Web Verification for PHP

Weverca

1. Motivace

Programovací jazyky s dynamickým typovým systémem a dalšími dynamickými vlastnostmi jsou oblíbené pro svoji flexibilitu, která může výrazně usnadnit a zrychlit vývoj aplikací. Tato flexibilita ale komplikuje automatickou detekci chyb a automatizaci důležitých programátorských operací jako je navigace v kódu a refactoring. Dynamické programovací jazyky se přitom používají i při vývoji internetových aplikací. Ty často obsahují citlivá data a v nich obsažené bezpečnostní chyby mohou způsobit velké škody. Dostupnost nástrojů pro zajištění jejich bezpečnosti a spolehlivosti je proto velmi důležitá. V současné době takové nástroje pro dynamické jazyky buď fungují nespolehlivě nebo nejsou vůbec k dispozici. Výjimkou jsou nástroje pro takzvanou black-box analýzu internetových aplikací. Tyto nástroje testují aplikace s použitím dat z knihoven nebezpečných vstupů a jejich využití a zejména pak jejich přesnost jsou proto omezené.

PHP je v oblasti webových aplikací nejpoužívanější programovací jazyk s mnoha dynamickými vlastnostmi. Především to je dynamický typový systém. Typ proměnných není deklarovaný, navíc jedna proměnná může obsahovat data různých typů na různých místech programu nebo dokonce i na stejném místě programu v různých kontextech. Objektový model PHP umožňuje za běhu přidat nové atributy a metody k objektu bez jeho předchozí deklarace. PHP obsahuje také konstrukt “variable-variable”, který umožňuje použít proměnnou, jejíž název je dán hodnotou jiné proměnné. Dále PHP umožňuje řídit includování souborů pomocí informací známých až za běhu, například hodnot proměnných. Další konstrukt problematický z pohledu (bezpečnostní) analýzy je příkaz “eval” umožňující interpretovat libovolný řetězec jako kód programu. Specifikem jazyka PHP a internetových aplikací je také častá manipulace s řetězci a častá manipulace s daty pomocí (z pohledu analýzy) složitých datových struktur jako jsou například asociativní pole. To dále zvyšuje požadavky na analýzu těchto aplikací.

2. Cíle projektu

Softwarový projekt má za cíl vyvinout framework pro statickou analýzu zdrojového kódu aplikací napsaných v jazyce PHP a výsledky analýzy použít pro jejich bezpečnostní analýzu a analýzu kvality kódu. Framework bude navržený tak, aby ho bylo možné v budoucnu použít pro další úkony jako je refactoring, navigace v kódu, hledání běhových chyb, zobrazení informací užitečných pro vývojáře ve vývojovém přostředí (možné typy, hodnoty a aliasy proměnných na daném místě v programu, místo posledního přiřazení do proměnné, atp.)

Framework bude začleněn do projektu *Phalanger* (http://phalanger.codeplex.com/), ze kterého bude používat zejména část pro parsování zdrojových kódů PHP a vytváření derivačního stromu (abstract syntax tree).

Nástroj dále vytvoří control-flow graf a provede statickou analýzu. Statická analýza bude používat vhodnou abstrakci, která zohlední specifika jazyka PHP a webových aplikací. Tedy zohlední velké množství informací, které nejsou přímo staticky známy, vkládání souborů na základě těchto informací, volání metod a přístup k atributům objektů za absence staticky deklarované typové informace, objektový model PHP, manipulaci s daty za použití asociativních polí a možnost tvorby

aliasů mezi přoměnnými.

Výsledek statické analýzy budou informace o možných hodnotách (včetně informace, zda může být daná hodnota ovlivněná uživatelským vstupem), typech, aliasech každé proměnné pro každý program point. Výstupem bezpečnostní analýzy pak bude seznam potenciálních bezpečnostních problémů ve vstupní aplikaci s dostatečným množstvím informací pro identifikaci zdrojů těchto problémů.

Analyzátor pak na základě těchto informací odhalí bezpečnostní chyby v důsledku toku dat z nedůvěryhodných zdrojů (uživatelský vstup, databáze, cookies) do kritických příkazů (posílání dat do prohlížeče, manipulace s databází). Další užitečný výstup bude upozornění na syntaktické chyby v aplikaci (PHP poskytuje pouze kontrolu až při interpretaci), upozornění na nedoporučené použití PHP konstruktů a informace o některých mertrikách kódu (code metrics) poukazující na nekvalitní kód.

3. Části projektu

Projekt se bude skládat z těchto částí:

Modul pro analýzu na syntaktické úrovni: tento modul bude pracovat nad AST reprezentací a bude hledat použití nedoporučených konstrukcí a bude vyhodnocovat metriky kódu.

- CFG reprezentace: pro účely statické analýzy kódu bude kód z AST převeden do reprezentace control-flow graph (CFG). Ta bude navržena s ohledem na to, že control-flow graf programu může být ovlivněn množstvím informací, které nejsou přímo staticky známy (jména includovaných souborů konstruovaných za běhu). To znamená, že hrany CFG grafu odpovídající cílům vkládání souborů a cílům volání funkcí nebudou počítány při převodu z AST do CFG reprezentace, ale budou počítány až při samotné statické analýze.
- Paměťový model: umožní rozlišit, jaké informace v daném řádku programu jsou platné pro všechny vstupy programu a jaké platí pouze pro některé. Paměťový model bude také definovat, jak se tyto informace budou v reprezentaci propagovat v případě přiřazení do proměnné a příkazu pro vytvoření aliasu. Paměťový model bude zohledňovat práci s asociativními poli s objektový model PHP.
- Statický analyzátor: bude založený na standardním výpočtu fix pointu informací uložených v uzlech CFG reprezentace. Výstup budou informace o každém bodu v programu (uzlu v CFG grafu). To bude zahrnovat informace o možných hodnotách, typech a aliasech proměnných v každém místě programu.
- Bezpečnostní analyzátor: na základě informací ze statického analyzátoru odhalí bezpečnostní chyby v důsledku toku dat z nedůvěryhodných zdrojů a u každé chyby vypíše její příčinu. Tedy způsob, kterým se informace z nedůvěryhodného zdroje mohou dostat na kritické místo v programu. Nástroj bude schopen detekovat problémy typu SQL injection a Cross-site-scripting.

4. Další informace

Nástroj bude implementován v jazyku C#.

Projekt je určen pro šest studentů.

Předpokládané zahájení projektu: ihned od schválení komisí
Předpokládaná doba trvání: 9 měsíců od zahájení

Seznam členů projektového týmu:

David Škorvaga
Marcel Kikta
Matyáš Brenner
Michal Staša
Miroslav Vodolán
Pavel Baštecký