Vol.2016-MPS-111 No.21
2016/12/12

機械学習によるクロスサイトスクリプティング攻撃検知と誤検知要因の分析

梅原章宏[†1]　松田健[†2]　園田道夫[†1]　水野信也[†3]　趙晋輝[†4]

概要：近年のインターネット普及に伴い,Web アプリケーションを構成する HTML の入力部分などにおける脆弱性を利用したクロスサイトスクリプティング攻撃(XSS 攻撃)による被害が増加している.対策として機械学習を用いたアノマリ式の検知などが用いられているが,学習データの設定はランダムサンプリングや経験的な手法などを用いることが多い.そのため,どのような学習データが有効であるかの調査が必要であると考えられる.本研究では XSS 攻撃入力と,正常に用いられる URL などの入力に対して ASCII 文字の出現頻度に基づく特徴ベクトルを構成し,機械学習アルゴリズムである SVM と SCW を用いた攻撃検知を行った.また,検知結果の分析を行うことで,各アルゴリズムが作り出す特徴空間の構造を調べる方法と,誤検知を減らすための手法について考察する.

キーワード：機械学習, クロスサイトスクリプティング攻撃(XSS 攻撃), 攻撃検知

1. はじめに

近年インターネットの普及が進み,それに伴い個人情報を含む取引を行う Web アプリケーションが増加している.その一方で,個人情報の窃取や個人・法人に対する成りすましを目的としたサイバー攻撃も同様に増加している.

クロスサイトスクリプティング攻撃(以下 XSS 攻撃)はこのような被害を引き起こすサイバー攻撃の一種である.XSS 攻撃は Web アプリケーションを構築する HTML において,入力部分や外部入力されたパラメータを扱う部分などにおける脆弱性に対して攻撃を行う[1].従来の対策として,構文解析によるフィルタが提案・実現されている[2]が,XSS 攻撃に用いられる攻撃入力は,正常に用いられるスクリプトなどとの区別が難しく,機械的な攻撃検知が容易ではない.

また,別の方法として機械学習を用いたアノマリ式の攻撃検知の方法についても様々な研究[3]が行われている.しかし,機械学習アルゴリズムに用いられる学習データの設定は,ランダムサンプリングや経験的手法を用いることが多く,それらを利用する場合は本当に効率的な学習が行われているかの判断をすることが難しい.そのため,どのような形式の学習データが有効であるかの調査を行うことは重要であると考えられる.

本研究では,攻撃入力と正常入力に含まれる標準 ASCII 記号の頻度を基とした特徴ベクトルを先行研究とは異なる方法で生成し,それを用いて機械学習アルゴリズムである Support Vector Machine (SVM), Soft-Confidence Weighted Learning (SCW) [4] による攻撃検知を行った.その後検知結果を分析することで,各アルゴリズムが作り出す特徴空間構造を調べる方法と,攻撃検知における誤検知を減らすための手法について考察した.

†1 中央大学大学院理工学研究科情報工学専攻.
Departments of Information and System Engineering, Graduate School of Science and Engineering, Chuo University.
†2 長崎県立大学情報システム学部情報セキュリティ学科.
Department of Information Security, Faculty of Information Systems, University of Nagasaki.
†3 静岡理工科大学総合情報学部コンピュータシステム学科.
Department of Computer Science, Faculty of Comprehensive Informatics, Shizuoka Institute of Science and Technology.
†4 中央大学理工学部情報工学科.
Departments of Information and System Engineering, Faculty of Science and Engineering, Chuo University.

2. XSS 攻撃[1]

XSS 攻撃は HTML で構成された脆弱性のある入力部分などに対して,不正なスクリプト文を入力することにより,アプリケーション開発者が本来意図しない動作を誘発させるサイバー攻撃である.主な被害としては, Cookie に格納されているセッション ID などの窃取による成りすまし被害や,ポップアップなどを利用した Web ページなどの改ざんが挙げられる.

既存の対策の一つとして,あらかじめ登録した識別規則(シグネチャ)を基に与えられた入力を処理する方法がある.主な手法としてはブラックリスト・ホワイトリスト方式が挙げられる.しかし,この方法はあらかじめシグネチャを登録しておく必要があるため,登録されていない未知の攻撃に対して対応することが難しく,またシグネチャの数が増加すると登録作業が煩雑となることが考えられる.

別の対策手法として入力に対するエスケープ処理がある.これは HTML において特殊な意味を持つ記号('&', '<', '>' など)を別の文字記号で置き換えることにより無害化する方法である.エスケープ処理は対象となる記号や入力部分などに漏れなく行うことが可能であれば, XSS 攻撃を防ぐための根本的な対策として非常に有効である.しかし,例外処理の対象となる記号が増加しプログラムの記述が煩雑となった場合,処理を行う部分の記述に漏れが生じる可能性が高くなってしまい,結果として脆弱性が生じる恐れがある.

3. 機械学習アルゴリズム

本章では実験で用いた機械学習アルゴリズムの概要について述べる.

3.1 SVM

SVM は与えられた入力に対してバッチ式の教師あり学

習を行う機械学習アルゴリズムの一つである. SVM によるデータ分類は以下の式で定義される[5].

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) + b$$
$$= \sum_{n=1}^{N} a_n t_n k(\mathbf{x}, \mathbf{x}_n) + b$$

ここで $\mathbf{x}$ は入力ベクトル, $\mathbf{w}$ は重みベクトル, ϕ は特徴空間変換関数, a_n はラグランジュ乗数, b はバイアスパラメータである.また $k(\mathbf{x}, \mathbf{x}_n)$ はカーネル関数である.

SVM は分類境界と最も近いデータとの距離であるマージンの最大化を行うことで,汎化誤差が最小になるような分類境界を求める.マージンを最適化する解は以下の目的関数を最小化することで求められる.

$$C\sum_{n=1}^{N} \xi_n + \frac{1}{2}\|\mathbf{w}\|^2$$
$$\text{s.t.} \quad t_n(y(\mathbf{x}_n)) \geq 1 - \xi_n, \quad n = 1, \ldots, N$$
$$\xi_n \geq 0$$

ここで C は誤分類に対するペナルティの大きさを制御するパラメータであり,大きいほど誤分類を許さないような分類境界を求める.C の値が大きくなりすぎると,データに対する汎化性が失われてしまう.また ξ_n はスラック変数であり, $0 \leq \xi_n \leq 1$ となるデータは正しいクラスに分類されている.一方で $\xi_n \geq 1$ となるデータは誤ったクラスに分類されている.

3.2 SCW

SCW は与えられた入力に対してオンライン式の教師あり学習を行う機械学習アルゴリズムである. SCW は同じオンライン式の機械学習アルゴリズムである CW[6]を改良したものであり,特徴としてデータの信頼度による重みづけを行う.また, SVM のようなマージンの最大化することで, 本来の CW において弱点であったノイズに弱い面を克服している.

SCW によるデータ分類は以下の式で定義される.

$$\hat{y}(x) = \mathrm{sgn}(\boldsymbol{\mu}_{t-1}\mathbf{x}_t)$$
$$\text{if } \boldsymbol{\mu}\mathbf{x} \geq 0 : \hat{y}(\mathbf{x}) = 1$$
$$\text{else} \quad : \hat{y}(\mathbf{x}) = -1$$

ここで $\mathbf{x}$ は入力ベクトル, $\hat{y}(\mathbf{x})$ はデータ x がどのクラスに属するかの予測ラベル, $\boldsymbol{\mu}$ は重みをあらわす平均ベクトルである.バイアスパラメータは存在しない.

また,損失関数 l^{ϕ} は以下の式であらわされる.

$$l^{\phi} = \max(0, \phi\sqrt{\mathbf{x}_t^{\mathrm{T}}\Sigma\mathbf{x}_t} - y_t\boldsymbol{\mu}\mathbf{x}_t)$$

ここで y はデータ $\mathbf{x}$ がどのクラスに属するかの正解ラベル, $\boldsymbol{\Sigma}$ は共分散行列, $\phi = \Phi^{-1}(\eta)$である.Φ は正規分布の累積密度関数, η は誤差を許容するパラメータをあらわす.

$\boldsymbol{\mu}, \boldsymbol{\Sigma}$ の更新式は以下の最適化問題であらわされる.

$$(\boldsymbol{\mu}_{t+1}, \boldsymbol{\Sigma}_{t+1}) = \arg\min_{\boldsymbol{\mu},\boldsymbol{\Sigma}} D_{KL}(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \| \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t))$$
$$+ Cl^{\phi}(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}): (\mathbf{x}_t, y_t))$$

ここで D_{KL} はカルバック情報量, $\mathcal{N}$ は $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ による多変量正規分布, C は重みの更新を制御するパラメータである.

4. 実験準備

本章では検知実験を行う上で必要なデータセットや機械学習アルゴリズムに用いる特徴ベクトルの生成法,実験の評価指標について述べる.

4.1 実験用データセットの作成

検知実験を行うにあたり,以下の 4 パターンの入力を収集した.

1. URL 形式の攻撃入力(AU)
2. スクリプト形式の攻撃入力(AS)
3. URL 形式の正常入力(NU)
4. TeX で用いられる数式形式の正常入力(NF)

学習データセットとしては,上記 4 つの入力集合からそれぞれ 50 個ずつをランダムサンプリングすることで,要素数 200 のデータセットを生成した.

テストデータセットとしては, 0~600 個目のデータまでは上記 4 つの入力集合からそれぞれ 150 個ずつ, 601~800 個目のデータは 2 種類の攻撃入力をそれぞれ 100 個ずつ, 801~1000 個目までのデータは 2 種類の正常入力をそれぞれ 100 個ずつ,合計 1000 個のデータをそれぞれランダムサンプリングした.また,テスト時において与えられる入力の傾向が変化した際の挙動を調べるために,テストデータについて 200 データを 1 つのフェーズとし,フェーズごとに検知結果を計測した.各データセットの内訳を表 1 に示す.

表 1 各データセットの内訳

	AU	AS	NU	NS	Total
Learning	50	50	50	50	200
Phase1	150	150	150	150	200
Phase2					200
Phase3					200
Phase4	100	100	0	0	200
Phase5	0	0	100	100	200
Test Total	250	250	250	250	1000

4.2 特徴ベクトルの生成

生成した各データセット内の各入力に対して標準 ASCII 記号の出現頻度に基づく特徴抽出を行い, 128 次元の特徴ベクトルに変換を行った.変換の過程を以下に示す.

1. N個のデータからなるデータセット $\mathbf{X}$ を以下のベクトルであらわす..

$$\mathbf{X} = \begin{pmatrix} \vdots \\ X_t \\ \vdots \end{pmatrix} \quad t = 0,1, \ldots, N$$

あるデータ X_t において 10 進 ASCII コードにおける

0-127 までのそれぞれの記号における出現頻度を $x_i\ (i = 0,1,\dots,127)$ とする.データセット $\mathbf{X}$ 中のある 1 つのデータ X_t はベクトル

$$X_t = (x_0, x_1, \dots, x_i, \dots, x_{127})$$

であらわすことができる.

2. データベクトル X_t の各要素 x_i に定数値 c を足し,

$$X_t = (x_0 + c, x_1 + c, \dots, x_i + c, \dots, x_{127} + c)$$

とする.今回の実験では $c = 0.27$ として生成を行った.

3. データベクトル X_t の各要素 x_i を X_t の全ての要素の和 $\sum_{k=0}^{127} x_k$ で割ることで 1 に正規化する.特徴ベクトルに変換後のデータセット $\mathbf{X}'$ は以下のように表すことができる.

$$\mathbf{X}' = \begin{pmatrix} \vdots \\ X'_t \\ \vdots \end{pmatrix} \quad t = 0,1,\dots,N$$

$$\text{s.t.} \quad X'_t = \{(\cdots, x'_i, \cdots) \mid i = 0,1,\dots,127\}$$

$$x'_i = \frac{x_i + c}{\sum_{k=0}^{127} x_k}$$

4. 1. ~ 3. のプロセスをデータセット内の全てのデータに対して行う.

上記のプロセスで生成された特徴ベクトルを機械学習アルゴリズムの入力ベクトルとして用いた.

4.3 評価指標

検知実験の評価項目として,以下を用いた.

1. 正解率(Accuracy) ... データ全体に対して予測が正しかったものの割合
2. 精度(Precision) ... 予測したラベルのうち,真の結果と一致するものの割合
3. 再現率(Recall) ... 真の結果のうち,予測したラベルと一致するものの割合
4. F 値(F-measure) ... 精度と再現率の調和平均であり,総合的な性能の指標

5. 検知実験

5.1 機械学習に用いたプログラム

実験に用いた機械学習アルゴリズムは SVM, SCW の 2 種類である. SVM の実行プログラムには Python の機械学習ライブラリである scikit-learn0.15.2 の関数 SVC を利用した. SVM のカーネル関数は以下の式であらわされるガウスカーネルを用いた.なお,カーネル関数の式は上記ライブラリのマニュアル[7]に準拠するものである.

$$k(x, x') = \exp\left(\frac{1}{2\sigma}|x - x'|^2\right)$$

SCW の実行プログラムは,[4]に記載されたアルゴリズムを参考に Python で実装した. Python のバージョンは 2.7.8, OS は Windows7 を使用した.

5.2 検知実験

SVM と SCW それぞれを用いた場合のフェーズごとの検知実験結果をそれぞれ表 2,表 3 に,全体を通しての結果のグラフを図 1 に示す.

表 2 SVM による検知結果(単位:%)

	Ph.1	Ph.2	Ph.3	Ph.4	Ph.5	Total
正解率	85.0	79.0	83.0	98.5	63.5	81.8
攻撃精度	100.0	100.0	100.0	98.5	---	99.6
正常精度	71.4	58.4	63.8	---	63.5	64.3
攻撃再現率	76.0	70.2	75.7	100.0	---	80.5
正常再現率	100.0	100.0	100.0	---	100.0	100.0
攻撃 F 値	86.4	82.5	86.2	99.2	---	88.6
正常 F 値	83.3	73.8	77.9	---	77.7	79.2

表 3 SCW による検知結果(単位:%)

	Ph.1	Ph.2	Ph.3	Ph.4	Ph.5	Total
正解率	92.0	96.0	95.0	94.5	93.0	94.1
攻撃精度	91.6	97.0	95.3	94.5	---	94.6
正常精度	92.4	95.0	94.7	---	93.0	93.8
攻撃再現率	91.6	95.0	95.3	100.0	---	95.5
正常再現率	92.4	97.0	94.7	---	100.0	96.0
攻撃 F 値	91.6	96.0	95.3	97.2	---	95.0
正常 F 値	92.4	96.0	94.7	---	96.4	94.9

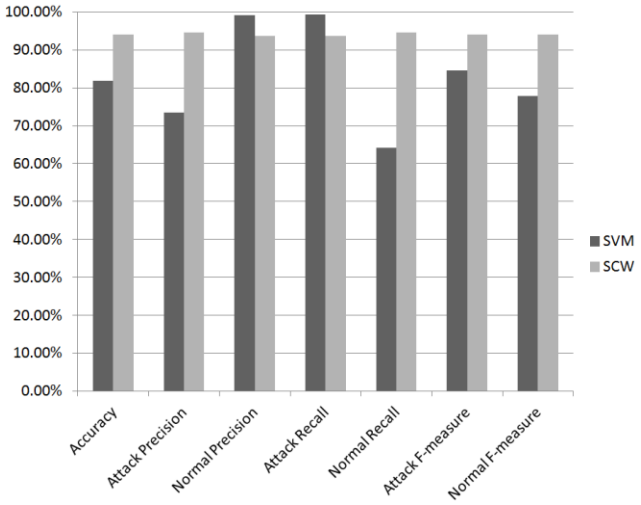

図 1 SVM, SCW の Total の比較

SVM は攻撃入力の検知精度が SCW より優れているものの,正常入力の検知精度が大きく劣っており,これが正解率や F 値を押し下げた原因であると考えられる.特にフェーズ 4 とフェーズ 5 の結果をみると,攻撃入力は約 98%と高いレートで検知できているのに対して正常入力が 63%程度までしか検知できていない.一方の SCW はどのフェーズにおいても 94%-95%程度の安定した正解率と F 値を見せている.このことから,今回の特徴ベクトル生成法を用いて作られたデータセットは, SCW を用いた検知において有効である可能性がある.この予想に確証を持つためには,同様な形式

のデータセットを複数作成して実験を繰り返すことが必須であると考えられる.

6. 考察

先行研究[8][9]では特徴ベクトルは 4.2 節における 1.の頻度をそのまま数え上げたベクトルを使用し,結果としては実数値として SVM の方が優れたスコアを出していた.一方で今回のベクトル生成法を用いた場合には, SCW の方が正解率, F 値ともに優れていた.

SVM の結果が悪くなったのは正常入力の検知精度が悪くなったからであるが,検知精度が悪くなった一つの理由としてベクトルの各要素を 1 に正規化したことが原因に考えられる.これによりデータの各要素が[0,1]という狭い範囲に密集したことで,データ集合を明確に分離することができなくなり,結果として境界平面を引いた際の攻撃入力側に多数の正常入力が混ざってしまったことが推測される. 別な理由としては先行研究[9]で観測された, SVM はバッチ学習のためにテストデータの形式が変わると対応できないことや,今回の特徴抽出法では SVM の分類アルゴリズムにとっては特徴をうまく見出すことができない不都合なデータになってしまったことなどが候補として考えられるが, これに関しては実験を重ねて詳細に分析していくことが必要であると考えられる.

一方今回の実験において SCW は非常に安定して高いスコアを記録し, SVM とは逆に今回の特徴抽出法で生成したベクトルでうまく特徴を見出すことができた可能性が高いと考えられる.また, SCW はオンライン学習であることから,フェーズ 4 とフェーズ 5 においてデータの傾向が変化した際にも,対応することができていると予想される.

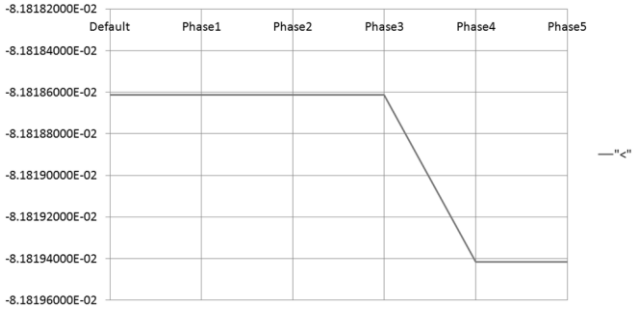

図 2 SCW における $x_{60}('<')$の重みベクトルの変化

図 2 は x_{60} ('<')に対応する重みベクトル $\boldsymbol{\mu}$ の変化を抜粋しグラフ化したものである.グラフを見ると,フェーズ 3 終了時からフェーズ 4 終了時の間に重みが大きく変化していることがわかる.ほかの要素においてもこの部分を境に重みが大きく変化しているものが多かった.このことから,フェーズ 4 が開始した時点で盛んに境界平面の再学習が行われていることが予想される.このことから, SCW はテストデータの傾向の変化を検知できる可能性があり,その通りであれば再学習に最適なタイミングを知ることができることが予想される.しかし,変化量は 10^{-6}~10^{-12}程度のごく微量な値であるため,汎化性を高めノイズに強くなるようにアルゴリズムを設計しているとはいえ,外れ値を含むデータが境界平面に悪影響を与える可能性は少なからず考えられる.どのくらいの値が外れ値となるのか,そのような外れ値が与えられても頑健であるか,頑健でない場合はどのような方策をとるべきなのかという点が今後の課題として考えられる.

7. まとめ

今までの先行研究で用いてきた特徴ベクトル生成法では SVM が優位な結果となることが多かったが,本研究では異なる特徴ベクトル生成法を用いることで,手法によっては SVM の検知精度が必ずしも優位になるとは限らないという結果を得た. SVM の検知精度が下落した理由については,実験を繰り返してさらに分析をする必要がある.

一方の SCW においては今回の特徴ベクトルを用いることにより先行研究より良い結果を得ることができた.また重みベクトルの変動に着目することで,テストデータの傾向が変動するタイミングを検知することができる可能性を見出した.しかし変化量はごく微小であるため,ノイズが与えられても頑健であるような数値尺度を探していくことが, 今後の課題として考えられる.

参考文献

[1]"安全なウェブサイトの作り方 改訂第 7 版", 独立行政法人情報処理推進機構セキュリティセンター, "https://www.ipa.go.jp/security/vuln/websecurity.html"
[2] "IE8 Security Part Ⅳ: The XSS Filter - IEBlog - SiteHome - MSDN Blogs", "http://blogs.msdn.com/b/ie/archive/2008/07/02/ie8-security-part-iv-the-xss-filter.aspx, (最終閲覧日:2015/1/7)
[3] 伊波 靖, 高良 富夫, "SVM を用いた WAF への異常検知機能の実装と評価", 情報処理学会第 74 回全国大会
[4] Jialei Wang, Peilin Zhao, Steven C.H. Hoi, "Exact Soft Confidence-Weighted Learning", ICML(2012)
[5]Christopher M.Bishop, "PATTERN RECOGNITION AND MACHINE LEARNING", Springer(2006)
[6] Mark Dredze, Koby Crammer, Fernando Pereira, "Confidence-weighted Linear Classification", ICML, pp.264-pp.271 (2008)
[7] "scikit-learn: machine learning in Python — scikit-learn 0.18 documentation", "http://scikit-learn.org/stable/", (最終閲覧日:2016/11/11)
[8] 梅原 章宏, 松田 健, 園田 道夫, 水野 信也, 趙 晋輝, “機械学習を用いたクロスサイトスクリプティング(XSS)攻撃の検知に関する考察”, 情報処理学会第 71 回 CSEC 研究会研究報告
[9] 梅原章宏, 松田健, 園田道夫, 水野信也, 趙晋輝, "クロスサイトスクリプティング**(XSS)** 攻撃の検知におけるバッチ学習とオンライン学習の比較実験", 情報処理学会第 78 回全国大会
[10]Jeremiah Grossman, Robert "Rsnake" hansen, Petko "pdp" D.petkov, Anton Rager, Seth Fogie, "XSS ATTACKS", SYNGRESS(2007)
[11] Wade Alcorn, Christian Frichot, Michele Orru, 園田 道夫, はせがわようすけ, 西村 宗晃, "ブラウザハック", 株式会社翔泳社(2016)