Computer Security Symposium 2016
11 - 13 October 2016

自動プランニングを用いたサイバー攻撃手順の生成

齊藤悠希　八槇 博史

概要：サイバー攻撃が複雑になる中，将来的に人工知能による攻撃手法自動化が行われると考えられる．現在，人工知能は車の自動運転やゲームなどのあらゆる面で活躍しているので，将来的にはすべての物事が人工知能を用いて自動化されると考えられる．それと同様にマルウェアによる攻撃手法も進化していくと考えられる．そこで本研究ではマルウェア自身が社内サーバに侵入した場合を仮定して，目的の攻撃行為を達成するまでの攻撃プランを人工知能の一分野であるプランニングを用いて攻撃経路を自動生成することに試みる．これにより，実際に想定される攻撃経路から自律型マルウェアによる攻撃の危険性について考察することが本研究の目標である．

1. はじめに

サイバー攻撃が複雑になる中，将来的に人工知能によるサイバー攻撃の手法や手段の自動化が行われると考えられる.現在，人工知能は車の自動運転やゲーム，ロボットなどのあらゆる面で活躍している.将来的にはすべての物事が人工知能を用いて自動化されると考えられる.

サイバー攻撃を行う者たちも，今後は，人工知能技術を用いて攻撃パターンを増やしあるいは最適化し，攻撃の規模を広げていくと考えられる.同様にマルウェアによる攻撃手法や感染方法も進化していくであろう．現在のサイバー攻撃に用いられる一般的なマルウェアの感染方法は大まかに分けて二種類あると考えられる.

はじめにソーシャルエンジニアリングを使った手法である．ソーシャルエンジニアリングの手法の一つとして、標的型メール攻撃などがある．標的型メール攻撃の手法としてはまず，攻撃者が攻撃対象の会社について調査を行う．社内の組織図や重要な人物である取締役や部長の名前やメールアドレスや部署などの情報を収集し，それらの情報を元に標的型メールを重要な人物になりすまして送信を行う.その際にメールに「この前の会議の議事録」などとマルウェアが仕込まれている添付ファイルを送りつけ，実行させる．マルウェアが侵入したのちマルウェアは，感染したサーバから C&C(Command and Control)サーバへ通信を行い，外部から指令を受け取りつつ感染を LAN 内部の他の機器へと拡大していく.

二つ目の感染方法として，ソフトウェアなどの脆弱性を突いた攻撃手法がある．現在，ゼロデイなどを売買するブラックマーケットが問題になっている．強力なゼロデイになればなるほど，価格が高く売買されている．これらの強力なゼロデイの中には，サーバソフトウェアなどの脆弱性を突くものがある．この場合，外部から実行されるゼロデイの攻撃自体を防ぐことは難しく，マルウェアの感染拡大が行われる可能性がある．サーバソフトウェアのゼロデイなどを使った場合の手法として，攻撃者はとあるサーバソフトウェアのゼロデイを利用して，サーバに侵入する．侵入したのちにマルウェアの設置を行い実行すると C&C サーバと通信が始まり，外部からの命令を受け取りつつ，マルウェアの感染拡大が行われる．これらを踏まえ，将来的には，C&C サーバにアクセスすることなく，人工知能を搭載したマルウェア単体でサーバに侵入し，人間の攻撃者からの指示を逐一受けずに社内ネットワーク内で侵入したマルウェア同士で C&C サーバとの間で行うのと同様の情報交換を行うようになる可能性が高い．そのような，マルウェアが外部の C&C サーバと通信しない状況では，現在の主要な対応策である通信先の C&C サーバの特定と確保といった対策の適用がより難しくなると考えられる.

本研究では，攻撃者が自律型マルウェアを社内サーバに侵入させた場合を仮定して，目的の攻撃行為を達成するまでの攻撃プランを，人工知能の一分野であるプランニング技法を用いて自動生成することを試みる．これにより，実際に想定される攻撃経路から自律型マルウェアによる攻撃の危険性について考察することが本研究の目標である.

2. ネットワーク探索アルゴリズム

攻撃手順の自動生成の方法として，線形プランナの一種の STRIPS(Stanford Research Institute Problem Solver)[1]を用いて，攻撃経路を自動生成する.STRIPS とは自動計画用の言語で，STRIPS のインスタンスは，初期状態，目標状態，行動群で構成され，行動群には事前条件と事後条件の２つからなる.事前条件とは，リテラルを使用するための条件であり，事後条件とは，リテラルの実行後に追加される条件である.STRIPS は，事前条件と事後条件から再帰的に実行していき，目標を達成していく仕組みであり，自動計画システムは行動の実行順序を生成する.線形仮説には，前向きプランニングと後ろ向きプランニングがある.前向きプランニングは初期状態から目標状態に向かってプランを組み立てていくため，探索空間を広く探索することになり効率がよくない.また，後ろ向きプランニングでは目標状態から初期状態を達成するために目標から副目標に分割を行い，副目標が達成されるまで探索を続け，全ての目標が達成されるとプランニングが終了する.そのため探索時間としては後ろ向きプランニングを適用するものとし，中でも古典的で代表的なものとして STRIPS を採用し，サイバー攻撃手順の生成システムとしての性能を評価した.

別の攻撃手順の自動生成の方法として，グラフ理論におけるアルゴリズムの一種であり，単一始点最短経路問題を解くための最良優先探索であるダイクストラ法を用いて，攻撃経路を自動生成する．ダイクストラ法とは，グラフ上の２頂点間の最短経路を求めるアルゴリズムである．また，最短経路の推定値を事前にわかっている場合では，A*アルゴリズムを用いるとより効率的に最短経路を求めることができるが，ネットワークに侵入した場合を仮定して実験を行うため，推定値がわからないのでダイクストラ法を選択した．ダイクストラ法では，スタート地点からノード n を通って目的地点までの最短距離値を求めるためにスタート地点からノード n までのコストである実コストとノード n から次のノードであるノード m へ移動するときのコストをステップ値として，最短距離値を求めていく．また，ダイクストラ法は前向きプランニングである.

本研究では，前向きプランニングである，ダイクストラ法と後ろ向きプランニングである，STRIPS のアルゴリズムをそれぞれ試した.

3. STRIPS を用いた攻撃手順の自動生成

3.1 STRIPS を用いたアルゴリズム

実際のネットワーク構成を STRIPS を用いてプランニングを行うアルゴリズムを構築した.

1. 目標状態と初期状態の差からまだ達成出来ていない目標を算出する．もし差の結果が空であれば，プランニングを終了する.
2. まだ，達成出来ていない目標を一つ選択し，副目標とする.
3. 選択した副目標を前提条件と事後条件を元に再帰的に実行をしていき，副目標を達成していく.
4. 副目標を達成できなかった場合,別の目標を選択し，副目標とし,実行する.達成できる目標がなければ，プランニングを終了する.
5. 目標状態がすべて達成されるとプランニングが終了する.

１から５の動作から実際に計画のプランニングを行う.図 1 のネットワークマップが社内のネットワークと仮定する.まず,攻撃者は目標を設定する.初期状態はグローバルネットワーク内の攻撃者自身になり,目標状態は管理サーバへのバックドアの設置とする.探索アルゴリズムのプログラムにはネットワークを階層ごとにわけ，ネットワークレベルという概念を用いている．図 1 を例に取ると，管理サーバをネットワークレベル 3 とし，業務サーバをネットワークレベル２, web サーバとメールサーバがネットワークレベルを１で最後に攻撃者自身はネットワークレベルを 0 として，探索を行っており，ネットワークレベルから目標が正確に目的の階層まで達成されるのかを判断するアルゴリズムになっている.また，この探索アルゴリズムのプログラム内にある，行動群には，Attack メソッドと Infected メソッドがある．Attack メソッドは，サーバに攻撃を行うメソッドである．Attack メソッドの事前条件は達成出来ていない目標のリテラルとネットワーク階層のリテラルである.事後条件は，ネットワークの階層を一つ下げることである.Infected メソッドは,マルウェアを仕掛けるリテラルであり，事前条件は，Attack メソッドと同様で，達成出来ていない目標のリテラルとネットワーク階層のリテラルである.事後条件も同様で，ネットワーク階層を一つ下げることである.

これらの二つの条件から，プランニングを行う.STRIPS は後ろ向きプランニングのため，管理サーバから探索を行っていく．まず始めに管理サーバから業務サーバの探索を行う．現在のネットワーク階層の次の階層にある IP アドレスを探す．業務サーバの IP アドレスが見つかると次に攻撃を行う.攻撃が成功するとサーバに侵入したと仮定して，バックドアを仕掛ける代わりに IP アドレスを登録し，登録した IP から探索できる範囲であるネットワーク階層の一つ下の IP アドレスに対して探索を行う．次の攻撃対象として，業務サーバからメールサーバもしくは web サーバに対して，攻撃を行う.攻撃が通った方，または両方にバックドアを仕掛ける代わりに IP アドレスを登録し，ネットワーク階層を一つ下げる．最後にグローバルネットワーク内の攻撃者に対して攻撃を行い，攻撃が通ると探索は終了する.後ろ向きプランニングの利点は，図２のネットワーク構成であった場合，後ろ向きプランニングを行うと 192.168.10.4 の web サーバを通ることはないが,前向き探索の場合,192.168.10.6 の web サーバやメールサーバ,192.168.10.4 の web サーバのすべてを探索しなければわからないので探索速度という面では，後ろ向き探索の方が探索効率の良いことがわかる.

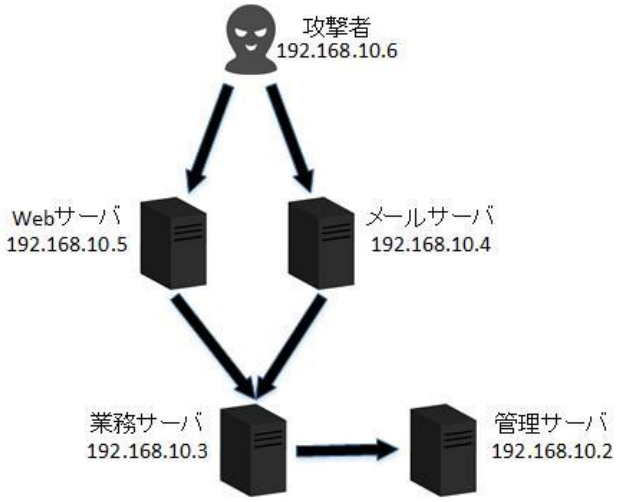

図 1:ケース①のネットワーク構成図

3.2 探索アルゴリズムの実行事例のケース①

図 1 からグローバルネットワーク上の攻撃者を 192.168.10.6 としSTRIPS を用いた計画自動生成アルゴリズムを実行する.まず，初期状態は攻撃者 192.168.10.6 とし,目標状態は，バックドアを管理サーバ 192.168.10.2 に仕掛けることである.実行すると下記になる.

'Hacker':'192.168.10.6'
'Backdoor':'192.168.10.2','192.168.10.3',
'192.168.10.4','192.168.10.5',' 192.168.10.6'

上記の実行結果から,攻撃者が 192.168.10.2 の管理サーバにバックドアを仕掛けている状態から探索が始まり，192.168.10.6 の攻撃者自身の位置までのプランニングを行う.Backdoor という項目では,管理サーバ192.168.10.2から始まり，業務サーバ 192.168.10.3 に対して，Attack メソッドによって攻撃を行い，その後，Infected メソッドによってバックドアを仕掛けた．次に Backdoor という項目にメールサーバ 192.168.10.4 があるので，メールサーバ 192.168.10.4 も同様にAttackメソッドによって攻撃を行い，Infected メソッドによって，バックドアを仕掛けた．次に Backdoor という項目では，web サーバ 192.168.10.5 も同様に Attack メソッドによって攻撃を行い,Infected メソッドによって，バックドアを仕掛けた．ここで，図１では，業務サーバ 192.168.10.3 の下の階層には，web サーバ 192.168.10.5とメールサーバの192.168.10.4の二つがある.この場合では，ランダムで web サーバに対して先に攻撃を行うのかまたはメールサーバに対して先に攻撃が行われるのかが決まる．ここでは，まず先にメールサーバに対して攻撃が行われた．また，ここでメールサーバに対して攻撃が成功しなかった場合，必然的に web サーバに対して攻撃を行い，攻撃が成功すれば，バックドアを仕掛けることができるが，攻撃が失敗すれば，ここで探索が終了となる.次に Backdoor という項目で web サーバ 192.168.10.5 に対して攻撃が行われ，バックドアを仕掛ける．最後に攻撃者自身である 192.168.10.6 に対して攻撃が行われ，バックドアを仕掛ける．ここで Hacker という項目では、攻撃者の現在のネットワーク階層でバックドアを仕掛けている現在位置を表している.なので，攻撃者は初期状態である攻撃者自

身の位置である 192.168.10.6 にいるので,初期状態から目標状態までの探索を達成できたことがわかる.

結果として，Backdoor という項目を逆から読むことで初期状態から目標状態までの経路がわかる．よって，192,168.10.6, 192.168.10.5, 192.168.10.4, 192.168.10.3, 192.168.10.2 の順番で探索すれば良いことがわかる.

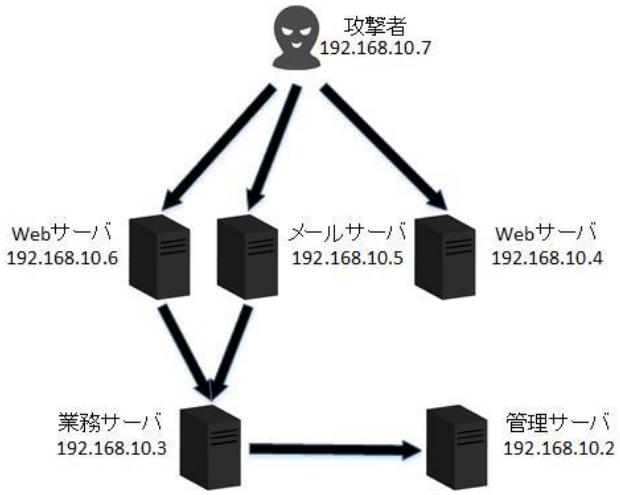

図 2:ケース②のネットワーク構成図

3.3 探索アルゴリズムの実行事例のケース②

別の実行事例として，図 2 のようなネットワークを構成が合ったと仮定して探索を行う.初期状態をグローバルネットワーク上の攻撃者 192.168.10.7 とし，目標状態はバックドアを管理サーバ 192.168.10.2 に仕掛けた状態とする.また，web サーバ 192.168.10.4. は社内ネットワークに繋がらないようにし，攻撃が通らないようにした．アルゴリズムを実行すると下記のようになる.

'Hacker'：'192.168.10.7'
'Backdoor':'192.168.10.2','192.168.10.3',
'192.168.10.5', '192.168.10.6','192.168.10.7'

上記の実行結果からSTRIPSは後ろ向きプランニングなので管理サーバ192.168.10.2から探索を行い，業務サーバ192.168.10.3に対して，Attackメソッドによって攻撃を行い，次にInfectedメソッドによってバックドアを仕掛けた.次に下の階層の業務サーバ192.168.10.3からwebサーバ192.168.10.6とメールサーバ192.168.10.5に対して，攻撃を行う．ここでwebサーバ192.168.10.4の階層はwebサーバ192.168.10.6，メールサーバ192.168.10.5と同じだが，業務サーバからは見えない位置にあるので，探索は行わない.しかし，STRIPSは後ろ向きプランニングなため，探索を行わないが前向きプランニングの場合では，webサーバ192.168.10.4に対して，探索を行ってしまうので非効率である．このような点で後ろ向き探索は有効である．次にBackdoorという項目からメールサーバ192.168.10.5に対して攻撃を行い，バックドアを仕掛けた．次にwebサーバ192.168.10.6に対しても同様に攻撃を行い，攻撃が成功したので，バックドアを仕掛けた．最後にBackdoorの項目から初期状態である攻撃者自身の192.168.10.7に対して攻撃が行われ，バックドアを仕掛けた．これにより，Hackerという項目では初期状態の攻撃者自身のIPアドレスである192.168.10.7になっているので，初期状態から目標状態までの探索を達成できた.ここで，Backdoor という項目を逆から読むことで初期状態から目標状態までの探索経路がわかるので，192.168.10.7，192.168.10.6，192.168.10.5，192.168.10.3，192.168.10.2の順番で探索を行えば良いことがわかる.

4. STRIPS を用いた探索アルゴリズム結果

実際にネットワーク構成図を元に STRIPS を用いて探索することができた．ケース①では，攻撃者自身の IP アドレスである 192.168.10.7 から管理サーバ 192.168.10.2 までの攻撃経路を生成することができた．次に図 2 では，攻撃者人の IP アドレスである 192.168.10.7 から管理サーバ 192.168.10.2 までの攻撃経路の生成することができた．これにより，目的には関係がないサーバに対しての必要のない攻撃の削減と攻撃経路生成の容易さがわかる.
図 1 と図 2 の探索結果から STRIPS を用いた攻撃経路の自動計画生成を行うことができることがわかった.STRIPS の利点としては，後ろ向きプランニングを行っているため，前向きプランニングに比べて探索速度が早いことだが，仮定したネットワーク構成図はシンプルなものであったため，探索速度面での有用性を評価できているとはまだいえない.実際の大きな社内ネットワークを想定した場合にはより STRIPS としての効力が発揮されると考えられる.また，STRIPS の欠点として副目標間に矛盾があるといった複雑なプランニングに弱いという点がある.
実験では，ネットワークマップが分かっている状態での探索をおこなったが，実際のサイバー攻撃では標的のシステム構成が未知である状態からのプランニングが必要であり，プランニングと動作を並行して行う動的プランニングの適用などが必要であると考える.

5. STRIPS に対する考察

本研究で行った STRIPS を用いた攻撃の計画自動生成ではネットワーク構成を仮定した場合の探索であったが，今後の目標として，探索アルゴリズムを用いた攻撃経路の計画自動生成にペネトレーションテストで使われる Metasploit などを組み合わせ，実際のネットワーク上で動作させることを考えている．また，ネットワーク上で動作させるだけでなく，攻撃の自動化を行い，より実際に近いネットワークとマルウェアの感染拡大を再現することで人工知能による攻撃の危険性と攻撃の自動化によるサイバー攻撃への具体的な対策といったことが可能になると考える．そこで，ダイクストラ法を用いることでより，実際のマルウェアに近い攻撃経路の計画自動生成ができると考える．また，非線形プランニングの一つの即応プランニングを用いて，ネットワークマップが分からない状態での攻撃手順の計画自動生成を行うことも考えている．その他にシステムに対して有効な攻撃技術の推定とそれへの対策の策定といった応用も考えている.

現在，実際に人工知能を使った攻撃が行われている可能性がある.また，ある程度の攻撃は自動化していると考えられる．このことから，本研究での攻撃の計画自動生成の重要性は高いといえる.また，将来的にも沢山の分野で人工知能の分野が応用されていくと考えられるので，より巧妙でかつ複雑な攻撃などが増えていくということを踏まえて，攻撃の防御対策を考えていかなくてはならない.

6. ダイクストラ法を用いた攻撃手順の生成

6.1 ダイクストラ法を用いたアルゴリズム

実際のネットワーク構成をダイクストラ法を用いてプランニングを行うアルゴリズムを構築した.

1. スタートサーバ S とゴールサーバ G を設定し、スタートサーバを優先度つきのキューの OPEN リストに格納する.
2. OPEN リストが空であるまたはサーバ n に G が含まれているのであれば，探索を終了する.
3. 隣接しているサーバを m とし、全体のコストである$f(n)$を計算する．$f(n) = g(n) + step(n,m)$ $step(n,m)$はサーバ n から m へ移動するときのコストである.
4. 計算後 OPEN リストにサーバ m を追加し、OPEN リストにあるサーバで最もコストが小さいものを選択する.
5. 探索終了後、G から S まで順次をたどっていくとスタート S からゴール G までの最短経路が得られる.

1 から 5 の動作から実際に攻撃経路のプランニングを行う. 図 3 のネットワークマップが社内のネットワークと仮定する. まず，攻撃者はスタートサーバ S とゴールサーバ G を設定する．スタートサーバを優先度つきのキューの OPEN リストに格納する．次に OPEN リストが空であるすなわち，攻撃する先がない状態である．または現在の攻撃者の位置がゴール G になっている場合は探索を終了する．次に隣接しているサーバを m とし，現在の位置であるサーバ n からサーバ m までのコストを$f(n)$でコスト計算を行う. $f(n)$は，全体のコストを指す．$g(n)$は，実コストとし，表 1 から php は17,731件のエクスプロイトがあり，もっとも件数が多いのでコストがもっとも低いとし，platform の値を0とする．次に windows の8005件が多く platform の値を *1* とし，次に linux の2391件が多かったので platform の値を *2* とする．同様に multiple, asp, hardware, cgi, unix, osx, linx_x86 の順にコストが 1 ずつ高くなっていく．ポートでは，一番件数が多いのは, ポート80の963件が最もコストが低く port の値を *0* とし，次にポート21の145件が多いので，port の値を 1 とする．次に8080は，三番目に件数が多く，89件だったため，port の値を 2 とする．同様に 443，143，25，69，8000，22，445 の順にコストが *1* ずつ高くなっていく．また，platform や port にそれぞれに値が二つ以上存在していた場合は，platform では最も件数が高い php を10とし，platform の元々の値から 2 つ目以降を引く. 例えば, 一つ目が php で二つ目が windows であれば，$0 - (10 - 1)$となり，コストは−9となる．Port も同様に計算を行い，コストを計算する．サーバ n の platform と port のそれぞれの計算後に$g(n) = platform + port$で計算を行い，実コストを求める．また，表 1 に存在しない port や platform が合った場合は，コストを11として計算を行う．次に，$Step(n,m)$のサーバ n とは，現在のサーバの位置であり，サーバ m とは，隣接しているサーバのことである．$Step(n,m)$とは，サーバ n からサーバ m まで移動したコストをステップ値として，+1 をする．しかし，スタートサーバ S から隣接しているサーバへのステップ値は 0 とする.

計算終了後にサーバ m を OPEN リストに追加し, OPEN リストの中でもっともコストが低いものから選択し，攻撃を行う．攻撃が失敗すれば，OPEN リストから攻撃が失敗したサーバを削除し，同様に全体のコストが最も低いものを選択し，探索を行う．また，全体コストが一緒であった場合，どちらか一方をランダムで選択する．最後に探索終了後に目標を達成できていれば，ゴールサーバ G からスタートサーバ S までを順に追っていくとスタートサーバ S からゴールサーバ G までの最短経路を求めることができる.

表 1 : エクスプロイトの件数

Top 10 Platforms		Top 10 Ports	
Platform	件数	Port	件数
php	17,731	80	963
windows	8,005	21	145
linux	2,391	8080	89
multiple	2,015	443	49
asp	1,509	143	47
hardware	1,124	25	35
cgi	692	69	22
unix	305	8000	21
osx	291	22	18
lin_x86	231	445	16

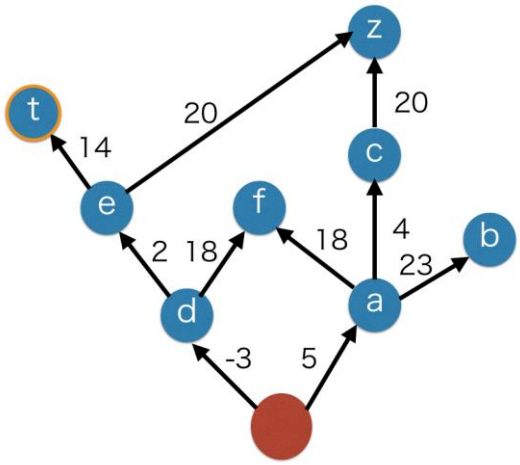

図 3 : ネットワーク構成のデータ構造

表 2 ： サーバごとの Platform と Ports

サーバ名	Platform	Ports
a	windows	143(IMAP)
b	None	None
c	hardware	25(SMTP), 443(HTTPS)
d	linux, asp	21(FTP)
e	php	21(FTP)
f	cgi	None
t	None	22(SSH), 443(HTTPS)
z	None	8000(HTTP Proxy)

6.2 ダイクストラ法を用いた探索アルゴリズムの実行事例

図 3 からスタートサーバ S を赤丸とし，ゴールサーバは t

として，ダイクストラ法を用いた探索アルゴリズムで計算を行った．実行結果は下記のようになり，図 4 になった．
'Hacking': {0: ['d'], 1: ['e'], 2: ['a'], 3: ['c'], 4: ['t']}
'Backdoor': {0: ['d'], 1: ['e'], 2: ['a'], 3: ['c'], 4: ['t']}
まず，Hacking 項目では実際に攻撃を行ったサーバを指し，Backdoor 項目では，攻撃が成功し，バックドアを仕掛けたサーバを指す．上記の実験結果では，サーバ a では，表 2 から Platform は windows で Ports は143（IMAP）であるので，$f(n) = g(n) + step(n,m)$の式に当てはめて$f(n)$を求める．Platform では windows は表 1 から二番目である8005件とあるので，低い方から0，1,と数えるので Platform の windows のコストは1となり，同様に Ports の143は表 1 からコストは4になるので Platform と Ports の値を足して5になる．これを$g(n)$とする．次に$Step(n,m)$のステップ値では，スタート地点からステップ値は0として計算を行うので，ここでは0となる．よって，$f(n) = (1+4) + 0$ となり，$f(n)$は5となる．同様の計算方法でサーバ d を計算する．サーバ d では, Platform は linux,asp を使っており, Ports は21になる．Platform が 2 つ以上あるため, 表 1 から最もコストが低いものを選ぶ. ここでは, linux は三番目に件数が多いので 0 から数えてコスト 2 になる．このコストから 2 番目のコストのものを引いていく，ここでは asp のコストが4なので，結果として，$2-(10-4)$となり，-4となる．これが Platform のコストになる. 3 つ以上になった場合でも同様である. また, Ports も 2 つ以上あった場合も同様の計算を行う．サーバ d の Ports は21のみとなるので，コストは1となる．ステップ値はスタートサーバからの計算になるので0となる．最後に式に当てはめて全体のコスト求めると，$f(n) = ((2-(10-4) + 1) + 0$ となり，$f(n)$は-3となる．次にスタートサーバの赤丸からコストが低い方から攻撃を行うので，図 3 から最もコストが低いコスト-3のサーバ d を選択し，上記の結果から攻撃を行い，バックドアを仕掛けた．次に比較を行うのは，サーバ a，サーバ f，サーバ e であり，それぞれ全体のコスト計算を行う．図 3 では全体のコストをすでに計算しているので，ここでは最もコストが低いコスト2のサーバ e を攻撃し，バックドアを仕掛けた．次に探索を行うのは，サーバ t，サーバ z，サーバ a になり，最もコストが低いのはコスト5のサーバ a なのでサーバ a に対して攻撃を行い，バックドアを仕掛けた．次にサーバ t，サーバ z，サーバ f，サーバ c，サーバ b の中で最も低いコストのものに攻撃をするので，ここでは，コスト4のサーバ c が最も低いのでサーバ c に対して攻撃を行い，バックドアを仕掛けた．最後に，サーバ t，サーバ z，で最もコストが低いコスト14のサーバ t に対して攻撃とバックドアを仕掛け, ゴールサーバ G であるサーバ t にたどり着いたので探索を終了する．ここで，図 4 の探索結果から，t，e，d，S が最短経路である．これにより，ネットワーク構成を例とした，ダイクストラ法による，攻撃経路の生成が可能である．

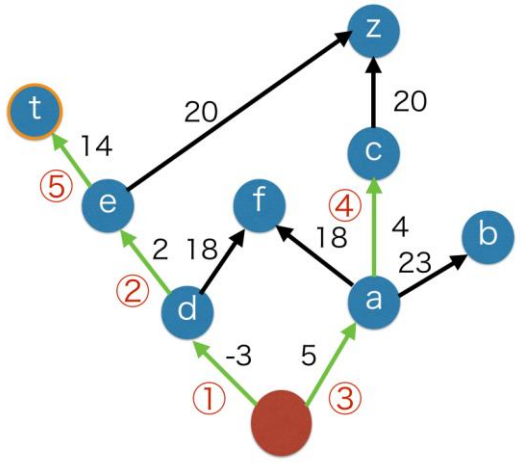

図 4：ダイクストラ法を用いた探索アルゴリズムの結果

7. ダイクストラ法を用いた探索アルゴリズム結果

探索結果から，ダイクストラ法を用いた探索が可能である．よって，実際のネットワーク上でも同様なコスト計算が可能である．しかし，この探索では，攻撃がすべて成功し，バックドアの設置も全て成功したが，実際のネットワークでは，最も低いコストに対して攻撃を行ったが，攻撃が成功しなかった．あるいは，攻撃は成功したがバックドアを設置することができなかったなどの状況も想定する必要がある．また，幅優先探索のため，深い階層に目的のサーバがある場合では，探索に時間がかかってしまう．さらに，ここでは目的のサーバに対しての Platform 情報と Ports の情報を事前に知っていたため探索が可能であったが，実際のネットワークでは，それらの情報をいかに入手するのかが重要になってくると考えられる．

8. ダイクストラ法に対する考察

ダイクストラ法での探索が可能であったが，ダイクストラ法にヒューリスティック関数を用いたアルゴリズムである，A*アルゴリズムでは，事前にゴール地点を知っている状態からヒューリスティック関数を用いてゴール地点の差から計算を行い，最短経路を求めるものになるが，本研究では，実際の攻撃者の観点からネットワークマップがわからない状態での探索を目的としているため，A*アルゴリズムでは，探索を行うことは出来ない．もし，A*アルゴリズムを使う場合は，攻撃者が目的としている社内ネットワークを事前に知っている元従業員という限定的なシナリオになってしまう．しかし，A*はダイクストラ法と比べ，探索速度が早い．ダイクストラ法を用いた探索アルゴリズムにより，最短経路問題として，ネットワークマップでの動作ができるということが分かったので，その他のグラフ理論における問題であれば，解くことが可能と考えられる．

9. STRIPS とダイクストラ法の比較

実際に STRIPS とダイクストラ法の比較を行った. 初期状態を攻撃者 red とし，目標状態は，サーバ t とする．図 3 のネットワークマップを STRIPS を用いて実行した結果が下記になり，図 5 の様になった．red は図 3 の赤丸を指す．

'Hacker': 'red',
'Backdoor': 't', 'e', 'd', 'z', 'c', 'f', 'red'
上記の結果から, Attack メソッドと Infected メソッドにより，まず，STRIPS は後ろ向き探索なので，Backdoor という項目からサーバ t から始まり，サーバ e に対して Attack メソッドによる攻撃を行い，Infected メソッドによってバックドアを仕掛けた．サーバ d も同様に Attack メソッドと Infected メソッドによって攻撃とバックドアが仕掛けた．サーバ z, サーバ c, サーバ f, 攻撃者 red も同様に Attack メソッドと Infected メソッドによって攻撃とバックドアが仕掛けた．最終的にサーバ t, サーバ e, サーバ d, サーバ z, サーバ c, サーバ f, 攻撃者 red の順番で感染拡大を行ったことがわかる．また，Hacker という項目から初期状態である，攻撃者 red にいるので目的を達成したことがわかる．さらに Backdoor の数が攻撃し，バックドアを仕掛けた回数になるので，開始位置であるサーバ t は含まずに数えると，攻撃回数は6回となる．次にダイクストラ法では，図 4 から攻撃して感染した回数は5回となる．よって，攻撃回数的には，ダイクストラ法の方が少ないことがわかる．

考察として，STRIPS では，図 3 のネットワークマップでは後ろ向き探索の利点である目標状態から初期状態に向かって目的と関係ないネットワークへの探索が不要になる点を生かしきれていないことがわかる．また，STRIPS も幅優先探索なため，深さがあるネットワークでは，探索に時間がかかる．しかし，攻撃経路は生成することができた．さらに大きな欠点として，後ろ向き探索のために実際のネットワーク環境では，事前にネットワークマップを知っている必要があるためにより現実的な人工知能を用いた経路探索実験ができないことである．次にダイクストラ法では，攻撃回数は STRIPS より少ないが, STRIPS と同様で幅優先探索なため，深さがあるネットワークでは時間がかかる．また，本研究の目標として，実際のネットワーク上で動作させることで，人工知能を使った攻撃に対する対策や策定を考えることなので，攻撃回数も少なく前向き探索である，ダイクストラ法を選択した．

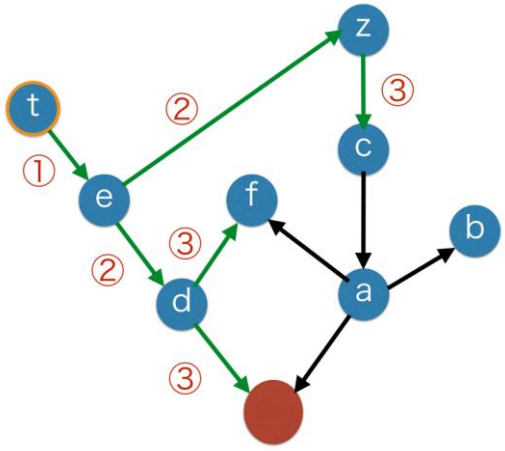

図 5 : STRIPS を用いた探索アルゴリズムの結果

10. 仮想環境内でのダイクストラ法を用いた探索

10.1 ネットワークの環境

ダイクストラ法を用いて仮想環境内のローカルネットワークで動作させ，人工知能による攻撃が可能であるかを試す．実際に攻撃が出来た場合，人工知能による攻撃が可能であるという前提の元に攻撃の対策を考えていくのが本研究の目的である．まず，実際に使用する環境は，Docker である．Docker とは，ソフトウェアコンテナ内にアプリケーションのデプロイメントを自動化するオープンソースソフトウェアである．この，コンテナ技術とは特殊なファイルシステムを利用して，コンテナ型の仮想化を行う．VMware などの完全仮想化を行うパイパーバイザーに比べ，ディスク使用量は少なく，インスタンスの実行速度も早いのが利点である. また, Docker は, Linux カーネルしか動作しない.

本実験では，ホスト側のみに Docker を使用する．ホスト側には Kali Linux を導入する．Kali Linux とは，ペネトレーションテストで使用するツールをまとめた Debian ベースの Linux ディストリビューションになる．Kali Linux にある，ペネトレーションテストで使われる Metasploit Framework を使用した．Metasploit Framework とはエクスプロイトの作成や実行などといったことが可能であり，実際のサイバー攻撃を想定したエクスプロイトによる攻撃を行うことができるため，ペネトレーションテストとして使用することが多い．ローカルネットワーク上の動作では，実際に使用されているマルウェアを使うことは難しいので，攻撃と感染の判定は，Metasploit Framework を使用して行う．また，Metasploit Framework からの攻撃を受ける, 被攻撃サーバは, Metasploitable2 を用いる．Metasploitable2 は，ペネトレーションの練習やテストに使用するためのわざと脆弱性を持たせたサーバである．これらを使用して，仮想環境内にネットワークを構築して，攻撃と感染を行う．また， Metasploit Framework では，自身で攻撃先の IP の指定であったり，使用するエクスプロイトやポートやペイロードなどを自身で設定を行わなくてはならない．しかし，本研究では攻撃を自動化するために Metasploit Framework のプラグインである autopwn を使用して攻撃の自動化を行う．autopwn は，Nmap などを使用して，ポートスキャンを行った結果を Metasploit Framework のデータベースに格納することで，Nmap から収集した IP 情報やポート情報，OS の種類などから自動的にエクスプロイトを選択し，エクスプロイトに必要な設定やペイロードなども自動で設定を行い，当てはまりそうなエクスプロイトを総当たりで自動実行するプラグインである．これを利用して，攻撃の自動化を行う．また，実際に攻撃が成功すると Session が立ちあがる. この Session とは，攻撃が成功した場合に被攻撃サーバとホスト間でのセッションを張り, Meterpreter といわれるコンソールが開き，自由にコマンドやスクリーンショットの撮影などが可能になる．Meterpreter は Metasploit Framework の主要コンポーネントの 1 つであり，脆弱性をエクスプロイトしたあとのペイロードとして利用される．ペイロードとは、エクスプロイトが成功したあとに実行される命令のことである. ここで本実験では, Session が立ち上がったら攻撃が成功したとして感染拡大を再現していく．しかし，サーバを踏み台にして, 感染拡大を行うわけではない. また, Meterpreter を使用して，サーバに対してコマンドなどの実行も行わない．本実験はあくまでもマルウェアによる人工知能を用いた攻撃の自動化で可能であるかの実験なので実際に本格的な攻撃を行う必要がないためである．最後に Metasploit Framework 内でのコマンドの自動実行を行うために-r オプションにより，指定したファイルに書き込んであるコマンドを自動で実行する.

10.2 ダイクストラ法を用いた動作実験

本実験の環境は図 6 となる. OSX のローカルネットワーク内で実験を行う．まず，ホスト側は，Docker を用いて，OS

には Kali Linux を使用する．Kali Linux には Metasploit Framework とポートスキャンで使用する Nmap が入っている．その他に Metasploit Framework で使用するデータベースは，PostgreSQL を使用する．また，ホスト以外のサーバは，それぞれ Metasploitable2 を使用する．Metasploitable2 をハイパーバイザーである， VMware Fusion7 を使用する．Metasploitable2 には事前に下記のポート番号が開いている状態である．

21, 22, 23, 25, 53, 80, 111, 139, 445, 512, 513, 514, 1099, 1524, 2049, 2121, 3306, 5432, 5900, 6000, 6667, 8009, 8180

これらのポートを閉じずに VMware の環境に Metasploitable2 を導入する．また、図 7 のようなネットワークマップ上で動作させる．よって，使用する VMware の環境は５つになる．ここでまず，なぜすべてを Docker で動かすことをしなかったかというと，Docker 同士による Nmap を使用したポートスキャンであったり，Metasploit Framework を使用したエクスプロイト攻撃が出来ず，うまくコンテナ内で動作させることができなかったためにホストのみを Docker とした．また，Nmap や Metasploit Framework や autopwn プラグインの自動実行とダイクストラ法を組み込んだプログラムを作成し，Docker のホストから実行を行う．Docker では，コンテナを起動する際に同時にコンテナ内にコマンド実行を行うことができる．これを利用して，プログラムを実行し，サーバに対しての攻撃を自動実行させる．次に図 7 から 172. 16. 23. 137 を初期状態とし，目標状態を 172. 16. 23. 142 とし，172. 16. 23. 142 にバックドアを仕掛けることができると探索を終了する．また，図 7 の 172. 16. 23. 137 に攻撃と感染が成功すると，172. 16. 23. 140 と 172. 16. 23. 142 にアクセスできるようにプログラムには事前に記入している状態にしてある．172. 16. 23. 143 も同様である．また，攻撃が本研究の目的ではないので，攻撃を突く脆弱性は，１つとし，exploit/multi/http/php_cgi_arg_injection のエクスプロイトのみとする．また，VM 環境内のポートは，開閉せずにすべて同じ Metasploitable2 の環境のままで探索を行った．

実際に図 6 の環境と図 7 のネットワークマップから探索を行った．実行すると下記のような結果になる．

'Hacking': {0: ['172. 16. 23. 137'], 1: ['172. 16. 23. 138'], 2: ['172. 16. 23. 140'], 3: ['172. 16. 23. 142']}}

'Backdoor': {0: ['172. 16. 23. 137'], 1: ['172. 16. 23. 138'], 2: ['172. 16. 23. 140'], 3: ['172. 16. 23. 142']}

上記の結果から目的である，172. 16. 23. 142 に攻撃を行い，バックドアを設置することができた．また，攻撃の探索の順番として，172. 16. 23. 137 から攻撃を行い，バックドアを仕掛けることができたので，次に 172. 16. 23. 138 に対して，攻撃とバックドアを仕掛け，同様に 172. 16. 23. 140, 172. 16. 23. 142 の順に攻撃とバックドアが仕掛けられた．図 7 からダイクストラ法はコストが低い方から探索を行うが図 7 では， 172. 16. 23. 137 と 172. 16. 23. 138 ではコストの値が同じなためにどちらを選択しても問題ないためにランダムで選択を行う．ここでは，172. 16. 23. 137 を選択し，攻撃を行い，バックドアを仕掛けた．次に低いコストである，172. 16. 23. 138 に攻撃を行い，バックドアを仕掛けた．次にコストが低いのは，172. 16. 23. 140, 172. 16. 23. 142, 172. 16. 23. 143 の三つである．これらすべてコストが同じなので，コストが同じ同士でランダムに選択をした．ここでは，まず、172. 16. 23. 140 に対して攻撃を行いバックドアを仕掛けた．次に 172. 16. 23. 142 に対して攻撃を行いバックドアを仕掛け，目的であった．172. 16. 23. 142 へのバックドアを仕掛け，探索が終了した．これらの結果から，ローカルネットワーク内で探索アルゴリズムを組み合わせた自動攻撃が可能である．

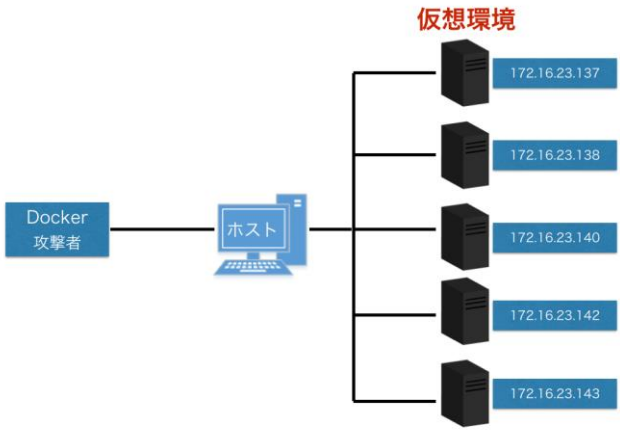

図 6：ネットワークの実験環境

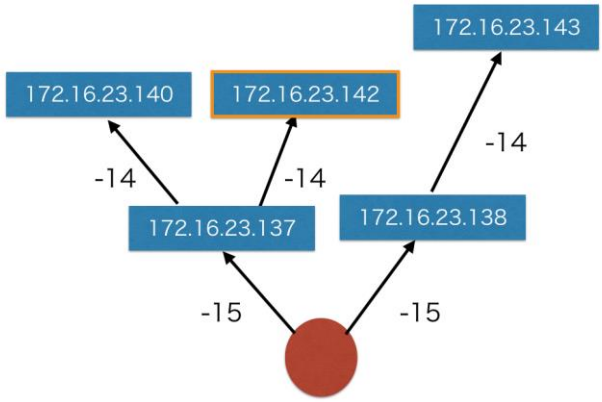

図 7：ネットワークマップ

11. ダイクストラ法を用いたローカルネットワーク内での動作結果

ローカルネットワーク内での動作が可能であることがわかった．このことから，実際に攻撃者が社内ネットワークに侵入したときに自動でダイクストラ法を用いた攻撃経路の探索と脆弱性を突いた攻撃が可能である．しかし，本実験では，シンプルなネットワークマップであり，複数の異なるコストの計算や巨大なネットワーク環境内で実験を行っているわけではないので，まだ，明確に可能であったというのは難しい．例えば，社内ネットワークがとても規模の広いものであれば，同様に探索にかかる時間も大きいと考えられる．図 7 のネットワークで目的を達成するのにかかった時間は，約30分になる．時間がかかる理由は，Metasploit Framework の総当たりによる攻撃に時間がかかる点である．もし，これが現実での攻撃であれば，IDS などの検知システムが攻撃だと判定して，IP をブロックする可能性がある．そのようになれば，攻撃は成立しない．また，攻撃者であれば，総当たりする攻撃を最近発見されたばかりの脆弱性やゼロデイのみでしぼって攻撃することでかな

りの時間短縮になると考えられる．これらのことから，実際に探索アルゴリズムを用いた攻撃経路の探索と攻撃の自動化は可能であり，今後，攻撃手法という観点でこのような攻撃が現実問題行われると考えられる．

12. 攻撃への対策

本研究で使われたダイクストラ法を用いた攻撃の対策として，まず，今回コスト計算を行ったが，コストの計算方法は，より脆弱性が多いものを最もコストが低いとして計算を行ったので，ハニーポットなどを用いて，ダイクストラ法を用いたプログラムを誘い込ませるという手法を用いることで対策が可能だと考えられる．この場合に外部につながらないローカル専用のハニーポットを設置する必要がある．また，それに伴って，攻撃者も脆弱性が多いサーバには侵入しないといった対策を行う可能性がある．このように攻撃者側と防御側はいたちごっこになる．しかし，ハニーポット用いたものなど，ある程度の対策は可能である．人工知能などを用いた攻撃の自動化や探索アルゴリズムを用いた社内ネットワーク探索はとても対策が難しいものであるのは確かである．なぜなら，人間が判断しているのであれば，タイピングミスや攻撃方法のミスと攻撃に時間がかかるなど，攻撃者側のヒューマンエラーなどがヒントとなり，攻撃を防ぐことが可能になることもがあるが，人工知能などを用いた場合の対策は難しく，攻撃も素早く，攻撃自体も最適化されていくと考えられるからである．なので，本研究から考えられる対策は，今後のサイバー攻撃の対策で重要になっていくと考えられる．今後の本研究の目標として，ネットワーク規模を広げることで，現実に近い社内ネットワークを再現し，また，攻撃者の手法を再現することで，サイバー攻撃対策を事前に行うことができると考えられる．さらにこれらを元に企業に導入できる人工知能による攻撃対策専用のセキュリティシステムの開発や社内に必要な注意すべき観点やルールなどの提供がすることで，サイバー攻撃を未然に防ぐことができると考えられる．

13. まとめ

本研究でははじめに後ろ向きプランニングを行う，STRIPS を用いて，攻撃経路の生成を行った．しかし，STRIPS は，後ろ向きプランニングなため，実際に攻撃を行う攻撃者は，ネットワークマップが分からない状態での攻撃を行うため，STRIPS では，実際の攻撃に近い経路探索ができない．そのため，前向きプランニングを行う，グラフ理論における代表的な最良優先探索問題で用いられるダイクストラ法を用いることにしたが，ダイクストラ法にはコスト計算を行う必要があるためにエクスプロイトデータベースから脆弱性が多い順にコストが低いとしてコスト計算を行うことで，ダイクストラ法による探索を実現し，攻撃経路の生成を行った．その結果，攻撃経路の生成が可能であるとわかった．そこで，STRIPS の利点と欠点とダイクストラ法の欠点と利点の比較からダイクストラ法を採用し，実際のローカルなネットワーク環境でダイクストラ法を用いた探索が可能であるか実験を行った．そこで，ローカルネットワーク内で Docker 側には，攻撃を行うために必要な Metasploit Framework と Nmap とそれらにダイクストラ法を用いた自動的に攻撃が行えるようなプログラムを開発した．また，仮想環境内には VMware Fusion7 を用いて，５つの仮想環境を立てて，その中に Metasploitable2 という脆弱性をもった，被攻撃サーバを仮想環境内に導入し，探索と攻撃の自動化を行った．実際に攻撃を行うことができ，探索も可能であった．このことから，どのようにして攻撃対策を行う必要があるのかを考えた．結果として，ダイクストラ法であれば，コスト計算を行うために，脆弱性が多い順番に攻撃を行っているためにハニーポットなど攻撃者をわざと誘い込むサーバを容易することで，対策が可能であると考えられる．しかし，その他にも攻撃方法はあるので，考えられる攻撃手法を再現し，対策する方法を考えていく必要があると考えた．また今後，攻撃手法や攻撃の最適化を人工知能などを用いて行ってくると考えられるので，そのための社内ルールであったり，対策を積極的に考えて行く必要があると考える．

14. おわりに

今後，マルウェアの進化だけでなく，サイバー攻撃手法も変化していくと考えられるので，脆弱性などの対策だけでなく，社内ネットワークや学校内ネットワークなどにマルウェアが侵入した場合にどのように攻撃者が求める情報が含まれるサーバにアクセスさせないかを考える必要がある．これは，セキュリティ関連会社だけではなく，中小企業などでも情報を共有し，徹底した対策やセキュリティ製品や社内ルールの導入など対策できる部分は沢山あるので，積極的に情報の共有とセキュリティ対策は，行っていかなくてはならない．また，セキュリティに詳しくない方へのセキュリティに関する講演や技術の基礎をしっかり身につけさせることで，セキュリティ技術と関心の向上とセキュリティの重要性が広がっていく必要がある．

参考文献

[1]Dijkstra, Edsger W. "A note on two problems in connexion with graphs." Numerische mathematik 1.1 (1959): 269-271.

[2]Hart, Peter E., Nils J. Nilsson, and Bertram Raphael. "A formal basis for the heuristic determination of minimum cost paths." IEEE transactions on Systems Science and Cybernetics 4.2 (1968): 100-107.

[3]“Metasploit“. www.metasploit.com,(参照 2016-07-01)

[4]“Nmap リファレンスガイド”. https://nmap.org/man/jp/,(参照 2016-07-01).

[5]Fikes, R. E., & Nilsson, N. J. (1972). STRIPS: A new approach to the application of theorem proving to problem solving. Artificial intelligence, 2(3), 189-208.

[6]“Docker - Build, Ship, and Run Any App, Anywhere”. https://www.docker.com/, (参照 2016-07-01).