文字列ごとの情報フロー追跡手法のPHPへの実装

都井 紘[†]　塩谷 亮太[††]
五島 正裕[††]　坂井 修一[††]

1. はじめに

近年，クロス・サイト・スクリプティングやSQLインジェクションといったWebアプリケーションの脆弱性を突いたインジェクション・アタックによる被害が深刻化している．インジェクション・アタックとは，プログラムの入力に攻撃コードなどを与えて（インジェクション），プログラマの意図に反する動作を起こさせる攻撃のことである．

インジェクション・アタックを検出する方法として**DTP(Dynamic Taint Propagation)**がある．DTPでは外部から入力されたデータにテイント(汚染)というマークをつける．このテイント情報を，演算の入力から出力に対して伝搬させる．テイント情報のついたデータが，システム・コールの入力に用いられるなどの"危険な"使われ方をした場合に，これを攻撃として検出する．DTPにはテイント情報の伝搬をソフトウェア上で行う言語依存DTPと，ハードウェア上で行うものプラットフォームDTP（PDTP）がある．

既存のPDPTは言語環境に依存せず適用できるプログラムがほとんど限定されず包括的である．また，実行速度のオーバーヘッドも小さくなる．しかし，データの依存関係に従って命令単位でテイント情報を伝搬させるためその伝搬精度が十分とは言えず，誤検出や検出漏れを生じる．

そこで李らは，文字列操作を識別して文字列から文字列にデータが移動する時にテイント情報を伝搬させる手法として**SWIFT(String-Wise Information Flow Tracking)**を提案している．これは，命令単位ではなくよりセマンティックな文字列操作の単位で伝搬させるので既存のPDTPより高精度な伝搬を実現する．しかし，SWIFTはハードウェアに実装する必要があるため，普及のハードルが高い．

そこで，本研究では，このSWIFTをPHPに対して実装することを目的とする．PHPはWeb用スクリプト言語として現在最も多く用いられている．SWIFTをPHPに実装することにより適用範囲がPHP上に制限されるが，PHPの範囲内では高精度なものが作れる．またSWIFTの普及が容易となる．

† 東京大学 工学部
Faculty of Engineering, The University of Tokyo
†† 東京大学大学院 情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo

2. SWIFT

2.1 従来手法とその問題点

まず，従来のDTPは命令間の依存関係のみに着目してテイント情報を伝播させるために問題がある．

データの依存関係には，データの転送や演算などの直接的な依存だけでなく，

```
$o = table[$i]; //$iから$oへの依存
```

のようなアドレス依存や，

```
if($i == '+') $o = ' ';//$iから$oへの依存
```

のような制御依存といった間接的な依存関係がある．間接的な依存は扱いが難しい．

すべての実用的なアプリケーションの出力はユーザーの入力に依存しているため，直説的な依存関係だけでなく間接的な依存関係でもテイント情報を伝播させてしまうとすべての出力にテイントがついてしまい意味がなくなる．そこで実際には，言語依存DTPではヒューリスティクスに基づきサニタイズされたものやテーブルを参照する時はテイントしない，また，PDTPではアドレス依存は伝播させないなどの非伝播ポリシーが用いられている．しかし，このため検出漏れや誤検出が生じる．

また，多くのWebアプリケーションのインターフェースとしてラジオボタン式とテキストボックス式があるが，これらを区別できないという問題もある．前者はプログラマが与えた文字列をユーザーが選択するだけであるので安全であるが，後者はユーザーが任意の文字列を入力することができるため危険である．

2.2 SWIFTの原理

SWIFTでは，load及びstore命令のメモリアドレスの動的な取得により文字列及び文字列操作を識別し，

表1 WEBアプリケーションでの比較

program	attack	SWIFT		Raksha-a		Raksha+a		PHP-taint	
		FN	FP	FN	FP	FN	FP	FN	FP
phpSysInfo 2.3	XSS						√		
Qwikiwiki 1.4.1	directory traversal						√	√	
phpBB 2.0.8	SQL injection						√	√	
phpBB 2.0.8	XSS						√	√	
PHP-Nuke 7.5	SQL injection						√	√	
CubeCart 3.0.3	XSS			√			√	√	
PHP-Nuke 7.1	XSS			√			√	√	
PHP-Nuke 7.1	SQL injection			√			√	√	

FN: false negative FP: false positive

文字列の移動元から移動先へテイント情報を伝播させる．このため命令間のデータ依存関係に透過的な伝播が可能となり，従来の伝播精度の低下の原因となる間接的依存にも対応可能となる．さらにラジオボタン式とテキストボックス式の区別をすることができる．

文字列はメモリ領域の連続した領域に格納されていることから識別し，また，文字列操作は移動元の文字列のロードと移動先の文字列のストアが交互に行われることから識別する．

2.3 SWIFTの評価

脆弱性の報告されているWebアプリケーションにおいて検出精度を比較したのが表1である．Raksha[3]とはPDTPの最新のものであり，アドレス伝播をしないものとするものがそれぞれ-aと+aである．

SWIFTでは誤検出も検出漏れもなく高い検出精度があることがわかる．

3. PHPへの実装

SWIFTはメモリアドレスのみに着目すればよいのでインタプリタ上にもプロセッサ上にも実装することができる．インタプリタ上に実装する利点は，言語に依存した情報を使える点やスクリプトをインタープリットする際のノイズの影響を受けない点である．

3.1 関連研究

PHPにDTPを実装した過去研究として，Nguyen-Tuongらの手法[1]，Wietse Venema[2]の手法が挙げられる．

前者は文字列を操作する，または返す関数毎にデータの依存関係を定義しているのでその範囲では伝播精度は高いが，アドレス依存には対応しておらず伝播精度が十分ではない．

また，後者は伝播のルールが単純なため実行速度やメモリのオーバーヘッドは少ないが伝播精度は十分ではない．

3.2 実装方針

インタープリタはC言語で書かれている．

PHPにおいて$_GETや$_POSTといったスーパーグローバル変数が外部からの入力に依存するので，まずこれらの変数のメモリアドレスにテイントをつける．あとは変数に触る部分からメモリアドレスを取得していけばよい．

PHPではスクリプトはopcodeと呼ばれる中間言語にコンパイルされてから実行される．この中間言語には実体は全てzvalという構造体であるが，CV，VAR，TMPと呼ばれる三つの変数が存在する．opcodeの処理系においてこれらの三つの変数にアクセスするところからメモリアドレスを取得していけばよいと考えられる．

ただし，base64_encode()などの一部のnative関数については個別に対応してアドレスを取得する．

4. おわりに

本稿では，文字列操作の単位でテイント情報を伝播させるため高精度な伝播を実現するSWIFTとそのPHPへの実装方針について説明した．

実装後は実行速度のオーバーヘッドの測定や誤検出・検出漏れがないかの確認を行う予定である．

参考文献

1) D.Greene J.Shirley A.Nguyen-Tuong, S.Guarnieri and D.Evans. Automatically hardening web applications using precise tainting. In 20th IFIP International Information Security Conference (SEC), pp. 295307, 2005.
2) Wietse Venema. Taint support for php. http://wiki.php.net/rfc/taint.
3) Michael Dalton, Hari Kannan, and Christos Kozyrakis. Raksha: A flexible information flow architecture for software security. In 34th Int' l Symp. on Computer Architecture, pp. 482—493, 2007.