Exalted Community Prototype

A study in web application security

Marie Hogebrandt

Sammanfattning

Detta är en rapport om ett system i PHP byggt på ramverket Laravel, med MySQL som databas. Det är en prototyp för ett större system och har ett grundläggande användarsystem samt funktioner för att kunna posta och se meddelanden. Det har byggts för att minimera de vanligaste säkerhetsriskerna.

Nyckelord: *Laravel, Webbapplikationssäkerhet, PHP, MySQL*

Abstract

This is a report about a system in PHP built on the framework Laravel, with MySQL as a database. It is a prototype for a large system and has a basic user system as well as functionality to post and view messages. It has been built to minimise the most common security risks.

Keywords: *Laravel, Web Application Security, PHP, MySQL*

Contents

1 Introduction

The

1.1 Bakgrund och problemmotivering

I detta underkapitel ska du snabbt försöka skapa intresse hos läsaren för det problemområde du har valt att undersöka. Visa att du inte bara är insatt i ditt smala tekniska problem, utan att du har förståelse för det sammanhang där ditt problem dyker upp, att du kan betrakta det ur ett icketekniskt perspektiv och att du känner till den praktiska nyttan av den teknik du undersöker eller av den kunskap din studie förväntas ge upphov till.

Det är vanligt att den första meningen innehåller en visionär formulering eller historisk återblick. Tänk emellertid på att du inte kan veta säkert hur framtiden kommer att te sig, utan bör uttrycka dina visioner på ett nyanserat och sakligt sätt för att framstå som trovärdig.

Exempel: ”Mänskligheten har under historien gång ... Användandet av Internet och mobiltelefoner har vuxit explosionsartat sedan ... Nästa steg i utvecklingen förväntas bli ... Detta kan leda till problem med ... Inom denna studie undersöks om problemet kan lösas med hjälp av ... Denna teknologi kan bli särskilt värdefull om några år med tanke på att allt fler människor ..., och på att det finns en växande efterfrågan på marknaden efter ....”

En teknisk rapport som skrivs på uppdrag av ett företag kan till exempel inledas: ”Inom organisationen finns ett ökande behov av ... och samtidigt växande problem med .... Vi har därför fått i uppdrag att genomföra en förstudie om ... En lösning på detta problem är angelägen därför att den kan leda till avsevärd minskning av kostnader för ..., ökade marknadsandelar inom ... samt en förbättrad arbetsmiljö.”

1.2 Övergripande syfte

Projektets övergripande syfte är en visionär beskrivning av den riktning i vilken du vill arbeta, av vad du hoppas att projektet ska resultera i det långa loppet, samt av projektets motiv. Nyckeln till framgångsrik forskning är ofta att man lyckas formulera en intressant frågeställning, syftet blir då att besvara den. Syftesformuleringen kan vara på hög nivå, det vill säga den behöver inte vara klart avgränsad eller konkret. Det kan vara ett mål som du kanske aldrig kommer att uppnå, eller inte säkert kan veta när du har uppnått. Det kan även vara en problemformulering på hög nivå som inte kan besvaras av undersökningens diagram, tabeller eller andra objektiva resultat, men som senare kan diskuteras i det avslutande kapitlet.

Exempel: ”Projektets övergripande syfte är att ge upphov till förklaringar till varför . . . ”, ”Projektets syfte är att jämföra teknik A med teknik B som lösning på behov C”, ”Projektets syfte är att identifiera generella principer för sambandet mellan X och Y”, ”Projektet syftar till att ge upphov till nya tekniska lösningsförslag inom följande problemområde: . . . ”, ”Syftet är att ge upphov till ny kunskap inom organisationen om . . . ”, ”Projektet syftar till att utgöra ett beslutsunderlag för . . . ”.

1.3 Avgränsningar

Exempel: ”Studien har fokus på . . . Undersökningen är avgränsad till utvärdering av fall F1 och F2 . . . , undersökningens slutsatser bör emellertid vara generellt giltiga för alla . . . I undersökningen negligeras inverkan av Z, därför att . . . ”.

1.4 Frågeställning

Frågeställningen, eller målformuleringen, är en konkretisering av ovanstående syftesformulering. De frågor som specificeras ska besvaras av rapportens resultat, och i dess avslutande slutsatser. Målformuleringen ska vara så konkret att det i efterhand ska gå att avgöra om den har uppfyllts, och syftar till att utgöra stoppkriterium för när arbetet är slutfört. Specificera de objektiva numeriska resultat du söker. Du kan ange vad x- och y-axlarna eller kolumnerna ska visa i de diagram och tabeller du har för avsikt att ta fram.

Detta underkapitel skrivs vanligen efter det att du har genomfört teoristudien i chapter 2, och revideras ofta under projektets gång. Det förekommer att den konkreta problemformuleringen placeras efter teoristudien, eftersom det annars kan vara svårt för läsaren att förstå de begrepp du använder. Nackdelen med en sådan disposition är emellertid att läsaren kan tappa intresset för ämnet, till följd av att det dröjer så länge innan du som författare kommer till kärnpunkten.

Exempel på problemformulering för en vetenskaplig rapport: ”Undersökningen har som mål att besvara följande frågor:

1. Vilken betydelse har teknik A i jämförelse med teknik B för prestandamåttet Y vid olika värden på parameter X, för fall F1 och F2?

2. Vilken vinst ger . . . För matematiska definitioner av X och Y, se modellen i kapitel X.”

I kapitel X specificeras sedan objektivt de numeriska resultaten, till exempel vad man kommer att kunna se på x- och y-axlarna i det diagram där man tar diskussionen vidare.

Exempel på målformulering för en teknisk rapport: ”Undersökningens mål är att föreslå en lösning på följande tekniska problem: . . . Undersökningen har vidare som mål att verifiera att lösningsförslaget tillhandahåller användbara kriterier, samt utvärderar förslaget med avseende på prestandamått Y.”

1.5 Disposition

Beskriv kort rapportens disposition. Exempel: ”Kapitel X beskriver . . . ”.

1.6 Författarens bidrag

Beskriv vilken del av arbetet som du själv har gjort, och vad du har fått hjälp med till exempel av kollegor. Ange om du har redovisat någon del av arbetet under tidigare kurser eller examensarbeten. Utförs arbetet i grupp kan rapporten redovisa hur ansvaret för arbetets olika delar har fördelats mellan författarna. Givetvis ska alla medförfattare omnämnas för det arbete de lagt ned.

2 Development of Modern Web Applications

When developing in a modern context, in particular for the Internet, one needs to understand how to develop securely, efficiently and without repeating oneself. Though not as much is needed to understand the pertinent areas that this course covers, they are still important to understand the broader strokes.

2.1 Separation of Concerns

Per [7] Multitier architecture is a "client–server architecture in which presentation, application processing, and data management functions are logically separated". In Web Development, that is most commonly generalized to three tiers:

Frontend

What is actually served to the browser, whether it's dynamically generated or static. It includes JavaScript, CSS, HTML, typography, graphics, etc. The user interface, or presentation layer. In this system, the least concerns have been put into the frontend, excepting details needed to show understanding of JavaScript-based security issues.

Middle

What is most commonly referred to as "web development", be it in Ruby on Rails, PHP, .NET or any other kind of server-side languages. The bulk of this system is written in PHP, using the framework Laravel.

Backend

Where the middle processing technologies generate the frontend consumed by the browser, in most dynamic systems it gets its' information – data – from the backend database or data store, whether through a MySQL database, a NoSQL MongoDB or any other kind of system, the backend provides access to the data, and should be properly secured.

2.2 Technologies

Termer och förkortningar som är viktiga för läsarens förståelse av den fortsatta framställningen förklaras i detta kapitel. Första gången du i den löpande texten använder ett

begrepp eller en förkortning ska du förklara det, även om det dessutom finns definierat i ett terminologiavsnitt. När begrepp introduceras skrivs de med emfas.

Första gången en *förkortning* (förk.) används skrivs den inom parentes efter dess förklaring, såsom exemplifieras i denna mening.

Använd svenska termer så långt det är möjligt. Se Svenska datatermgruppens rekommendationer på URL

`http://www.datatermgruppen.se/`.

2.3 Att referera eller citera

Du refererar när du sammanfattar eller återger en text med egna ord. Exempelvis: Forsslund förespråkar mer berättande rubriker i tekniska rapporter och menar att man särskilt i underrubrikerna kan ge viktig information

Du citerar när du ordagrant återger en fras, en mening eller ett stycke. I normalfallet refererar man istället för att citera källor. Du kan använda direkta citat om du har speciella skäl, till exempel om du vill återge vedertagna definitioner av begrepp, när du tycker att en författare formulerat sig på ett särskilt träffande sätt, när du behöver stöd av en auktoritet, eller när du vill visa att en författare har fel.

Korta citat omges med citationstecken. Att citera Strömqvist kan vara en passande illustration i detta sammanhang: ”Det må vara svårt att skriva, men det är roligt också.”

Långa citat kan återges i form av blockcitat. Textmassan placeras då på sidan utan citationstecken, men med indrag, det vill säga något förskjutet åt höger, och med mindre teckenstorlek. Källan anges i direkt anslutning till citatet.

> Det här är ett blockcitat vilket innebär indragning, mindre teckenstorlek, rak vänstermarginal, inte nödvändigtvis rak högermarginal, och inga citationstecken. Blockcitatet kan tillämpas vid mer än ungefär 50 ord. Blockcitat avslutas alltid med källhänvisning.

Testing

. . . Se section 1.2 . . . Se också chapter 1 . . .

2.4 Källhänvisningar och -förteckning

Att kopiera in en text utan att ange dess källa betraktas som plagiat och därmed allvarligt fusk.

En källförteckning (lista över referenser) upprättas i slutet av rapporten för att ge läsaren en samlad upplysning om samtliga källor som du refererar, citerar eller av annat skäl hänvisar till i den löpande texten. Källor ska anges så noggrant att läsaren ska kunna kontrollera dem, om de finns tillgängliga via bibliotek eller på internet. Det förekommer även att muntliga källor och annan korrespondens inkluderas i källförteckningen, men det är ovanligt i tekniska rapporter.

Använd vederhäftiga källor, gärna författade av auktoriteter på området. Privata hemsidor och studentuppsatser har låg tillförlitlighet som källor, i synnerhet om studentuppsatsen

har lägre nivå (A, B, C eller D) än det egna arbetet. Var källkritisk, särskilt mot kommersiella försäljningsargument.

Ta endast med källor i förteckningen som du refererar eller citerar i den löpande texten, detta sker automatiskt i LaTeXoch BibTeX. Samtliga källor som tas upp i källförteckningen ska vara kopplade till rapporten genom hänvisning i den löpande texten, enligt Vancouver-systemet, som är vanligt förekommande i rapporter i tekniska ämnen.

Enligt Vancouver-systemet ordnas källförteckningen i den ordning källorna återges i den löpande texten, och källhänvisningen anges i texten med en siffra inom hakparenteser, som i detta dokument. Exempel på källhänvisning: Enligt kan dynamiska SFN ge betydande prestandavinster. På liknande sätt refereras till källor på internet, exempelvis kan du se för att läsa om *H.264*.

Eftersom information på webben kan revideras ofta, och eftersom webblänkar kan upphöra att fungera, måste datum anges då du själv hämtade information från webbsidan. Vid webbaserade källor krävs ibland anvisningar för hur källan kan hittas. Tänk på att kvaliteten på materialet på internet varierar.

3 Implementation

3.1 Vulnerabilities

A1-Injection

The Database layer uses Eloquent, the default ORM in Laravel. One of the large advantages to how Laravel is built is that it allows to change what kind of database is used. This system uses MySQL, but it is easy to change it to using SQLite or other drivers. This is doable because it uses PDO and prepared statements. Because it uses those, it is by default more secure against SQL Injection, since the prepared statements will only accept the proper arguments.

For instance, if the `username` column in a table is a string, and the data supplied from the user is "test); DROP TABLE users", that is exactly what the username column will recieve. It will not be interpreted as a command.

A2-Broken Authentication and Session Management

As the system is implemented (using Laravel default session management), the session expires when the browser is closed. It will also close the session if the user goes to the route `users/logout`, after which the user will be redirected back to the main page. However, it is important to know that there is a known feature/bug (per [3] in Chrome where if you have the setting "on startup" set to "continue where I left off", the session *will not* be destroyed.

The sessions are stored in the database, in a table called `sessions`, created by Laravel.

A3-Cross-Site Scripting (XSS)

There are two different routes that the system takes to protect against XSS. Fields that should not contain HTML (and which won't be hashed/encrypted) get `htmlspecialchars` used on them, whereas fields that are expected to contain HTML are stored as-is in the database.

Fields that are expected to contain a certain amount of HTML are then treated right before they're outputted. I have defined an HTML macro (a functionality of Laravel) called `markdown` to clean and transform the specified string. The macro is called in the blade template where it is relevant, so the original data is not transformed.

Note that users are discouraged to use HTML in their posts and such, and instead encouraged to use Markdown, a "text-to-HTML conversion tool for web writers"[4], using the original syntax to be able to work with client side scripts that output a preview of the markdown (the client side scripts are not implemented in this version).

The following steps are taken to protect against XSS:

1. Using a quite permissive list of allowed HTML elements (see `app/config/purifier.php`), it cleans the data, in particular removing "dangerous" attributes, using the HTMLPurifier, "a standards-compliant HTML filter library"[5]. The only allowed attributes are `href`, `class`, `width`, `height`, `alt` and `src`. Style is forbidden not because it is particularly dangerous, but because it risks messing up the design, and all the `on*` attributes are forbidden as JavaScript should not be allowed for authors. An important HTML element that is of course removed is `<script>`, for the same reasons that `on*` being forbidden.

2. PHP Markdown is then used to transform markdown into HTML. This includes some encoding into HTML entities, in particular code contained in codeblocks.

3. Final step of the `HTML::markdown` macro is unrelevant to protecting against XSS, but quite relevant for a more pleasing style, namely using PHP Typography to create "curly quotes" and similar things.

A4-Insecure Direct Object References

At current there are few places where this would be a concern. The system does not use the traditional definition of directories (being an MVC framework), with the exception of the `assets` directory. However, all directories are protected using Apache2 from being listed.

In a future implementation, there will be files that can be accessed, but they will all be stored above the public folder and served using routes to protect them from being exposed to the public.

A5-Security Misconfiguration

The weakest link at the moment is this one. Laravel supports setting up different configs depending on environment, which will be done before the system goes live but is not yet. One of the key components is that rather than showing an error trace in production, it will log the errors and report more cleaned-up ones to the user.

The beginnings of ensuring a repeatable development/deployment/testing/production environment has been set up using Vagrant and shell recipes, but not finalized.

A6-Sensitive Data Exposure

In this system, the e-mail is protected by being encrypted in the database, and using SSL throughout the more sensitive areas. The e-mail is considered "sensitive" as it is the only way to really identify the user, as any other personal information is limited and opt-in only. Another reason can be found in an article by Daniel Cid [2] from 2013 which revisits a hack of PHPBB.com in the beginning of 2009.

The hacker revealed several things, but the key for why I want the e-mails can be found in the following quote:

> So I login and see what I can come across, wow 400,000 registered emails, I'm sure that will go quick on the black market, sorry people but expect a lot of spam

A7-Missing Function Level Access Control

The current iteration of the system does not have more granulated user roles than logged in/guest, but the principles that are applied currently will apply once it does.

The controller `Authorized` uses filters defined in Laravel (the details of individual filters can be found in `app/filters.php`) to determine whether a request should be let through or not. Controllers inheriting `Authorized` (for instance the `Users` controller) defines a whitelist of actions that *does not* require a user to be logged in, and may optionally define a guestlist of actions that requires the user to *not* be logged in.

That is, unless a specified action is in the array of whitelisted actions, before the HTTP-request is served, Laravel checks whether the person is logged in. If they are not, they will be redirected to the `login` route. Once they've logged in, the system will attempt to return them to the page they originally attempted to reach. Examples of whitelisted routes are `login` and `users.create`.

On the other hand, if the action is whitelisted and in the guestlist, the system will check that they are not logged in before giving them access to it. After all, why should someone access the login route when they are already logged in?

A8-Cross-Site Request Forgery (CSRF)

Another of Laravel's builtin filters is a CSRF-token that is made automatically on each form created using Laravel HTML functions. The `Base` controller runs a similar filter to the `Authorized` controller, but where `Authorized` checks the user capabilities, `Base` runs the `csrf` filter on all post-actions. If the token does not match, it throws a TokenMismatchException.

A9-Using Components with Known Vulnerabilities

To make it easier to keep track of which components are used where and to in the extension be able to write tasks to check everything against at least the more comprehensive databases, all components in PHP use Composer, and all frontend components use Bower.

One of the development components for PHP is the Security Checker[6] from SensioLabs, which using a gist by Barry vd. Heuvel[1] was made into a service for Laravel. To check the Composer components against Sensiolabs database, run `php artisan security:check`.

Obviously, running automated checks may not catch all vulnerabilities, but they're a good, automated tool to supplement occasional manual checks, and as all components are documented in either `composer.json` or `bower.json`, it is easier to track them. There is no mentioning of `package.json`, as the NPM modules are only used in development, not in a production environment.

A10-Unvalidated Redirects and Forwards

Not applicable, as no redirects use data supplied from the user to indicate destination.

4 Diskussion

Efter de objektiva resultaten följer kapitlet Diskussion[1], där du presenter dina egna slutsatser, din subjektiva uppfattning, samt kritiskt analyserar resultatens tillförlitlighet och generaliserbarhet.

Om denna del är omfattande kan den delas in i flera kapitel eller under-kapitel, exempelvis ett analys- eller diskussionskapitel med förklaringar till och kritisk granskning av resultaten, ett slutsatskapitel där de viktigaste resultaten och slutsatserna presenteras, samt ett avsnitt med förslag på fortsatt arbete inom området.

Att återknyta till undersökningens syftes- och målformulering hör till det viktigaste i detta kapitel.

Ge gärna utrymme åt svaren på följande frågor:

- Vad är projektets nyhetsvärde och viktigaste bidrag till forskningen eller teknikutvecklingen?
- Har projektets mål uppnåtts? Har uppdraget utförts?
- Vad är svaret på den inledande problemformuleringen?
- Har resultatet blivit det väntade?
- Är slutsatserna generella, eller gäller de bara under vissa förutsättningar?
- Vilken betydelse har metod- och modellvalet för resultaten?
- Har nya frågor väckts på grund av resultatet?

Den sista frågan inbjuder till möjligheten att ge förslag till andra, anknytande undersökningar, det vill säga förslag dels till åtgärder och rekommendationer, dels till fortsatt forskning eller utveckling för den som vill bygga vidare på ditt arbete.

I tekniska rapporter på uppdrag av företag presenterar du här den rekommenderade lösningen på ett problem. Du kan då göra en konsekvensanalys av lösningen ur tekniskt såväl som lekmannaperspektiv, till exempel i fråga om ekonomi, miljö och förändrade arbetsrutiner. Kapitlet innehåller då rekommenderade åtgärder samt förslag på vidare utveckling eller forskning, och utgör således beslutsunderlag för uppdragsgivaren.

[1] Alternativa rubriker är Slutsatser eller Analys.

Bibliography

[1] Barry vd. Heuvel. *Symfony Security Checker in Laravel*. Fetched the 19 October 2013. URL: https://gist.github.com/barryvdh/6696739.

[2] Daniel Cid. *Security Archive – Case Study: phpbb.com Compromised*. Fetched the 19 October 2013. URL: http://blog.sucuri.net/2013/09/security-case-study-archive-phpbb-com.html.

[3] Google Groups User. *Chrome is not deleting temporary cookies*. Fetched the 19 October 2013. URL: https://productforums.google.com/d/msg/chrome/9l-gKYIUg50/HOvdZbPiuXAJ.

[4] John Gruber. *Daring Fireball: Markdown*. Fetched the 19 October 2013. URL: http://daringfireball.net/projects/markdown/.

[5] MeWebStudio. *HTML Purifier*. Fetched the 19 October 2013. URL: http://htmlpurifier.org/.

[6] SensioLabs. *Security Checker*. Fetched the 19 October 2013. URL: https://packagist.org/packages/sensiolabs/security-checker.

[7] Wikipedia. *H.264/MPEG-4 AVC*. Fetched the 19 October 2013. URL: http://en.wikipedia.org/wiki/Multitier_architecture.