卒業論文 2018 年度 (平成 30 年)

低対話型 Honeypot のコマンド拡張による
高対話型 Honeypot への近似

慶應義塾大学 総合政策学部
菅藤 佑太

卒業論文要旨 - 2018 年度 (平成 30 年度)

低対話型 Honeypot のコマンド拡張による 高対話型 Honeypot への近似

PC の普及や IoT デバイスのシステム高度化により, 高度な処理系を組むことが可能になった. これによりデバイス上に Linux 系などの OS が搭載された機器が広く人々に使われるようになった. また,Linux 系 OS にリモートログインする手法として SSH がある．これを用いて不正に侵入する攻撃が行われている. 侵入された際に侵入者がどのような挙動をしているのかを知る手段として,Honeypot がある.Honeypot は SSH で侵入しやすいような環境を作ることで, 侵入者にログイン試行に成功したと検知させ, その際に実行したコマンドのログを収集するものである. また現在では Shell の挙動をエミュレートした Honeypot が広く使用されており, この Honeypot は実行できるコマンドが少ない実装になっている. そのため Honeypot への侵入者に侵入先が Honeypot であると検知されてしまう. そこで事前実験では Honeypot のコマンドを拡張し, 拡張をしていない Honeypot とコマンドの拡張をした Honeypot で侵入ログを収集した. 収集したログを確率的な算出方法を使用することで比較した結果, より多くの侵入者のコマンド実行ログのパターンを取得できることを示した. 本研究ではコマンドを拡張した Honeypot の侵入ログがどれほど実際の OS に不正な SSH の侵入をされた際の侵入ログの近似を試みた．評価として，拡張をしていない Honeypot とコマンドの拡張をした Honeypot と, さらに実際の OS を使用した Honeypot で侵入ログを収集し, 比較を行なった. この 3 つの侵入ログを自然言語処理の意味解析を用い, コマンドの一つ一つの意味をベクトル表現することで, 拡張した Honeypot で収集した侵入ログが, 拡張をしていない Honeypot で収集した侵入ログよりも, 実際の OS を使用した Honeypot で侵入ログに意味的に近くなることが明らかとなった.

キーワード:
1. SSH, 2. Honeypot, 3. 機械学習, 4. 自然言語処理

慶應義塾大学 総合政策学部
菅藤 佑太

Abstract of Bachelor's Thesis - Academic Year 2018

Approximation of high-interaction honeypots by Command Extension of low-interaction honeypot

I was able to make high processing system, and a moving apparatus is wide, and the OS's such as the Linux system came to be in this way used for people on a device, and, meanwhile, one's apparatus is stepped over by an unjust invasion of SSH, and I am attacked to the network apparatus of the third party from an apparatus of oneself by the spread of PCs and the system advancement of the IoT device, and a problem made in the environment that a virus and a back door are installed, and is carried out a virus an apparatus of oneself illegally occurs. Let an intruder detect it by making the environment that there is ,Honeypot as a window what kind of ways an intruder behaves when was invaded, and is easy to invade it daringly in SSH when succeeded in a login trial, and Honeypot which collect the log of the command that carried out on this occasion, and emulated behavior of Shell now again is wide, and is used, and there are few commands that this Honeypot can carry out; as a result of being implemented, and therefore being detected by an intruder to Honeypot when invasion ahead was Honeypot, and expanding the command of Honeypot by the there prior experiment, and having compared the log that collected invasion log in Honeypot which expanded Honeypot and the command that did not expand, and collected by using a probabilistic calculation method, with Honeypot which expanded Honeypot and the command that did not expand, collected invasion log more in Honeypot using the real OS, and invasion log of Honeypot which showed that could acquire a pattern of the command practice log of more intruders, and expanded the command in this study used semantic analysis of the natural language processing for the real OS in the invasion log of these three how to show approximation したのかをを to invasion log when was invaded of unjust SSH, and evaluated a semantically thing nearby in invasion log than the invasion log that collected in Honeypot which the invasion log that collected in Honeypot which expanded because a vector expressed one one meaning of the command did not expand in Honeypot using the real OS.

Keywords :
1. SSH, 2. Honeypot, 3. Machine Learning, 4. Natural Language Processing

Keio University, Faculty of Policy Management Studies
Yuta Sugafuji

目 次

図 目 次

表 目 次

第1章　序論

本章では本研究の背景，課題及び手法を提示し，本研究の概要を示す．

1.1　本研究の背景

1.1.1　通信機器の普及

PC の普及や IoT デバイスのシステム高度化により，高度な処理系を組むことが可能になった．これによりデバイス上に Linux 系などの OS が搭載された機器が広く人々に使われるようになった．また，Linux 系 OS にリモートログインする手法として SSH がある．これを用いて不正に侵入する攻撃が行われている．

1.2　honeypot

侵入された際に侵入者がどのような挙動をしているのかを知る手段として，Honeypot がある．これは実際の OS を用いたり，Shell の擬似的な挙動をアプリケーション上で実現し，敢えて SSH で侵入しやすいような環境を作ることで，侵入者にログイン試行に成功したと検知させ，その際に実行したコマンドのログを収集する．

1.3　本研究の問題と仮説

SSH の Honeypot は大きく二種類に分けることができ，一つは低対話型 Honeypot，もう一つは高対話型 Honeypot である．低対話型 Honeypot は実際の Shell の挙動をエミュレートしたアプリケーションである．
高対話型 Honeypot は実際の機器を設置し，その中に侵入させログを収集する．その設置時には他のホストに攻撃できないようにネットワークの設定や，root の権限が取られないように user 権限の設定を適切に行う．高対話型 Honeypot は低対話型 Honeypot と比較すると，Honeypot への侵入者が実行できるコマンドが多く，挙動も本物の OS と差異が極めて少なく，侵入先が Honeypot であると極めて検知しされにくい．そのため，高精度な攻撃ログを取得することができる．しかし，Honeypot として適切な設定を行なった OS が，OS 自体の新たな脆弱性を突かれることで，踏み台にされ他のホストに攻撃をしたりウイルスに犯されてしまうなどの危険を孕んでいる．そのため，設置コストが高く普及率

も非常に低い．[3]
一方で低対話型 Honeypot はアプリケーションであるため，root 権限を取られるような危険が極めて少なく，アプリケーション内での脆弱性に限った問題しか存在しない．そのため設置コストが低く，比較的誰でも安全に設置できるため，普及率が高い．しかし，あくまでエミュレーションを行なったアプリケーションであるため，実際の Shell とは異なる挙動や，Honeypot に特有な挙動をしてしまうことがある．そのため，設置した Honeypot に侵入した悪意のあるユーザーに侵入先が Honeypot であると検知されてしまう可能性がある．
本研究では低対話型 Honeypot に着目する．低対話型で実際の攻撃ログに近いログを収集するには，先述の Honeypot であることの検知を回避する必要がある．そこで本研究では，低対話型に実装されているコマンドの出力を，実際の Shell に近似することで検知を回避できるのではないかと考えた．

1.4 提案手法の実装

先述の Honeypot であることの検知を回避するために，本研究では低対話型 Honeypot を実際の Shell の挙動に近似するために，2 つの実験を行なった．一つは実際の Shell に実装されているが低対話型 Honeypot に実装されていないコマンドの実装した．もう一つは低対話型 Honeypot に特有の異常な挙動を修正を行った．

1.5 予備実験

本研究の予備実験として，SSH の低対話型 Honeypot に実装されていないコマンドで，悪意のある侵入者が使うコマンドを実装することで拡張を行なった低対話型 Honeypot と，素の低対話型 Honeypot でそれぞれ収集したコマンドログの比較を行なった．追加実装を施した SSH の低対話型 Honeypot の方がコマンドパターンとして多く収集できることを示した．

1.6 本研究の評価

提案手法の実装で拡張した低対話型 Honeypot と，素の Honeypot と，高対話型 Honeypot を設置し，それぞれ侵入者が実行したコマンドのログを収集し，比較を行った．収集したコマンドのログはコマンド 1 つ 1 つごとに自然言語処理の手法を用いて意味解析をし，コマンドの意味をベクトル空間上に表現した．本研究では，高対話型のログに近似することで，高度なログが収集できていると考えた．そこで，拡張した低対話型 Honeypot の侵入ログが素の Honeypot と比較して，高対話型 Honeypot の侵入ログにどれほど次元空間上で近似したのかを評価した．

1.6.1 予備実験

本研究の予備実験として，SSH の低対話型 Honeypot に実装されていないコマンドで，悪意のある侵入者が使うコマンドを実装することで拡張を行なった低対話型 Honeypot と，素の低対話型 Honeypot でそれぞれ収集したコマンドログの比較を行なった．追加実装を施した SSH の低対話型 Honeypot の方がコマンドパターンとして多く収集できることを示した．

1.7 本論文の構成

本論文における以降の構成は次の通りである．

2 章では，本研究の要素技術となる Shell と Honeypot と自然言語処理について整理する． 3 章では，本研究における問題の定義と，解決するための要件，仮説について説明する． 4 章では，本研究にあたっての事前実験の概要と結果を述べる． 5 章では，本提案手法について解説する． 6 章では，Honeypot の拡張についての実装方法や実装例について述べる． 7 章では，求められた課題に対しての評価を行い，考察する． 8 章では，関連研究を紹介し，本研究との比較を行う． 9 章では，本研究のまとめと今後の課題，展望についてまとめる．

第2章　本研究の要素技術

本章では，本研究の要素技術となる Shell と Honeypot と時系列データの扱いについて各々整理する．

2.1　Honeypot

使われているデバイスへの不正な SSH によって侵入された際に，侵入者のログを収集する手段としての Honeypot がある．SSH の Honeypot[4] は低対話型 Honeypot と高対話型 Honeypot の大きく二種類に分けることができる．

2.1.1　低対話型 Honeypot

SSH の低対話型 Honeypot は実際の Shell の挙動をエミュレートしたアプリケーションである．実際の Shell の挙動をエミュレートしただけのアプリケーションなので，脆弱性がアプリケーション内に限られる．そのため，root 権限を侵入者に許してしまい，踏み台にされてしまうなどの危険が極めて少ない．しかし，エミュレーションには限界があるため，コマンドやその挙動について，実際の Shell とは異なる挙動をすることがある．そのため，侵入者に侵入先が Honeypot であると検知されてしまう．検知されることで，攻撃者は実際の攻撃を行わず，本来取れるはずの攻撃ログが収集できない可能性を含んでいる．そのため，収集ログの精度に問題がある．

2.1.1.1　Kippo

Kippo は，悪意のある SSH のログイン試行者や侵入者の挙動やログを記録するために使用される Python で実装された SSH の低対話型 Honeypot である [?]．Kippo は前身の Kojoney[5] に大きく影響を受けている．ネットワークは Twisted[6] というフレームワークで組まれている．Kippo のプロジェクトは低対話型 Honeypot として 2009 年に登場し，Raspberry Pi[7] などを筆頭としたシングルボードコンピュータ [8] の普及と相まって広く設置された．Kippo の機能の特徴としては収集したコマンドログ を時系列データとして保存されており，”playlog”という Kippo 内にあるプログラムを実行することで，過去のコマンドログ を実際にタイピングしてるかのように出力できる．また，侵入者によってダウンロードされたファイルも実行ができないように保存できる．Kippo は Cowrie の登