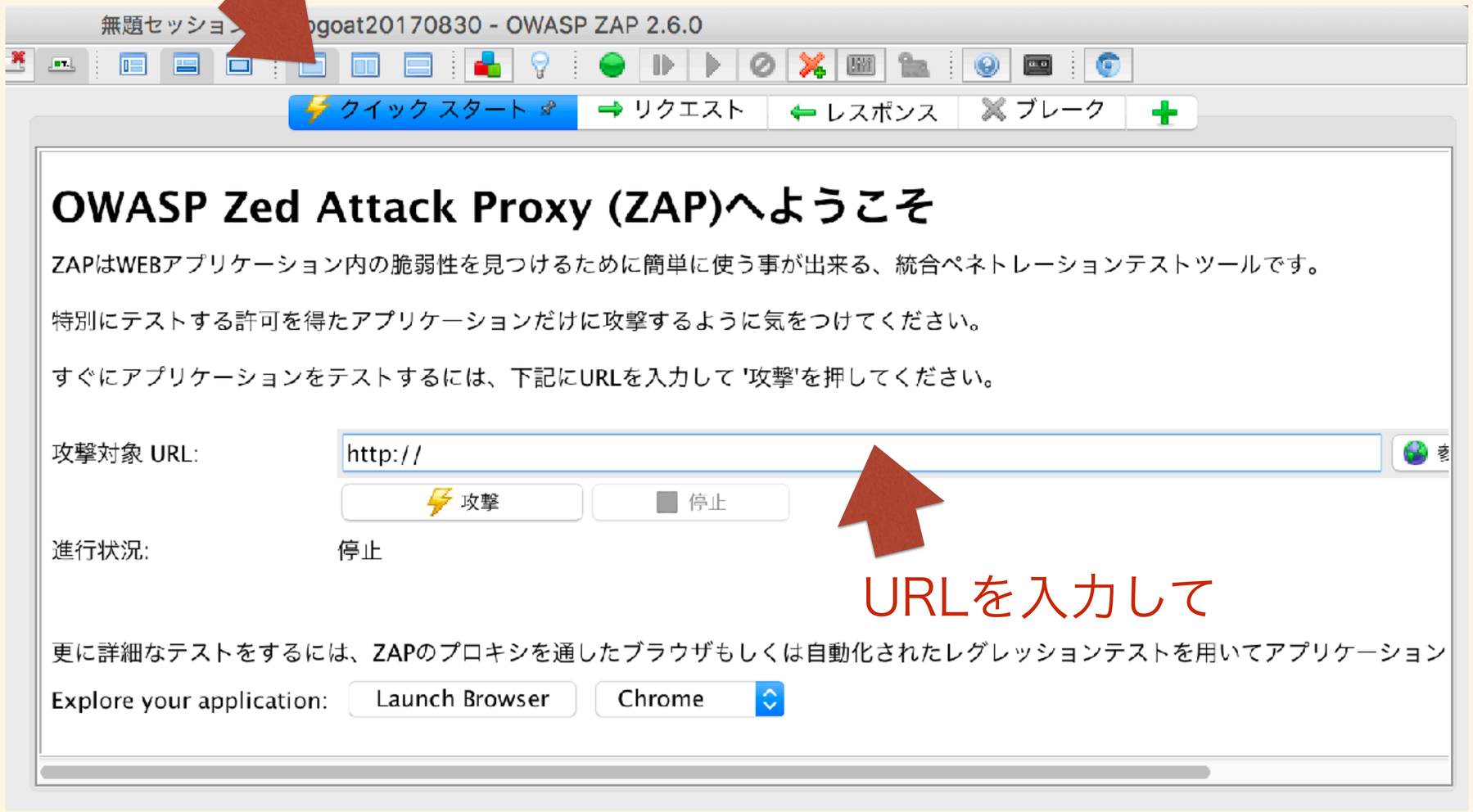
セキュリティスキャン編（1）

クイックスタートから

OWASP ZAPの使い方

セキュリティスキャン編（1）

クイックスタートから

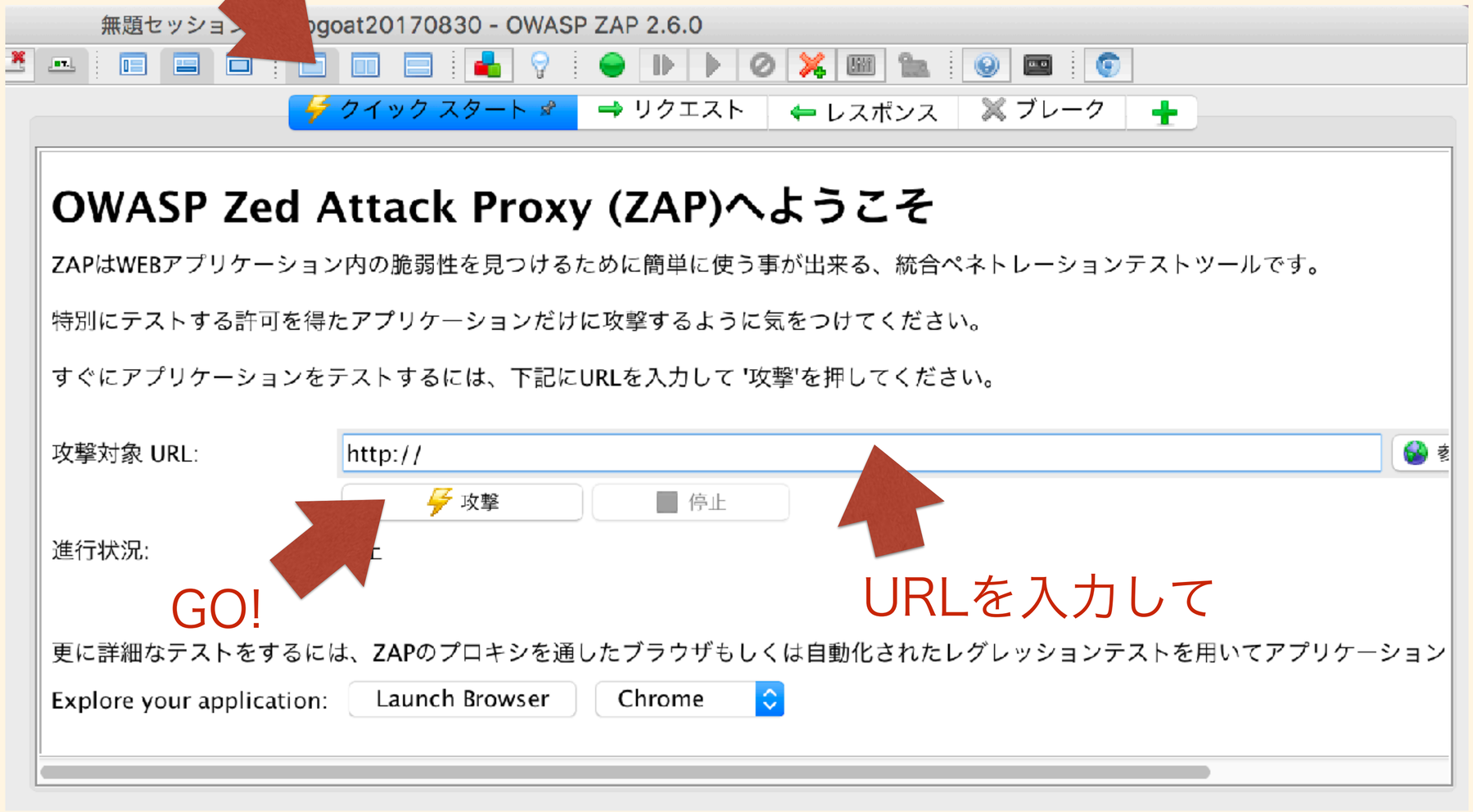


URLを入力して

OWASP ZAPの使い方

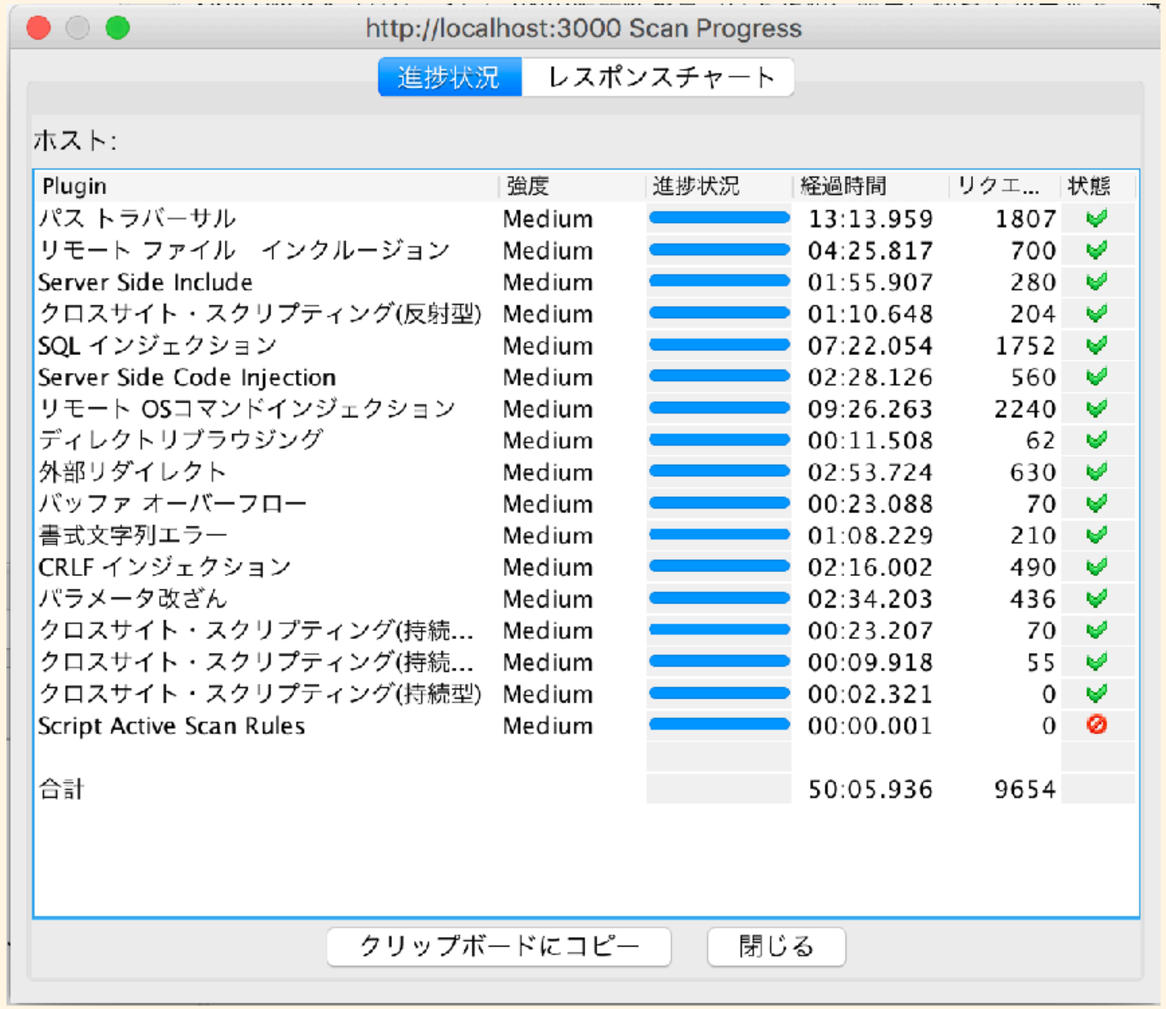
セキュリティスキャン編（1）

クイックスタートから



OWASP ZAPの使い方

セキュリティスキャン編（2）



時間は結構かかる。
（当然サイト規模による）

これはrubygoatという別のGoatでの結果。
約50分。