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 が動く機器が広く人々に使われるようになった. そうした中で SSH の不正な侵入によって自分の機器が踏み台にされ, 自身の機器から第三者のネットワーク機器に攻撃されていたり, ウイルスやバックドアが設置されて自身の機器が不正にウイルスが実行されるような環境を作られる問題が発生している. 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. 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 Honeypot のコマンドを拡張することで, 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, 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に近似したのかを自然言語処理を用いて評価した.

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; showed that could acquire a pattern of the command practice log of more intruders by being implemented, and therefore being detected by an intruder to Honeypot when invasion ahead was Honeypot, and expanding the command of Honeypot by a there prior experiment, and evaluated it using natural language processing whether invasion log of Honeypot which expanded the command in this study was similar to invasion log when was invaded of SSH which was how unjust for 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

第 1 章　序論 **1**
1.1　研究の背景 1
1.2　本論文の構成 2

第 2 章　本研究の要素技術 **3**
2.1　Honeypot 3
2.1.1　低対話型 Honeypot 3
2.1.2　製造責任と知的財産権に関する法制度 4
2.1.3　高対話型 Honeypot 4
2.1.4　SSH の Honeypot の比較 4
2.1.5　Shell 5
2.1.6　自然言語処理 6

第 3 章　本研究における問題定義と仮説 **7**
3.1　本研究における問題定義 7
3.1.1　SSH Honeypot の現状の問題 7
3.1.2　本研究の問題 9
3.2　問題解決のための要点 9
3.3　仮説 9

第 4 章　事前実験 **10**
4.1　概要 10
4.2　要素技術 10
4.3　問題定義 10
4.4　予備実験の手法 11
4.5　実装 11
4.6　評価 11
4.7　結果 11

第 5 章　本研究の手法 **14**
5.1　問題解決の為のアプローチ 14
5.1.1　コマンドの追加実装 14
5.1.2　既実装コマンドの修正 14

第 6 章　実装 **15**
6.1　実装環境 15
6.1.1　純正の Honeypot で未実装のコマンドの実装 15

第 7 章　評価および考察 **16**
7.1　評価手法 16
7.1.1　評価手法の実装 18

第 8 章　第 8 章 **21**
8.1　関連研究 21
8.1.1　SSH の Honeypot 21
8.1.2　時系列データの処理 21

第 9 章　結論 **22**
9.1　本研究のまとめ 22
9.2　本研究の課題と展望 22
9.2.1　うんちっち〜 22
9.2.2　うんちへの応用 22

謝辞 **23**

図 目 次

2.1 SSH の低対話型 Honeypot と SSH の高対話型 Honeypot の比較 5

3.1 不正な SSH 侵入者の想定行動フロー 7

4.1 収集した SSH の低対話型 Honeypot のデータ 12
4.2 純正の Cowrie と修正済みの Cowrie のスコアリングによる比較 13

7.1 予備実験の評価の概念図 17
7.2 本研究の評価の概念図 17
7.3 評価のフロー [1][2] 19
7.4 評価のフロー [1][2] 20

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　研究の背景

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 が動く機器が広く人々に使われるようになった. そうした中で SSH の不正な侵入によって自分の機器が踏み台にされ, 自身の機器から第三者のネットワーク機器に攻撃されていたり, ウイルスやバックドアが設置されて自身の機器が不正にウイルスが実行されるような環境を作られる問題が発生している. 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. これは 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.

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システムである. 高対話型 Honeypot は他のホストに攻撃しないようにネットワークの設定を適切に行い, また root の権限が取られないように user 権限を適切に行ったりするなどの設定を施した実際の OS を用いた仕組みである. 高対話型 Honeypot は低対話型 Honeypot と比較すると, 本物の OS を用いている分高精度な攻撃ログを取得することができるが, 乗っ取ら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ため設置コストが高く, 普及率も非常に低い. 一方で低対話型 Honeypot はアプリケーションシステムであるため,root 権限を取られるような危険が極めて少なく, アプリケーションシステム内での脆弱性だけ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, ドキュメントが多く存在し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, 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.

この低対話型 Honeypot の設置コストの低さで, かつ Honeypot に侵入してきた悪意のあるユーザーに侵入先が Honeypot であると検知されないように攻撃ログを取得するためには, 低対話型 Honeypot の挙動をできるだけ実際の Shell に近似すれば良いのではないかと考えられる.

本研究の予備実験では,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ようなコマンドを実装し, 何の追加実装も施していない SSH の低対話型 Honeypot で取れた侵入者の実行コマンドログと , 追加実装を施した SSH の低対話型 Honeypot の侵入者の実行コマンドログを比較することで, 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.

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, 実際の Shell に実装されているもので低対話型 Honeypot に実装されていないコマンドの実装や, 低対話型 Honeypot に特有の異常な挙動を修正を行いこの問題を解決できるのではないかと考えた.

実際の Shell の挙動に近似するように実装した低対話型 Honeypot で取れた攻撃ログが, 何の実装を施していない純正の低対話型 Honeypot で取れた攻撃ログと比較して, 実際の OS で用いた攻撃ログにどれほど近しいかを時系列データを教師データとした機械学習を用いて検証することで評価した.

1.2 本論文の構成

本論文の構成を以下に示す. 第 1 章では, 本研究の背景と目的について述べた. 第 2 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整理する. 第 3 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, 第 4 章では, 本研究の問題の定義をする. 第 5 章では, 本提案手法のシステムについて解説する. 第 6 章では, 実装方法や実装例について述べ, 第 7 章で提案システムの評価手法について説明する. 第 8 章では先行研究や事例について紹介する. 第 9 章では本研究のまとめと展望について言及する.

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で不正な SSH によって侵入された際に侵入者がどのような挙動をしているかのログを収集する手段としての Honeypot について, その技術について解説する.SSH の Honeypot[3] 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システムである. エミュレートされたものであるのでコマンドやその挙動についての機能が限定されており, 実際の Shell の機能として不足があり, 侵入者に侵入先が Honeypot であると検知され, 本来取れるはずの攻撃ログが収集できなくなってしまう可能性を含んでいるため, 収集ログの精度に問題がある. しかし, 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システムなので, 脆弱性がアプリケーションシステム内に限られ,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.

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 であり [?],Kippo は前身の Kojoney[4] に大きく影響を受けている. 実装は Python で行われており, ネットワークは Twisted[5] 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[6] などを筆頭としたシングルボードコンピュータ [7] の普及により 2010 年頃から勃興した IoT デバイスの高度化 [8] 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 をそのまま再生することができる. 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しておくことができる.Kippo は 2014 年頃を最後に現在はプロジェクトが進んでいない.[9] Kippo は IoT デバイスの高度化広く設置された SSH の低対話型 Honeypot のうちの一つであったが, 実装されているコマンドも 17[10] と少なく, また Kippo 特有の異常な挙動があったりと多くの問題があった.

2.1.1.2 Cowrie

Cowrie は Kippo 同様, 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 であり, 実装は Kippo のコマンドや運用における機能の拡張をしたものとなっている. Kippo 特有の異常な挙動を改善しているものの, 実装コマンド数は 38[11] と Kippo より少し多くなっているものの [12],Cowrie 特有の異常な挙動もまだまだ多い.

2.1.2 製造責任と知的財産権に関する法制度

本項では既存の製造責任と知的財産権に関する法制度を概説し，それらのパーソナルファブリケーションの中での問題について整理する.

2.1.3 高対話型 Honeypot

2.1.3.1 Honeynet Project

2.1.4 SSH の Honeypot の比較

以上をまとめた SSH の低対話型 Honeypot と SSH の高対話型 Honeypot の比較を行った表を図 2 に示す.

	設置コスト (リスク)	Honeypotであることの 検知されにくさ
低対話型Honeypot	設置コストが低い	検知されやすい
高対話型Honeypot	設置コストが高い	検知されにくい

図 2.1: SSH の低対話型 Honeypot と SSH の高対話型 Honeypot の比較

2.1.5 Shell

Shell は OS のユーザーのためにインタフェースでカーネルのサービスへのアクセスを提供するソフトウェアである. 本研究での”Shell”はコマンドラインシェルのことについてのことを指す.

2.1.5.1 Secure Shell

Secure Shell（セキュアシェル、SSH）は、暗号や認証の技術を利用して、安全にリモートコンピュータと通信するためのプロトコル。パスワードなどの認証部分を含むすべてのネットワーク上の通信が暗号化される.[13]SSH における問題としては, 通信する上での認証方法が鍵認証ではなくパスワード認証になっていてパスワードの総当たり攻撃を受けたり, パスワードが標準のままの設定になっていることで不正なログイン試行によって侵入を許してしまうというものである.

2.1.5.2 BusyBox

BusyBox は標準 UNIX コマンドで重要な多数のプログラムを単一のバイナリファイルに含むプログラムである.BusyBox に含まれる, 多数の標準 UNIX コマンドで必要とするプログラムの実行ファイルは”Linux 上で最小の実行ファイル”となるよう設計されている. 一般にインストールされる実行ファイルは一部だけを実装できるように選択することができ, 一般的には BusyBox の機能は 200 以上も用意されている.[14](今回使用したものに含

まれるコマンドの数は 219) この BusyBox をインストールして実際にこれを実行するためには, 例えば「/hoge/busybox/ xxx」(”xxx”は標準 UNIX コマンドなどが入る) とすれば良い.

2.1.6 自然言語処理

過去のデータの入力に対して未知のデータをどのようにして出力するのかについては様々な手法がある. 本研究において自然言語処理は意味解析についてこれを使用した.

2.1.6.1 統計的意味解析

過去のデータの入力に対して未知のデータを統計的に出力する.

2.1.6.1.1 マルコフ連鎖

2.1.6.1.2 コーパス

2.1.6.1.3 シソーラス

シソーラス [15]

2.1.6.2 ベクトル空間表現

2.1.6.2.1 word2vec