

セキュリティ勉強会

実践編

2017/08/30

目次

1. 例のアレ
2. Webシステムが受ける攻撃とは
3. やるべき対策とは
4. 脆弱性の見つけ方とは

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。

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、これらの事実を照らすと、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えることができる。

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、これらの事実を照らすと、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えることができる。

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、これらの事実を照らすと、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える。

**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、これらの事実を照らすと、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えることができる。

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、これらの事実を照らすと、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えることができる。

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、これらの事実を照らすと、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えることができる。

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

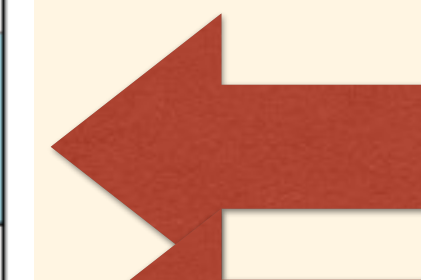
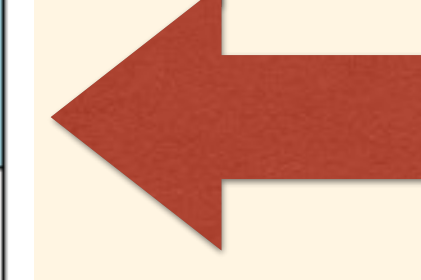
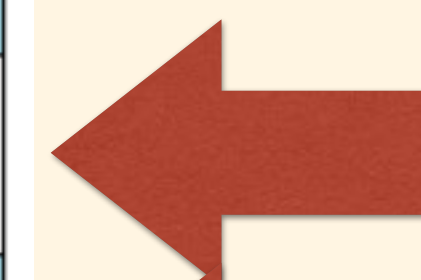
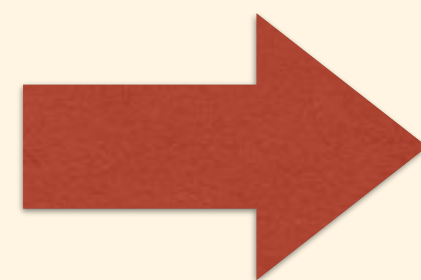
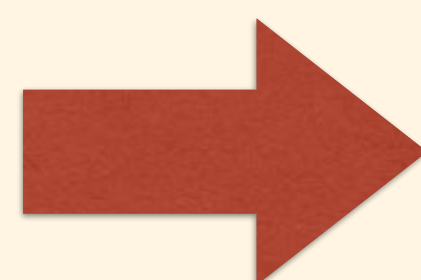
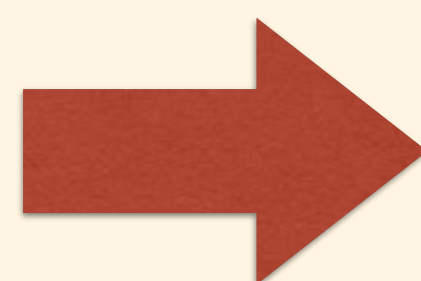
表 2.1 情報セキュリティ10大脅威 2017 「個人」および「組織」向けの脅威の順位

「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せん。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。




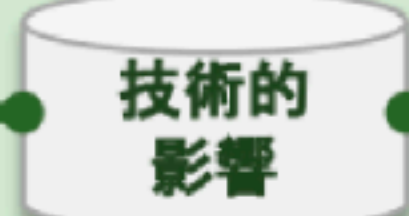

A10 – 未検証のリダイレクトとフォワード

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)

A1

インジェクション

 脅威となる人	 攻撃手法	 セキュリティ上の弱点		 技術的影響	 ビジネスへの影響
アプリケーション依存	悪用難易度容易	普及度中	検出難易度普通	深刻な影響	アプリケーション/ビジネス依存
組織内外のユーザ・管理者を含む信頼されていないデータを送信できる全ての人です。	攻撃者は、標的のインタプリタの構文を悪用するため、簡単なテキストベース攻撃を送信します。内部ソースを含む、ほぼ全てのデータソースはインジェクションの経路になる可能性があります。	<u>インジェクション欠陥</u> は、信頼していないデータをアプリケーションがインタプリタに送る際に起こります。インジェクションの欠陥は、しばしばSQLクエリ、LDAPクエリ、XPathクエリ、NoSQLクエリ、OSコマンド、プログラム引数などで発見されます。インジェクションの欠陥は、コードを検査すれば検出が容易です。スキャンツールとファジングツールは、攻撃者がインジェクション欠陥を見つけることを手助けします。		インジェクションによるデータ損失または破壊、アカウントビリティの欠如、アクセス拒否などの結果をもたらします。インジェクションにより、時には完全なホスト乗っ取りを引き起こす可能性があります。	インタプリタを稼働させているプラットフォームと影響を受けるデータの事業価値を考慮下さい。全てのデータは窃盗・変更・削除される可能性があります。それにより評判に傷が付くことになりませんか？

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ー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があります。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なります。

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、ストアー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。

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。

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

IPA Better Life with IT 情報処理推進機構

文字サイズ 標準 拡大 検索

・ IPAについて ・ お知らせ一覧 ・ サイトマップ ・ お問い合わせ ・ ENGLISH

HOME 情報セキュリティ 産業サイバーセキュリティセンター ソフトウェア高信頼化 未踏/セキュリティキャンプ IT人材の育成 情報処理技術者試験 情報処理安全確保支援士試験 国際標準の推進

HOME > 情報セキュリティ > 情報セキュリティ対策 > 脆弱性対策 > 安全なウェブサイトの作り方 本文を印刷する

情報セキュリティ

安全なウェブサイトの作り方

最終更新日 2016年1月27日
独立行政法人 情報処理推進機構
セキュリティセンター

「安全なウェブサイトの作り方」は、IPAが届出⁽¹⁾を受けた脆弱性関連情報を基に、届出件数の多かった脆弱性や攻撃による影響度が大きい脆弱性を取り上げ、ウェブサイト開発者や運営者が適切なセキュリティを考慮したウェブサイトを作成するための資料です。

「安全なウェブサイトの作り方」改訂第7版の内容

・第1章では、「ウェブアプリケーションのセキュリティ実装」として、SQLインジェクション、OSコマンド・インジェクション、クロスサイト・スクリプティング等11種類の脆弱性を取り上げ、それぞれの脆弱性で発生しうる脅威や特に注意が必要な

情報セキュリティ

- > 脆弱性対策情報
- > 届出・相談・情報提供
- > 特集コンテンツ
- > 情報セキュリティ啓発
- > 情報セキュリティ対策
 - > 日常における情報セキュリティ対策
 - > 長期休暇における情報セキュリティ対策

<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	※ <input type="checkbox"/> 対応済 <input type="checkbox"/> 未対策 <input type="checkbox"/> 対応不要	<input type="checkbox"/> 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
				<input type="checkbox"/> 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	<input type="checkbox"/> 対応済 <input type="checkbox"/> 未対策 <input type="checkbox"/> 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	<input type="checkbox"/> 対応済 <input type="checkbox"/> 未対策 <input type="checkbox"/> 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	<input type="checkbox"/> 対応済 <input type="checkbox"/> 未対策 <input type="checkbox"/> 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	<input type="checkbox"/> 対応済 <input type="checkbox"/> 未対策 <input type="checkbox"/> 対応不要	<input type="checkbox"/> 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	<input type="checkbox"/> 対応済 <input type="checkbox"/> 未対策 <input type="checkbox"/> 対応不要	<input type="checkbox"/> 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

ウェブ健康診断 仕様

「安全なウェブサイトの作り方」別冊

注意事項

本診断は検査パターンを絞り込んだものです。安全宣言には繋がりません。



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。

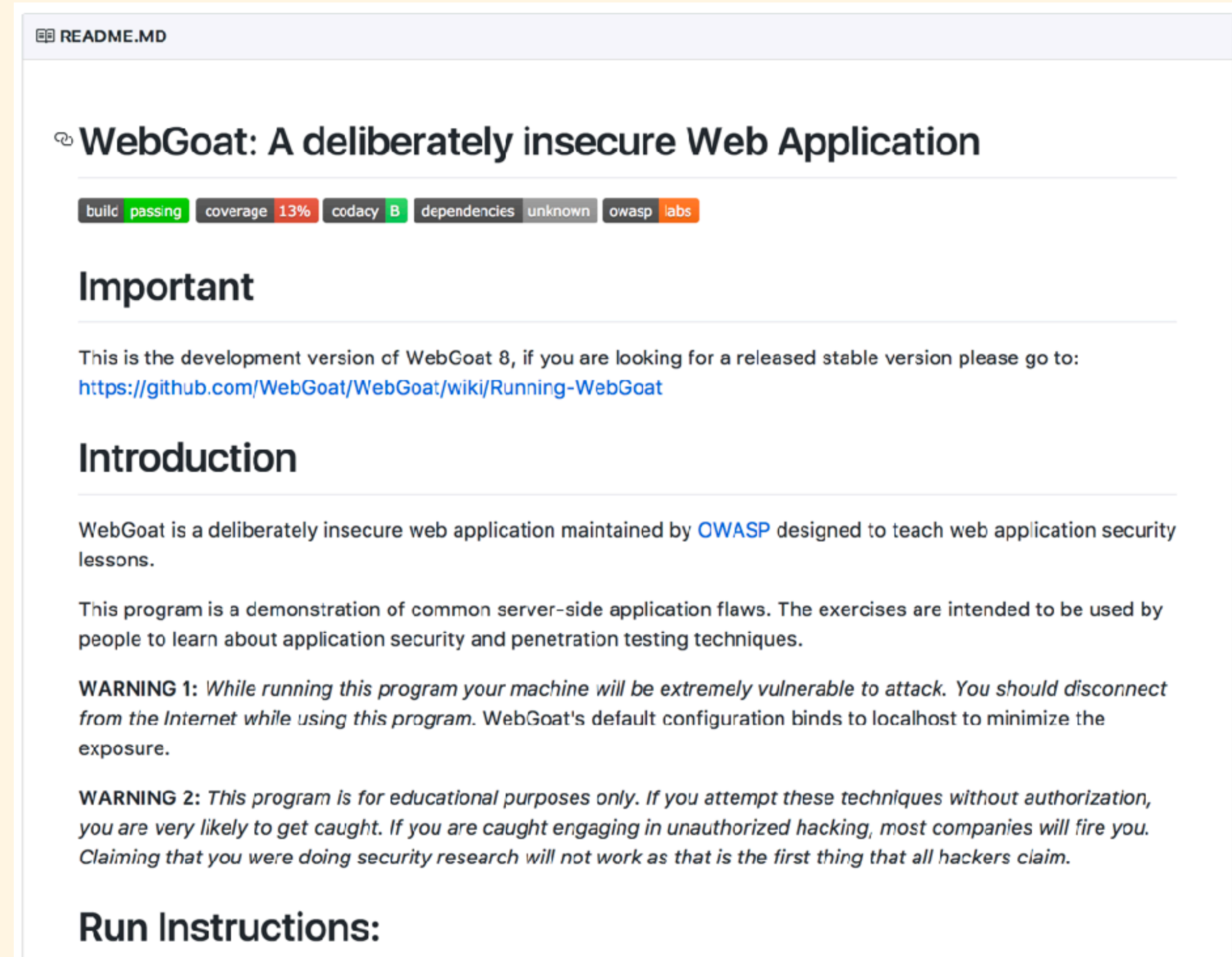
脆弱なサイトで脆弱性を見つけてみよう

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
```




Username

Password

Sign in

The following accounts are built into Webgoat

Account	User	Password
Webgoat User	guest	guest
Webgoat Admin	webgoat	webgoat

<http://localhost:8080/WebGoat/>

guest / guestでログインしたら



- Introduction >
- General >
- Access Control Flaws >
- AJAX Security >
- Authentication Flaws >
- Buffer Overflows >
- Code Quality >
- Concurrency >
- Cross-Site Scripting (XSS) >
- Improper Error Handling >
- Injection Flaws >
 - Command Injection
 - Numeric SQL Injection
 - Log Spoofing
 - XPATH Injection
 - String SQL Injection
 - LAB: SQL Injection
 - Stage 1: String SQL Injection
 - Stage 2: Parameterized Query #1
 - Stage 3: Numeric SQL Injection
 - Stage 4: Parameterized Query #2
 - Database Backdoors
 - Blind Numeric SQL Injection
 - Blind String SQL Injection
- Denial of Service >
- Insecure Communication >

Numeric SQL Injection

[Show Source](#)
[Show Solution](#)
[Show Plan](#)
[Show Hints](#)
[Restart Lesson](#)

SQL injection attacks represent a serious threat to any database-driven site. The methods behind an attack are easy to learn and the damage caused can range from considerable to complete system compromise. Despite these risks, an incredible number of systems on the internet are susceptible to this form of attack.

Not only is it a threat easily instigated, it is also a threat that, with a little common-sense and forethought, can easily be prevented.

It is always good practice to sanitize all input data, especially data that will used in OS command, scripts, and database queries, even if the threat of SQL injection has been prevented in some other manner.

General Goal(s):

The form below allows a user to view weather data. Try to inject an SQL string that results in all the weather data being displayed.

Select your local weather station:

```
SELECT * FROM weather_data WHERE station = [station]
```



Cookies / Parameters

Cookie/s

name	_rails Goat_session
value	TkdxWJZOG9UN0ISQUUyS3dMU3pHTzMTUUV30VE4TkR4WHlrT3BubTFRK01MQzNUJDI DeHIXQStXeldacVRQN3d6N3dSVE9KaGRxQz NwcFNncy9qbk10MzR1THhtQ2IYbVhqOUF2a1FOUVBQUVFYUWNUQnNTcC96b3ZKdi9KVT NvMHdwMjA5cWhFcE82ZVhzQ0dUZm81WSt6ZGhPcjBPedqZEdnZIV6TrnJjTKJMT2J3QWx yaWImNmdKSzlyVEhLeS9tUDdmTEhFeUNnaGtuY2szMjlqeGHTW4V0FzcVZJQ3IQR1ducWZ1VVFzemNlbDRjSVBEam900XJpeWhVZFQ1RE ZXVmtjZThEV3JrSnJ0enU2N2E0RjVGVG45bz FkcE9Hemx6aDltaIU9LS15V2xKSVJDenNib3Q2aFJzbnZrLzV3PT0%3D-ed2b0d1c895bd96ba9d54f7cb508cdf49ec0de38
comment	
domain	
maxAge	0
path	
secure	false
version	0
httpOnly	false

Parameters

scr	101829144
menu	1100
stage	
num	

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



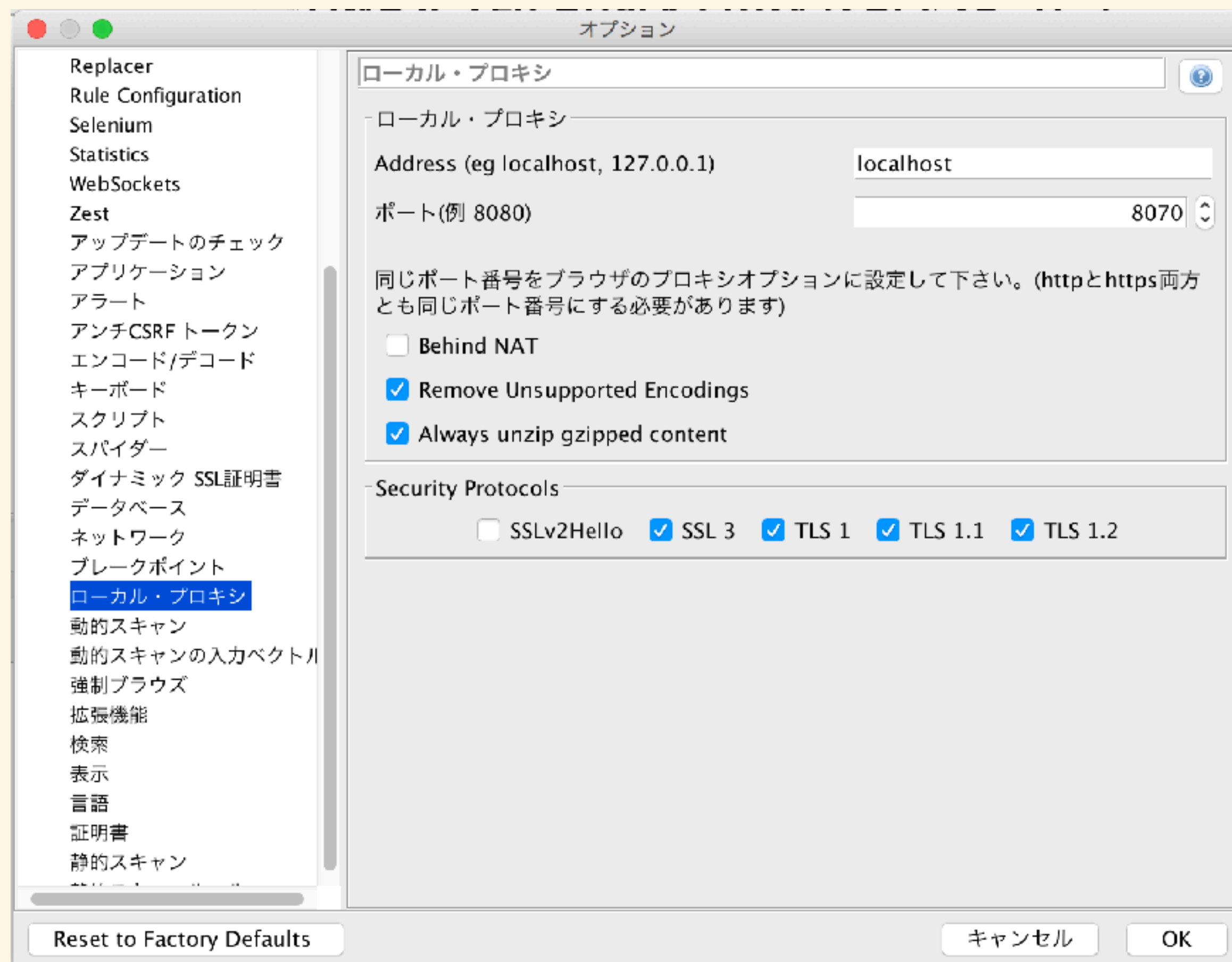
OWASP ZAP

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)

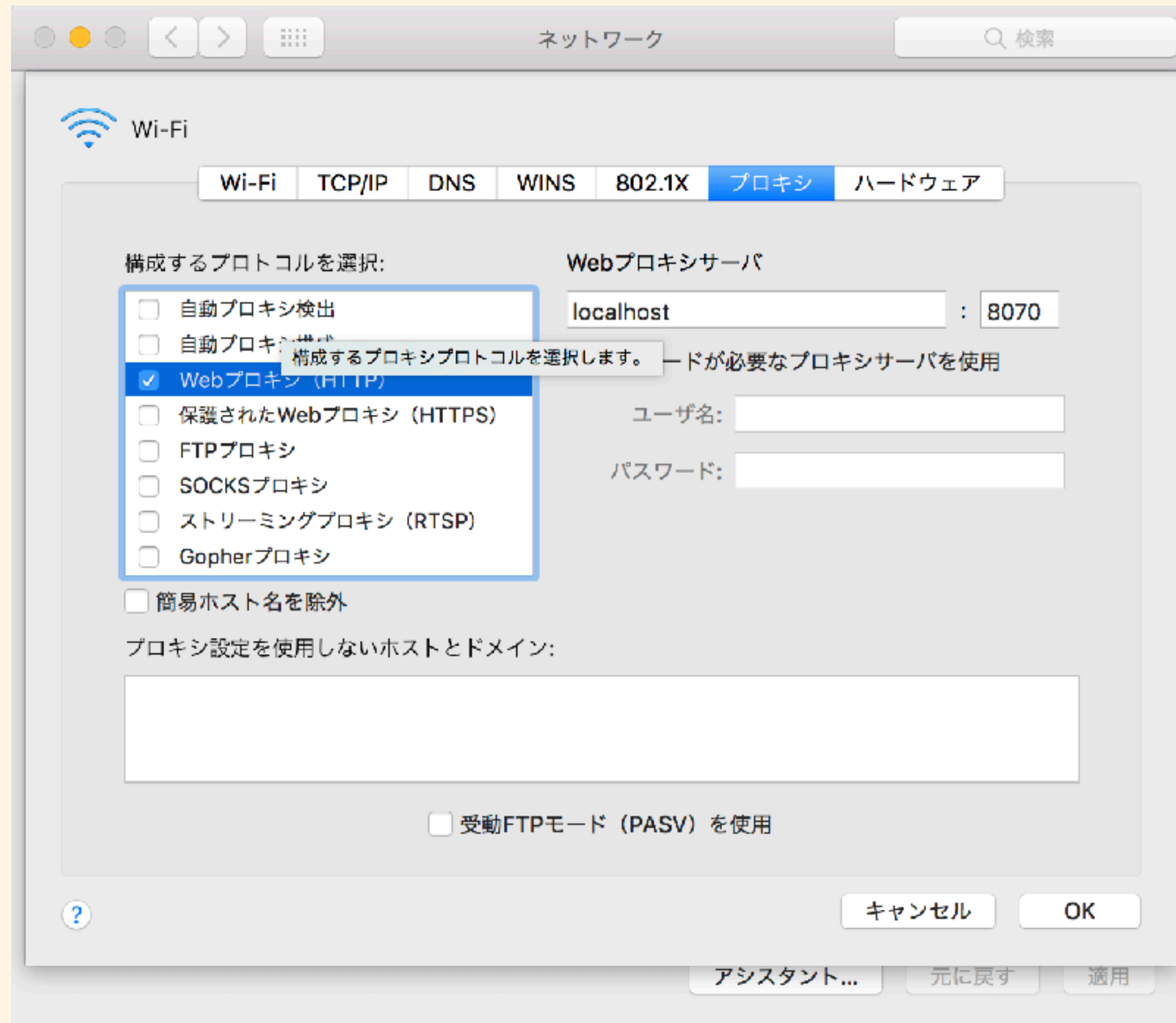


ZAPのオプションから、「ローカル・プロキシ」を設定する。

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)

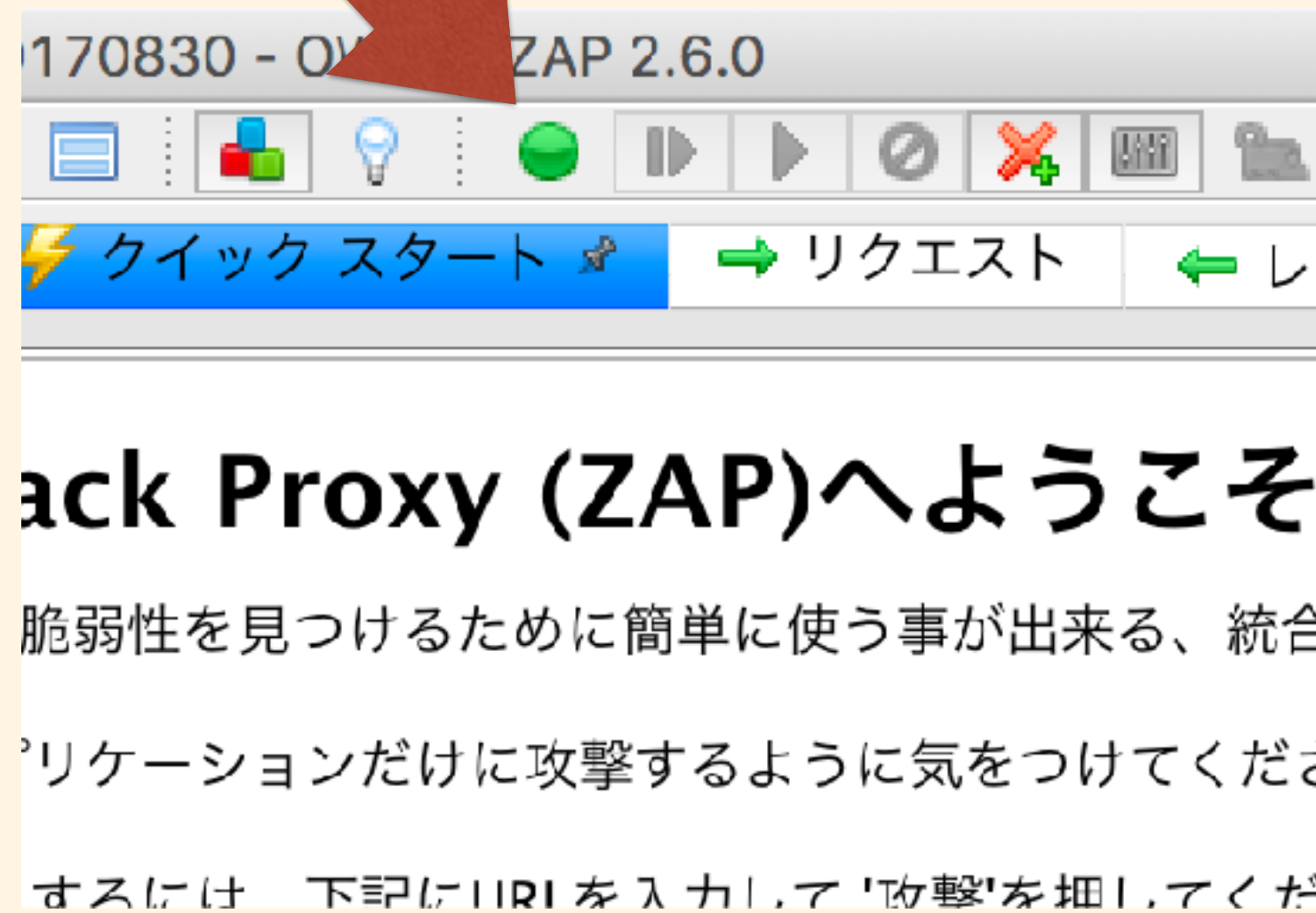


ネットワーク設定からWebプロキシを設定。先程のZAPで設定したポートを指定して下さい。

OWASP ZAPの使い方

ローカルプロキシ編 (3)

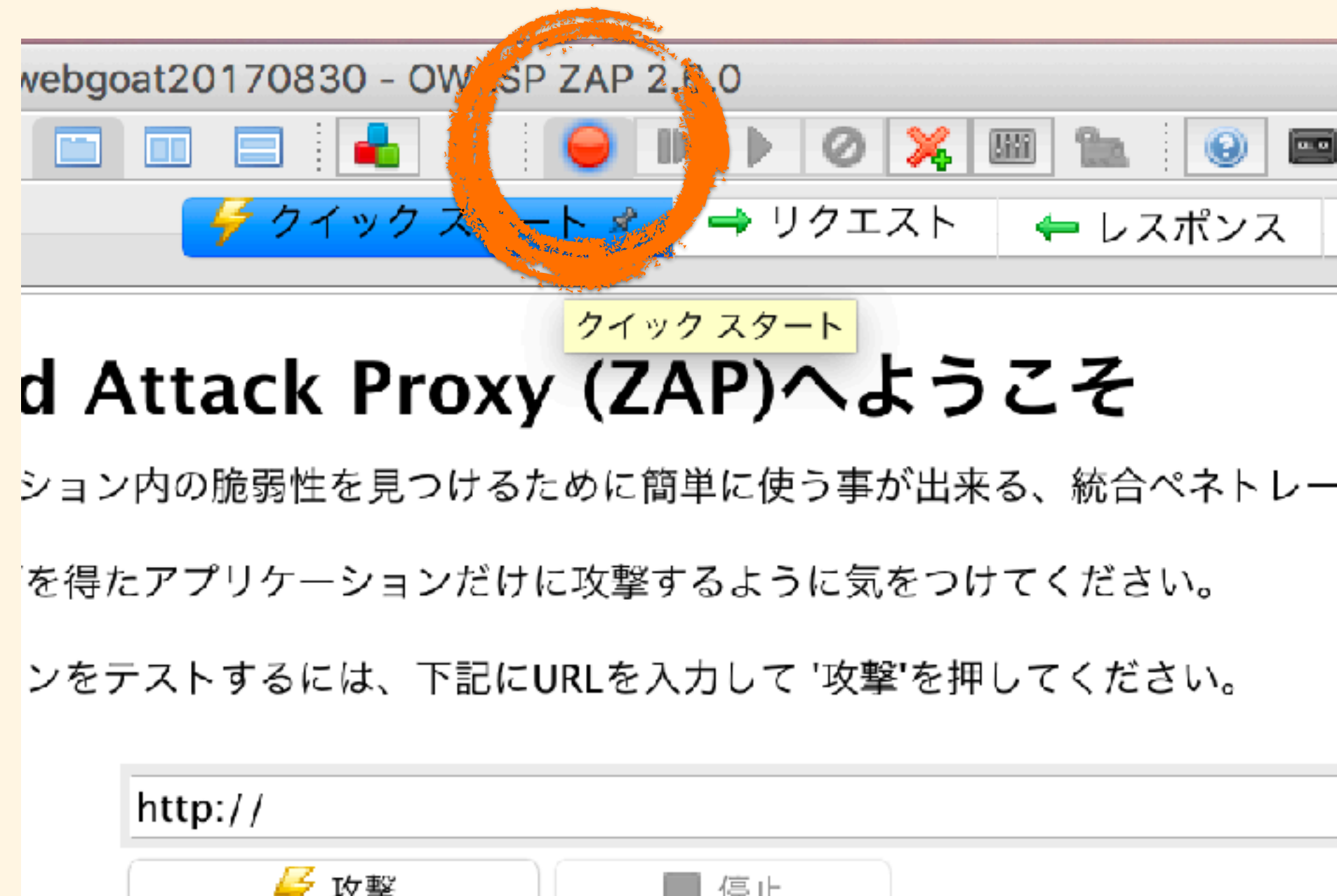
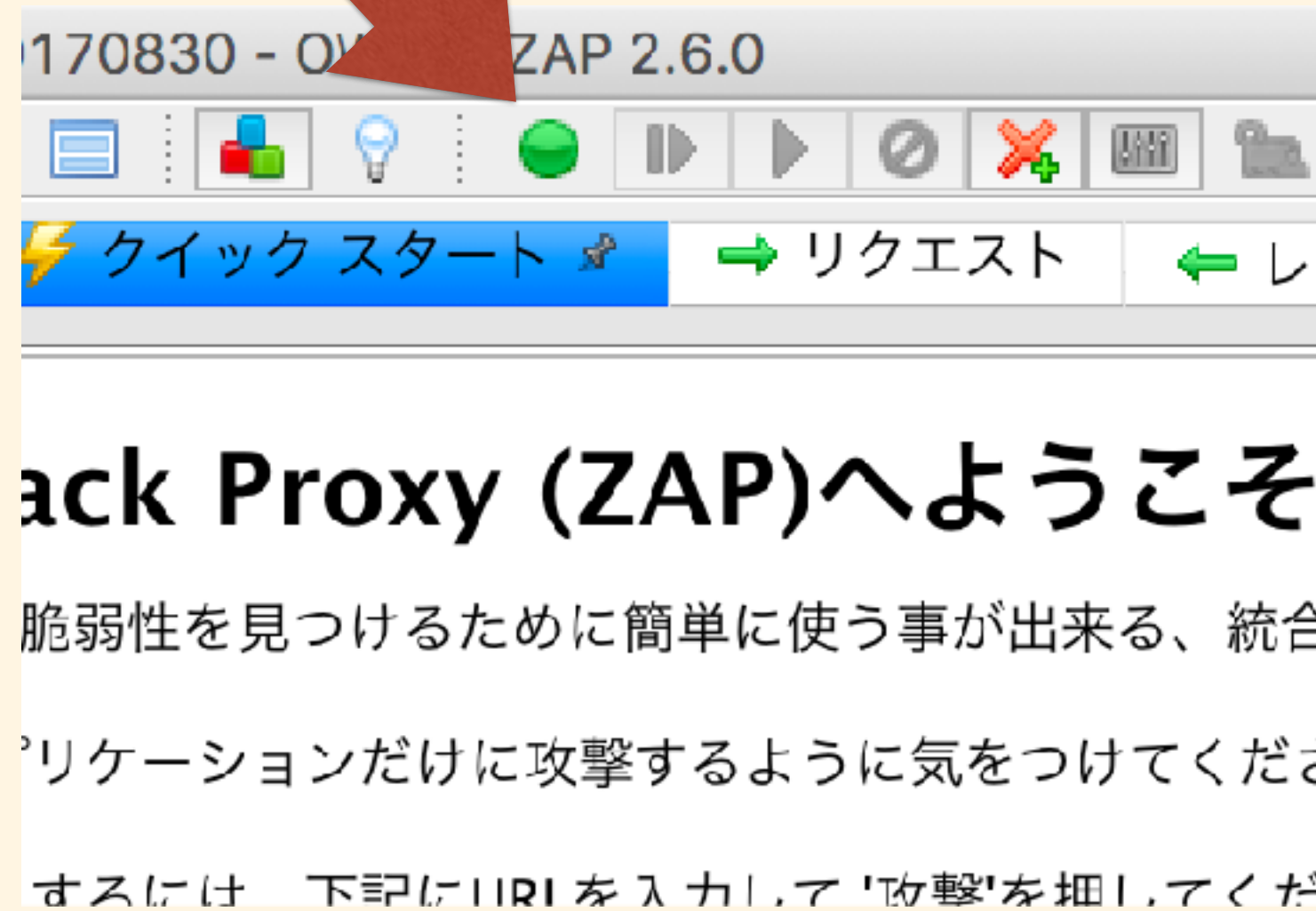
ブレーク設定をON



OWASP ZAPの使い方

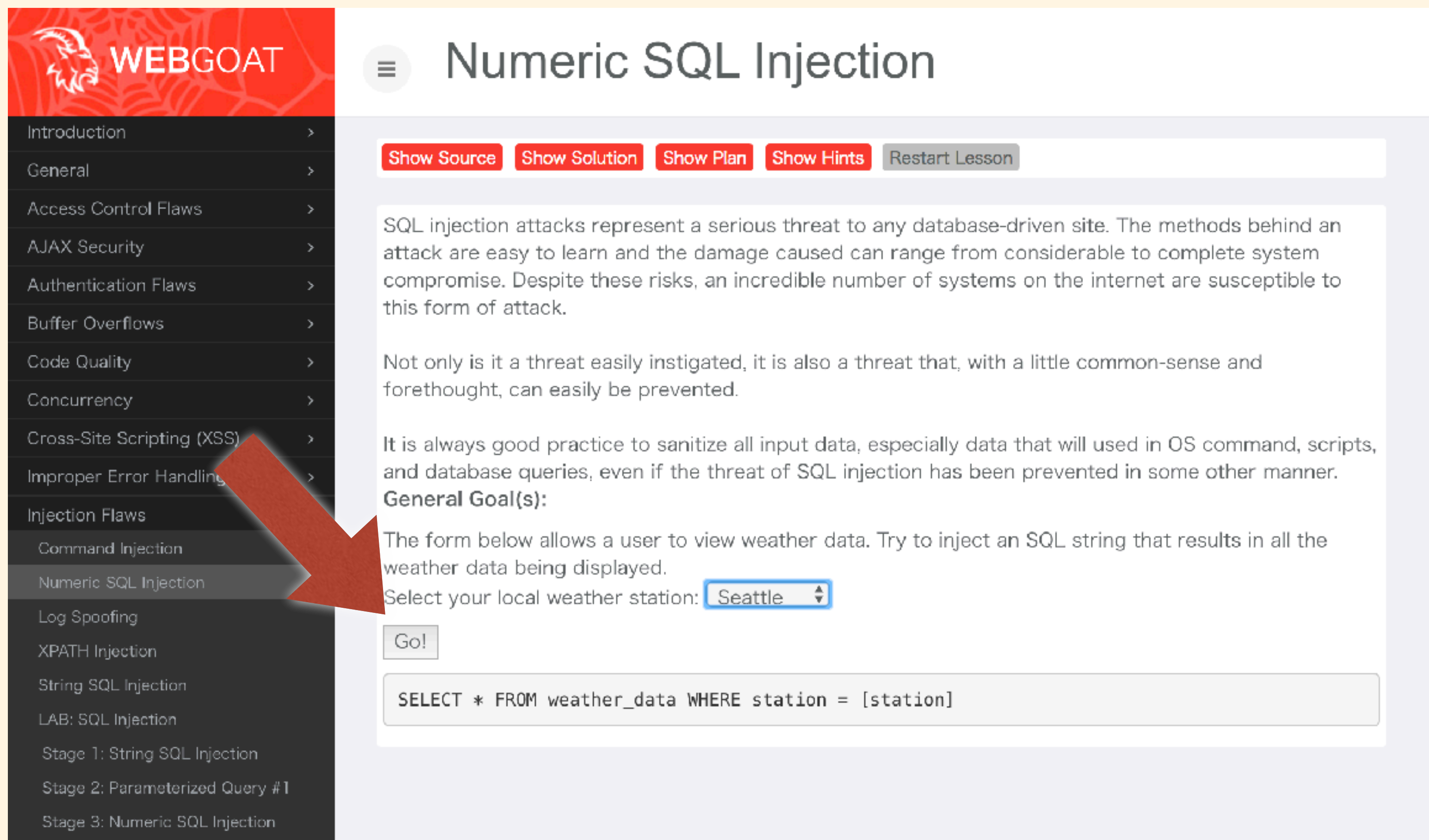
ローカルプロキシ編 (3)

ブレーク設定をON



OWASP ZAPの使い方

ローカルプロキシ編（4）



The screenshot shows the OWASP ZAP WebGoat interface. On the left is a navigation menu with the following items: Introduction, General, Access Control Flaws, AJAX Security, Authentication Flaws, Buffer Overflows, Code Quality, Concurrency, Cross-Site Scripting (XSS), Improper Error Handling, Injection Flaws, Command Injection, Numeric SQL Injection (highlighted), Log Spoofing, XPATH Injection, String SQL Injection, LAB: SQL Injection, Stage 1: String SQL Injection, Stage 2: Parameterized Query #1, and Stage 3: Numeric SQL Injection. A red arrow points from the 'Numeric SQL Injection' menu item to the main content area. The main content area is titled 'Numeric SQL Injection' and contains several buttons: 'Show Source', 'Show Solution', 'Show Plan', 'Show Hints', and 'Restart Lesson'. The text explains that SQL injection attacks are a serious threat and that it is good practice to sanitize all input data. It then presents a 'General Goal(s):' section with a form that says 'The form below allows a user to view weather data. Try to inject an SQL string that results in all the weather data being displayed.' The form includes a dropdown menu for 'Select your local weather station:' with 'Seattle' selected, a 'Go!' button, and a text input field containing the SQL query: `SELECT * FROM weather_data WHERE station = [station]`.

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)



無題セッション - webgoat20170830 - OWASP ZAP 2.6.0

メソッド: POST
Header: デフォルトビュー
Body: デフォルトビュー

```
POST http://localhost:8080/WebGoat/attack?Screen=101829144&menu=1100 HTTP/1.1
Host: localhost:8080
Proxy-Connection: keep-alive
Content-Length: 22
Accept: */*
Origin: http://localhost:8080
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko
station=102%200R%201%3D1&SUBMIT=Go!
```

OWASP ZAPの使い方

ローカルプロキシ編 (6)



無題セッション - webgoat20170830 - OWASP ZAP 2.6.0

メソッド: POST Header: デフォルトビュー Body: デフォルトビュー

```
POST http://localhost:8080/WebGoat/attack?Screen=101829144&menu=1100 HTTP/1.1
Host: localhost:8080
Proxy-Connection: keep-alive
Content-Length: 22
Accept: */*
Origin: http://localhost:8080
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko)
station=102%200R%201%3D1&SUBMIT=Go!
```

↑
パラメータを書き換えて

OWASP ZAPの使い方

ローカルプロキシ編 (6)

無題セッション - webgoat20170830 - OWASP ZAP 2.6.0

メソッド: POST Header: デフォルトビュー レスポンス: デフォルトビュー

POST http://localhost:8080/WebGoat/attack?Screen=101829144&menu=1100 HTTP/1.1
Host: localhost:8080
Proxy-Connection: keep-alive
Content-Length: 22
Accept: */*
Origin: http://localhost:8080
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko)

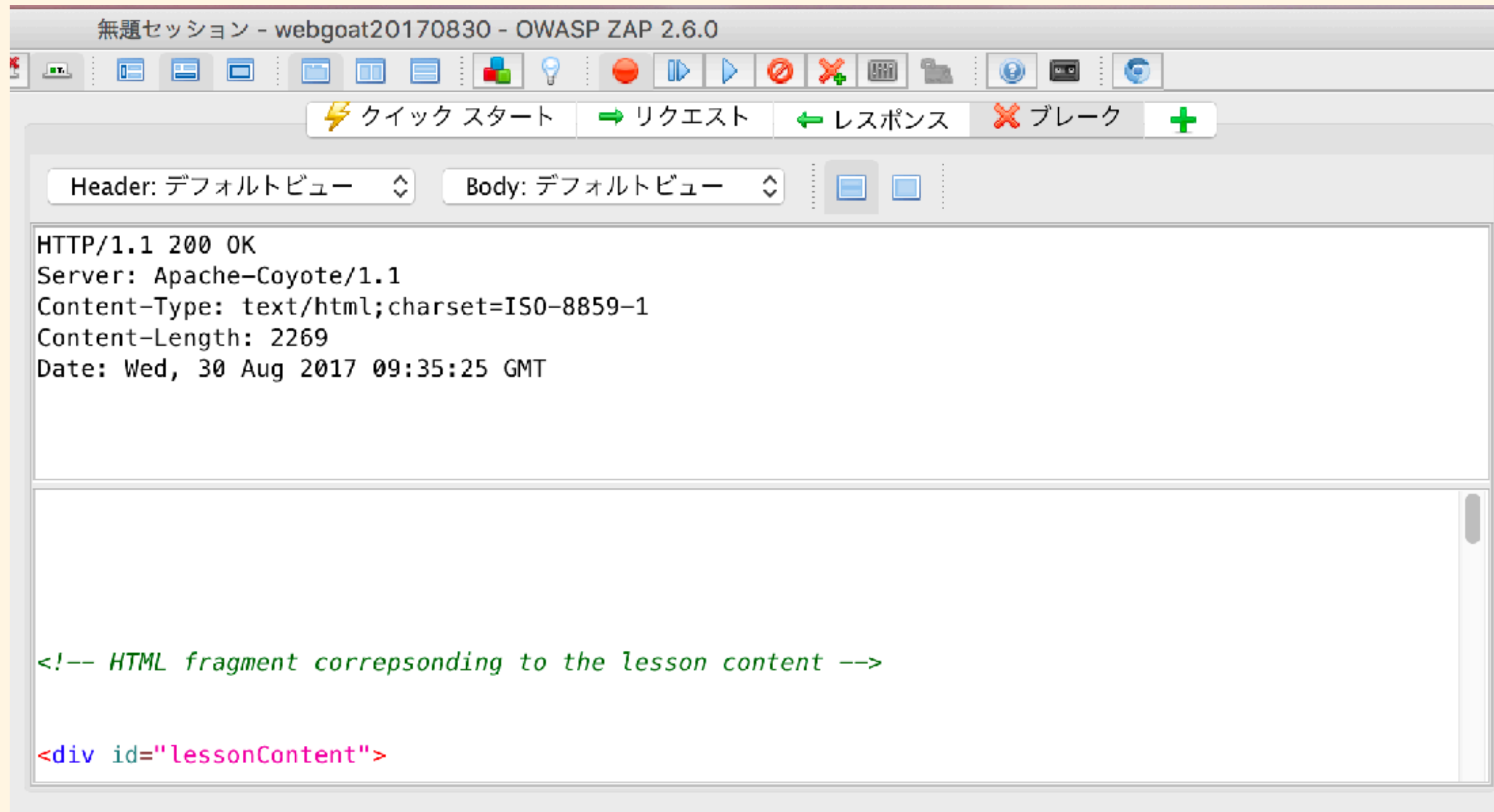
station=102%200R%201%3D1&SUBMIT=Go!

GO!

パラメータを書き換えて

OWASP ZAPの使い方

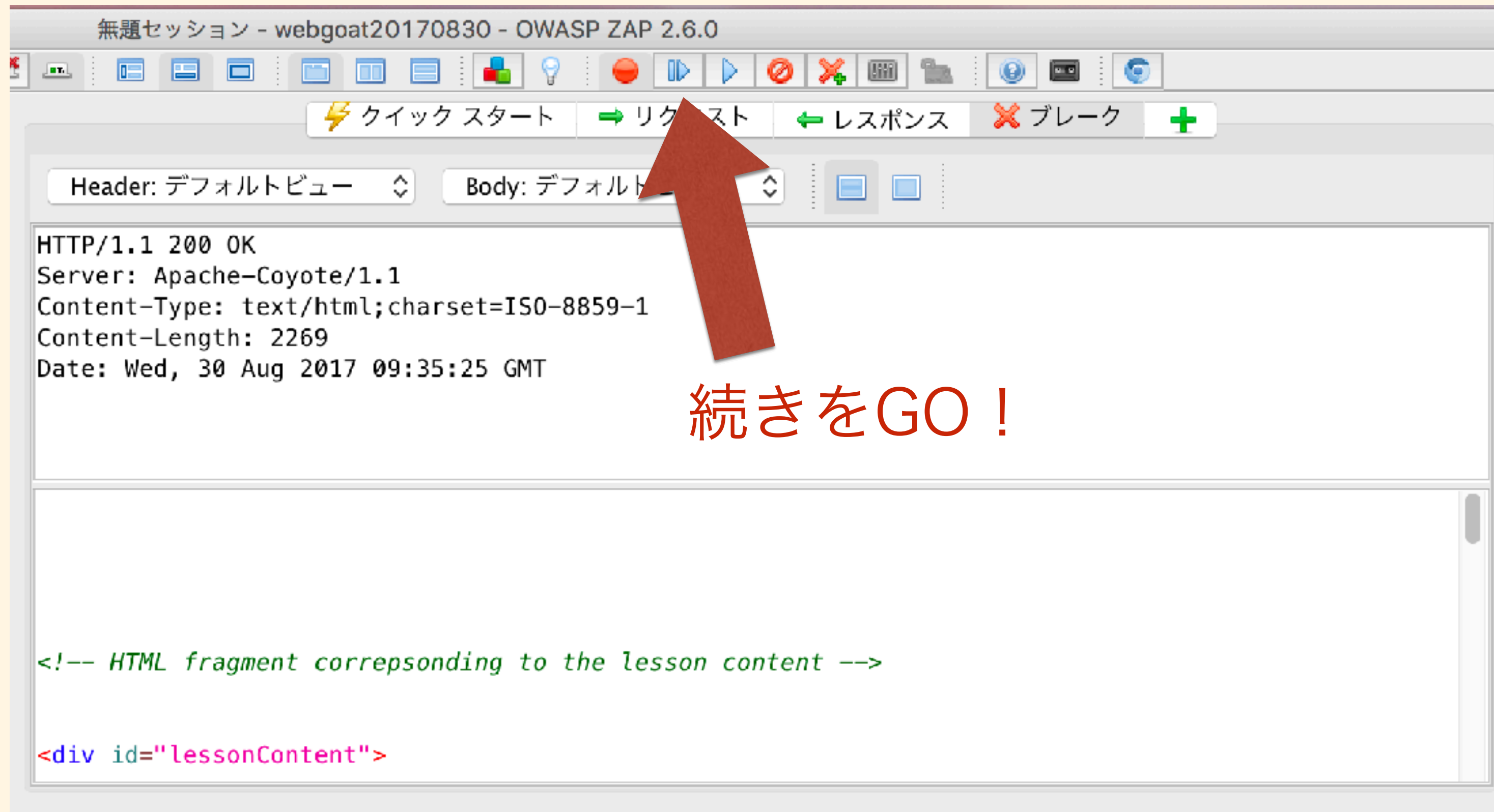
ローカルプロキシ編 (7)



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）

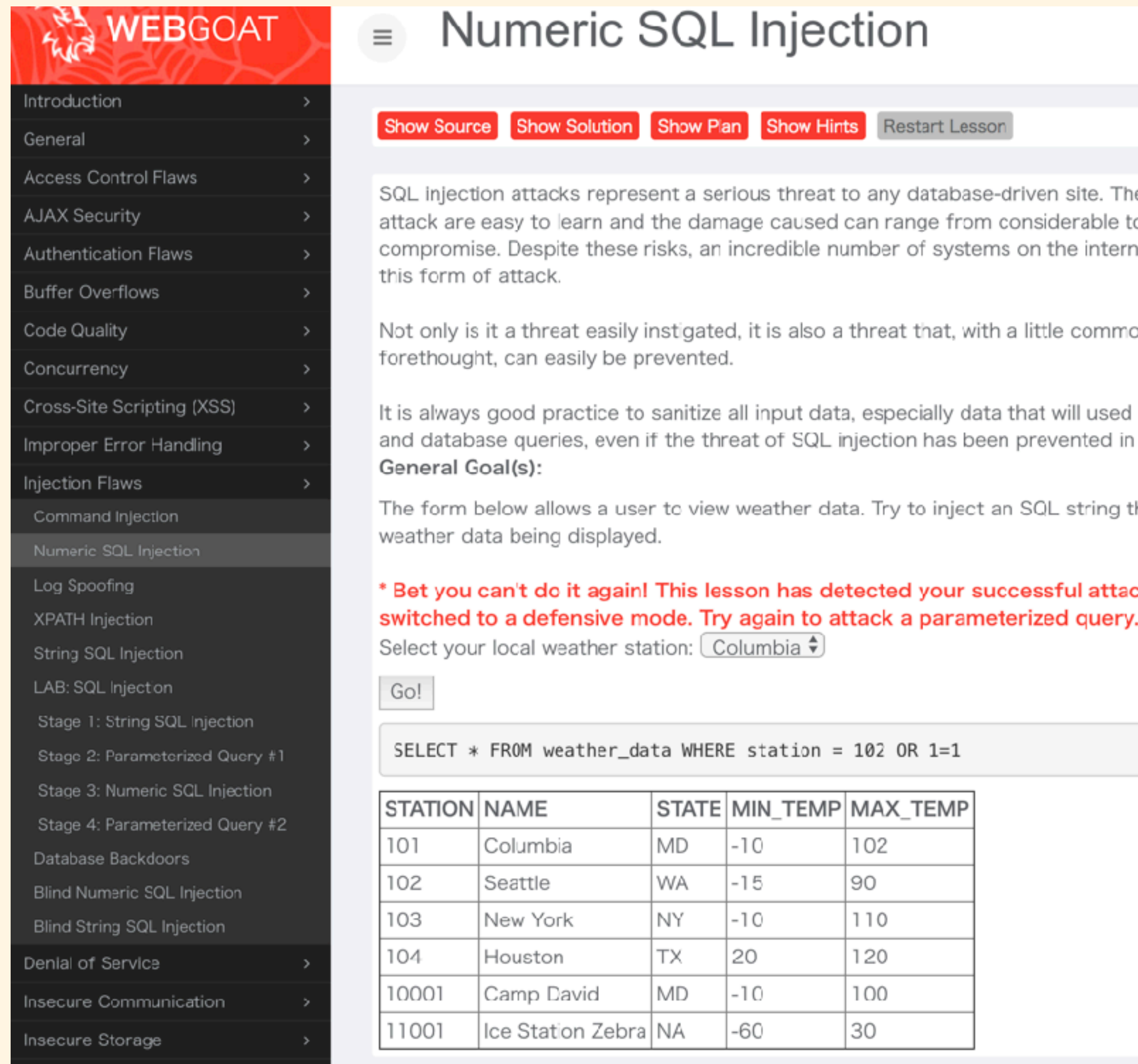


続きをGO！

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）



The screenshot shows the OWASP ZAP WebGoat interface. On the left is a navigation menu with various security topics. The main content area is titled "Numeric SQL Injection" and contains text explaining the attack, a "General Goal(s)" section, and a form for a weather data query. The form includes a dropdown menu for "Columbia" and a "Go!" button. Below the form, a table displays the results of the query, showing columns for STATION, NAME, STATE, MIN_TEMP, and MAX_TEMP.

WEBGOAT

Introduction >
General >
Access Control Flaws >
AJAX Security >
Authentication Flaws >
Buffer Overflows >
Code Quality >
Concurrency >
Cross-Site Scripting (XSS) >
Improper Error Handling >
Injection Flaws >
 Command Injection
 Numeric SQL Injection
 Log Spoofing
 XPath Injection
 String SQL Injection
 LAB: SQL Injection
 Stage 1: String SQL Injection
 Stage 2: Parameterized Query #1
 Stage 3: Numeric SQL Injection
 Stage 4: Parameterized Query #2
 Database Backdoors
 Blind Numeric SQL Injection
 Blind String SQL Injection
Denial of Service >
Insecure Communication >
Insecure Storage >

Numeric SQL Injection

Show Source Show Solution Show Plan Show Hints Restart Lesson

SQL injection attacks represent a serious threat to any database-driven site. The attack are easy to learn and the damage caused can range from considerable to compromise. Despite these risks, an incredible number of systems on the internet use this form of attack.

Not only is it a threat easily instigated, it is also a threat that, with a little common forethought, can easily be prevented.

It is always good practice to sanitize all input data, especially data that will be used in and database queries, even if the threat of SQL injection has been prevented in some other way.

General Goal(s):

The form below allows a user to view weather data. Try to inject an SQL string that causes the weather data being displayed.

*** Bet you can't do it again! This lesson has detected your successful attack and switched to a defensive mode. Try again to attack a parameterized query.**

Select your local weather station:

Go!

```
SELECT * FROM weather_data WHERE station = 102 OR 1=1
```


STATION	NAME	STATE	MIN_TEMP	MAX_TEMP
101	Columbia	MD	-10	102
102	Seattle	WA	-15	90
103	New York	NY	-10	110
104	Houston	TX	20	120
10001	Camp David	MD	-10	100
11001	Ice Station Zebra	NA	-60	30

ブラウザにレスポンスが表示される！

**お手軽にいる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)



無題セッション - webgoat20170830 - OWASP ZAP 2.6.0

⚡ クイック スタート → リクエスト ← レスポンス ✕ ブレーク +

OWASP Zed Attack Proxy (ZAP)へようこそ

ZAPはWEBアプリケーション内の脆弱性を見つけるために簡単に使う事が出来る、統合ペネトレーションテストツールです。

特別にテストする許可を得たアプリケーションだけに攻撃するように気をつけてください。

すぐにアプリケーションをテストするには、下記にURLを入力して '攻撃'を押してください。

攻撃対象 URL:

進行状況: 停止

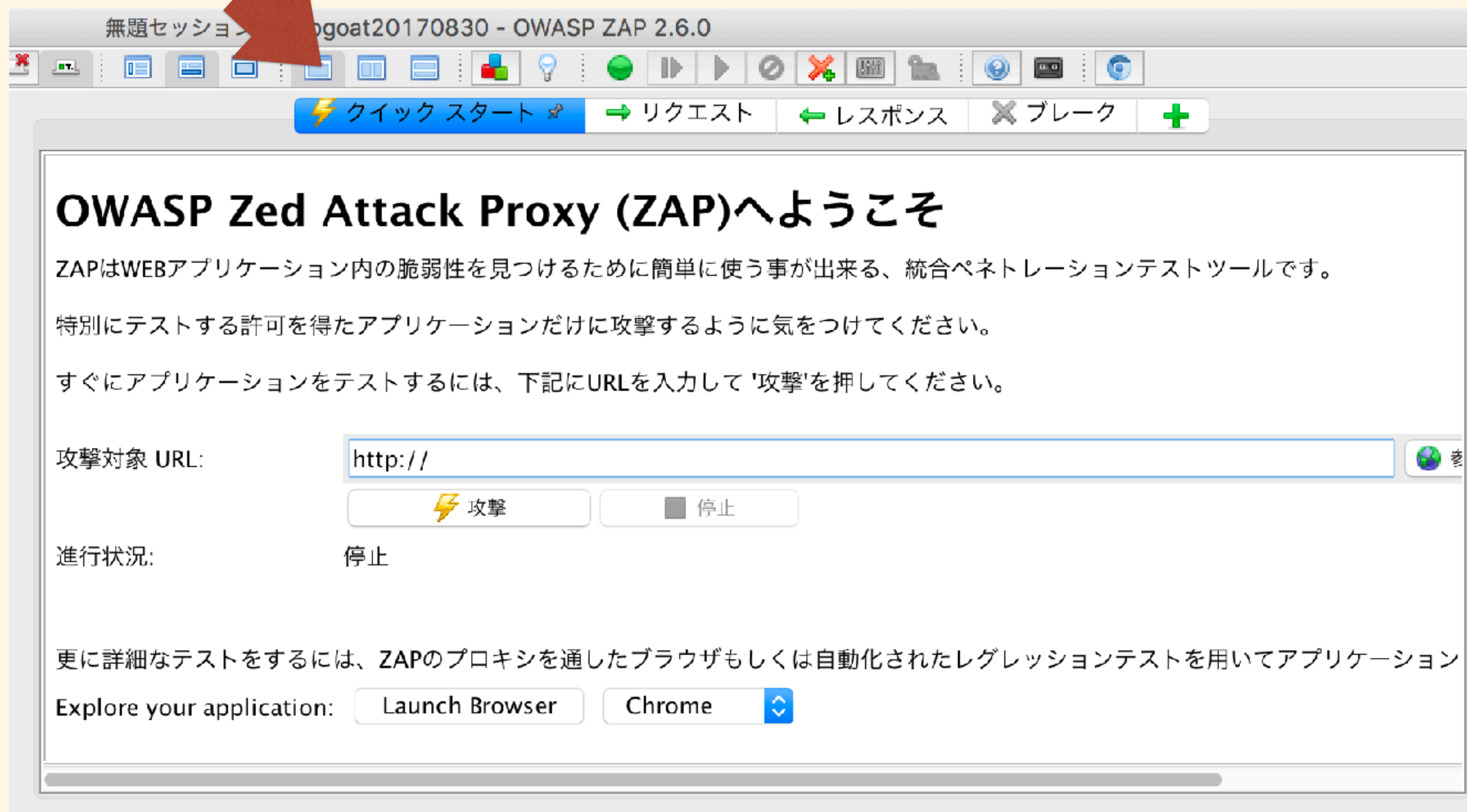
更に詳細なテストをするには、ZAPのプロキシを通したブラウザもしくは自動化されたレグレッションテストを用いてアプリケーション

Explore your application: ▾

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



無題セッション goat20170830 - OWASP ZAP 2.6.0

クイック スタート リクエスト レスポンス ブレーク +

OWASP Zed Attack Proxy (ZAP)へようこそ

ZAPはWEBアプリケーション内の脆弱性を見つけるために簡単に使う事が出来る、統合ペネトレーションテストツールです。

特別にテストする許可を得たアプリケーションだけに攻撃するように気をつけてください。

すぐにアプリケーションをテストするには、下記にURLを入力して '攻撃'を押してください。

攻撃対象 URL:

進行状況: 停止

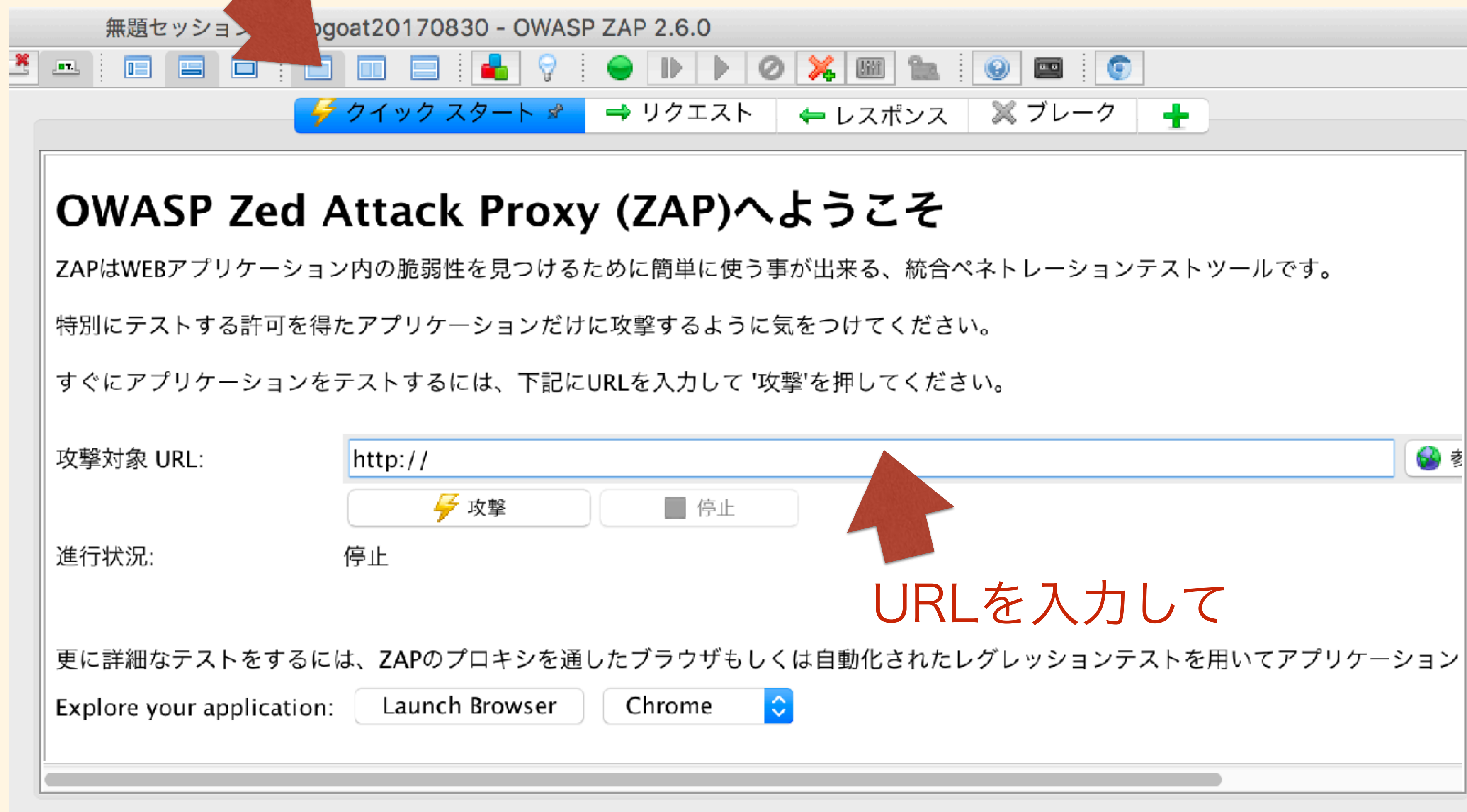
更に詳細なテストをするには、ZAPのプロキシを通したブラウザもしくは自動化されたレグレッションテストを用いてアプリケーション

Explore your application:

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



無題セッション goat20170830 - OWASP ZAP 2.6.0

クイック スタート リクエスト レスポンス ブレーク

OWASP Zed Attack Proxy (ZAP)へようこそ

ZAPはWEBアプリケーション内の脆弱性を見つけるために簡単に使う事が出来る、統合ペネトレーションテストツールです。

特別にテストする許可を得たアプリケーションだけに攻撃するように気をつけてください。

すぐにアプリケーションをテストするには、下記にURLを入力して '攻撃'を押してください。

攻撃対象 URL:

攻撃 停止

進行状況: 停止

更に詳細なテストをするには、ZAPのプロキシを通したブラウザもしくは自動化されたレグレッションテストを用いてアプリケーション

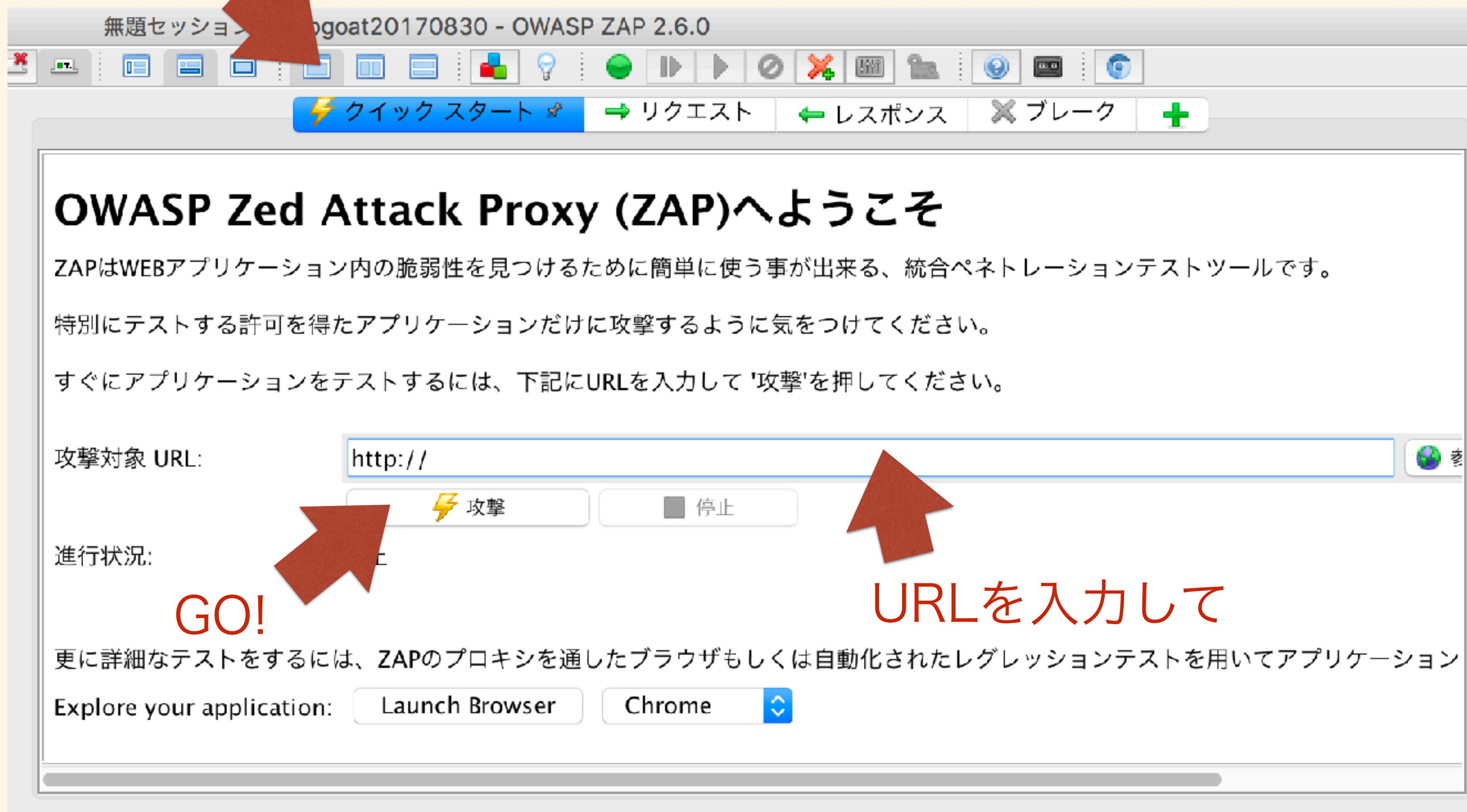
Explore your application:

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



The screenshot shows the OWASP Zed Attack Proxy (ZAP) 2.6.0 interface. At the top, there is a toolbar with several icons. Below the toolbar, there is a navigation bar with buttons for 'クイック スタート' (Quick Start), 'リクエスト' (Request), 'レスポンス' (Response), 'ブレイク' (Break), and '+'. The main content area has a heading 'OWASP Zed Attack Proxy (ZAP)へようこそ' (Welcome to OWASP Zed Attack Proxy (ZAP)). Below the heading, there is a paragraph of text: 'ZAPはWEBアプリケーション内の脆弱性を見つけるために簡単に使う事が出来る、統合ペネトレーションテストツールです。' (ZAP is a unified penetration testing tool that can be used easily to find vulnerabilities in web applications.) and another paragraph: '特別にテストする許可を得たアプリケーションだけに攻撃するように気をつけてください。' (Please be careful to attack only applications for which you have special permission to test.) and 'すぐにアプリケーションをテストするには、下記にURLを入力して '攻撃' を押してください。' (To test the application immediately, enter the URL in the field below and click 'Attack'). Below this text, there is a text input field labeled '攻撃対象 URL:' (Target URL) containing 'http://'. To the right of the input field is a globe icon. Below the input field, there are two buttons: '攻撃' (Attack) with a lightning bolt icon and '停止' (Stop) with a square icon. Below these buttons, there is a '進行状況:' (Progress) label. At the bottom, there is a section labeled 'Explore your application:' with a 'Launch Browser' button and a dropdown menu currently showing 'Chrome'. There are three red arrows pointing to the 'クイック スタート' button, the '攻撃' button, and the URL input field. The text 'GO!' is written in red next to the '攻撃' button, and 'URLを入力して' (Enter URL) is written in red next to the URL input field.

無題セッション goat20170830 - OWASP ZAP 2.6.0

クイック スタート リクエスト レスポンス ブレイク +

OWASP Zed Attack Proxy (ZAP)へようこそ

ZAPはWEBアプリケーション内の脆弱性を見つけるために簡単に使う事が出来る、統合ペネトレーションテストツールです。

特別にテストする許可を得たアプリケーションだけに攻撃するように気をつけてください。

すぐにアプリケーションをテストするには、下記にURLを入力して '攻撃' を押してください。

攻撃対象 URL:

進行状況:

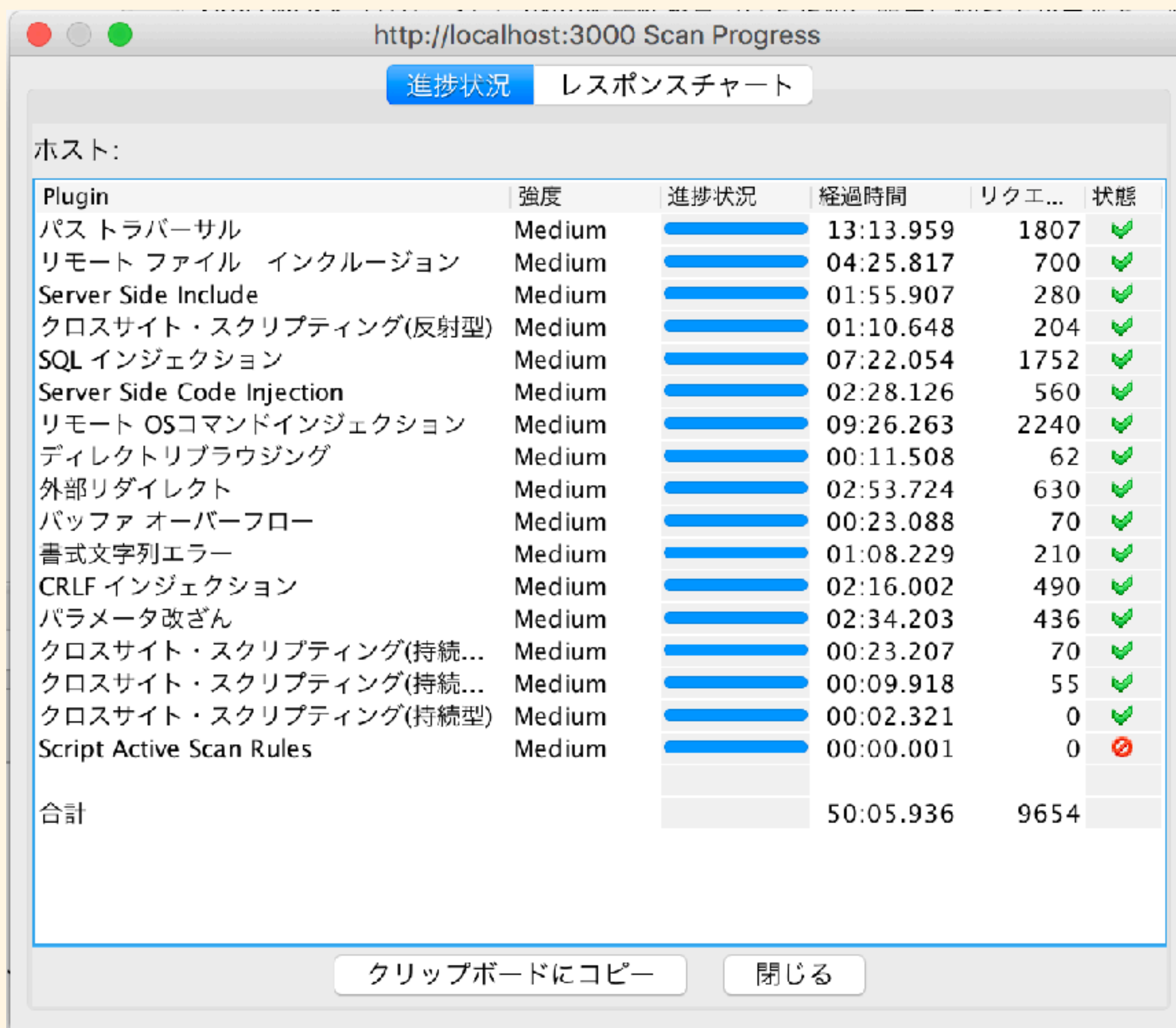
GO!

URLを入力して

Explore your application:

OWASP ZAPの使い方

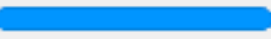
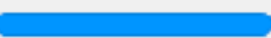
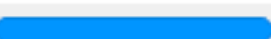
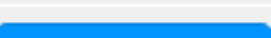
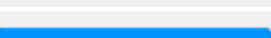
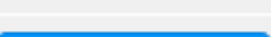
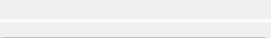
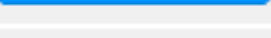
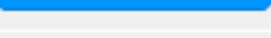
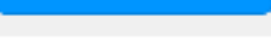
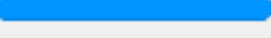
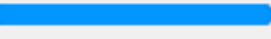
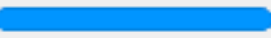
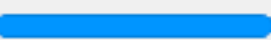

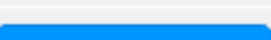
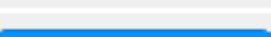
セキュリティスキャン編 (2)



http://localhost:3000 Scan Progress

進捗状況 レスポンスチャート

ホスト:

Plugin	強度	進捗状況	経過時間	リクエ...	状態
パス トラバーサル	Medium		13:13.959	1807	✓
リモート ファイル インクルージョン	Medium		04:25.817	700	✓
Server Side Include	Medium		01:55.907	280	✓
クロスサイト・スクリプティング(反射型)	Medium		01:10.648	204	✓
SQL インジェクション	Medium		07:22.054	1752	✓
Server Side Code Injection	Medium		02:28.126	560	✓
リモート OSコマンドインジェクション	Medium		09:26.263	2240	✓
ディレクトリブラウジング	Medium		00:11.508	62	✓
外部リダイレクト	Medium		02:53.724	630	✓
バッファ オーバーフロー	Medium		00:23.088	70	✓
書式文字列エラー	Medium		01:08.229	210	✓
CRLF インジェクション	Medium		02:16.002	490	✓
パラメータ改ざん	Medium		02:34.203	436	✓
クロスサイト・スクリプティング(持続...	Medium		00:23.207	70	✓
クロスサイト・スクリプティング(持続...	Medium		00:09.918	55	✓
クロスサイト・スクリプティング(持続型)	Medium		00:02.321	0	✓
Script Active Scan Rules	Medium		00:00.001	0	✗
合計			50:05.936	9654	

クリップボードにコピー 閉じる

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。