Moving Target Defense

Bettina Noglik
Betreuer: Schlamp Johann
Seminar Innovative Internettechnologien und Mobilkommunikation
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: noglik@in.tum.de

KURZFASSUNG
Das Paper gibt eine breit gefächerte Einführung in die Moving Target Defense (MTD). Sowohl die theoretischen Ansätze als auch die Bezüge zu übergeordneten Konzepten werden erklärt und Problemstellungen und Herausforderungen werden dargestellt. Drei Beispiele sollen daraufhin einen Einblick in die Anwendung in der Praxis geben.

Schlüsselworte
Netzwerksicherheit, Computersicherheit, Moving Target Defense

1. EINLEITUNG
Heutzutage haben die meisten Informationssysteme eine eher statische Konfiguration bzw. behalten eine bestimmte Konfiguration über eine relativ lange Zeitdauer. Das ermöglicht beispielsweise Angreifern Systeme in aller Ruhe auszukundschaften und dann passende Angriffe zu starten.

Das ist beispielsweise beim *Heartbleed*-Bug geschehen. Nachdem der Fehler im Programmcode der Open SSL-Bibliothek gefunden und sich die Information in Hackerkreisen verbreitet hatte, wurden im großen Stile vertrauliche Daten von Servern, die diese Bibliothek benutzten, ausgelesen. Das war möglich, weil die TLS-Antwort aus dem Wert „Payload-length" der Anfrage generiert wurde anstatt aus der tatsächlichen Länge der Payload und keine Grenzenprüfung durchgeführt wurde. Da für jeden Benutzer nur begrenzter Speicher allokiert wird, können durch eine überlange Payload darauffolgende Daten, welche meistens zu Datenobjekten anderer User gehören, in die TLS-Antwort gelangen. Im April 2014 wurde der Bug behoben, aber wie könnte eine weiterreichende Lösung zum Schutz vor Buffer-Overread bzw. Overflow aussehen?

MTD ist ein Konzept, welches dynamisch kontrollierbare Veränderungen eines Systems in einem Netzwerk ermöglicht. Wenn Systeme ihre nach außen sichtbaren Konfigurationen oft ändern, haben Angreifer wesentlich mehr Schwierigkeiten, in ein bestimmtes System einzudringen. Ein Cyberangriff funktioniert normalerweise so, dass die Opfersysteme zuerst ausgespäht werden und möglichst viel Information über diese gesammelt wird. Mit diesen Informationen kann dann ein gut auf die jeweiligen Systeme zugeschnittener Angriff gestartet werden. Ändern sich aber gewisse Konfigurationsparameter schon wieder bevor der wirkliche Angriff stattfindet, so greift dieser ins Leere. Das Angriffsfenster hat sich verschoben. MTD will sich den Vorteil der Zeit, die zwischen Ausspähung und Angriff vergeht, zunutze machen. Konfigurationsparameter können beispielsweise IP-Adressen, Namen, Netzwerke usw. sein.

2. MTD – Theorie
Zunächst erkläre ich theoretische Grundlagen, die Zhuang in [1] als Basis definiert.

2.1 Exploration surface
Das *Exploration surface* ist der Raum aller Ressourcen, die ein Angreifer durchsuchen kann, bzw. die an der „Oberfläche" nach außen sichtbar sind. Diese werden bei einer Cyberattacke genau erforscht und aus den gewonnenen Informationen bestimmt der Angreifer das *Attack surface* zu seinen Gunsten. MTD hat nun den Ansatz, das Exploration surface möglichst groß zu machen, sodass es einen großen Aufwand bedeutet, alle Ressourcen auszuspähen und so die Zeit der Angriffsvorbereitung deutlich zu erhöhen. Komponenten der Exploration surface können zum Beispiel IP-Adressen, aktive Software oder Ports sein.

2.2 Attack surface
Das *Attack surface*, von dem die Rede ist, wird auch Angriffsfenster genannt. Der derzeitige Ansatz von MTD ist es, das Attack surface so klein wie möglich zu machen und somit das System zu verhärten und sicherer zu machen. Alle verwundbaren Ressourcen, die einem Angreifer zugänglich sind, befinden sich in diesem Attack surface. Um auf eines der vorangegangen Beispiele einzugehen, könnten das Softwarebugs und –verwundbarkeiten sein.

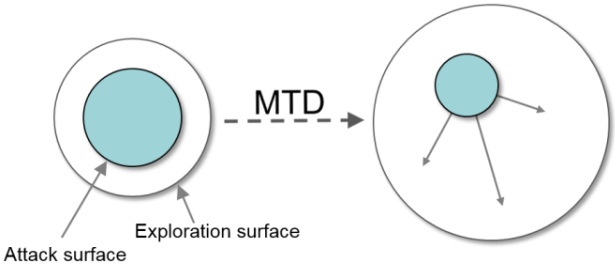

Abbildung 1: MTD Surfaces Konzept

Ziel ist es, den Angreifer möglichst stark zu „verwirren", indem das zu schützende System oft seine Erscheinung im Netz ändert und Angreifer das Ziel am besten aus den Augen verlieren. Das Attack surface wird „verschoben", was bedeutet, dass es einen

doi: 10.2313/NET-2015-03-1_12

Konfigurationswechsel durchmacht. Diese passieren durch Movement oder Transformation (=Adaptionen). Abbildung 1 zeigt links ein herkömmliches System und rechts ein MTD-System mit Bewegung bzw. Adaption des Attack surface innerhalb des Exploration surface.

2.3 Adaptionen und das konfigurierbare System

Die Verschiebung des Attack surface nennen wir Adaption. Diese bestehen entweder aus Bewegung, also Modifizierung einer bestimmten Konfigurationseinheit oder aus Transformation, worunter wir Veränderung der Anzahl der Konfigurations-einheiten verstehen. Adaption bedeutet, den Konfigurations-zustand eines MTD Systems in einen neuen validen Zustand mit veränderten Konfigurationsparametern zu transformieren.

Ein Konfigurationszustand beschreibt spezifische Zuordnungen von konkreten Werten an Konfigurationsparameter. Ein Beispiel wäre Arbeitsspeicher = 4 GB, HDD Größe = 500GB, IP Adresse 192.168.0.54 usw.

Konfigurierbare Systeme bestehen aus einer Menge von Konfigurationszuständen, in denen sich das System befinden kann, einer Menge an Aktionen, die es ausführen kann und einer Übergangsfunktion. Die Übergangsfunktion beschreibt die erlaubten Änderungen der Zustände.

In einem konfigurierbaren System gelten System Policies. Diese sind ein Regelwerk, das zur Prüfung von neuen Zuständen und Adaptionen auf Validität benutzt wird. Valide bedeutet hierbei, dass das System alle spezifizierten Ziele nach der Adaption noch erreichen kann. Diese unterteilt man in sicherheitstechnische und operationale Ziele. Die Sicherheitsziele bestimmen die kritischen Teile des Systems, die beschützt werden müssen, während die operationalen Ziele alle anderen Zwecke, die eine Plattform ausführen soll darstellt.

Ein MTD System besteht laut formaler Definition aus einem konfigurierbaren System, einer Menge an Policies und einer Menge an Zielen. Um einen möglichst hohen Grad an Unvorhersehbarkeit des zu beschützenden Systems zu erzeugen, soll hohe Diversifikation eingesetzt werden. Dies erreicht man durch einen sehr großen Raum an verschiedenen Konfigurationszuständen und durch eine Technik, die die Größe des Konfigurationsraums vergrößern soll. Letztere wird auch künstliche Diversifikation genannt. Als Faustregel gilt also: Je größer der Raum möglichst voneinander verschiedenen Zuständen ist, desto besser ist der Schutz vor Erraten.

2.4 Konfigurationswechsel

Zunächst werden ein neuer Konfigurationszustand aus dem Zustandsraum und eine dazu passende Adaption gewählt. Die Wahl derer kann innerhalb fester oder zufälliger Zeitintervalle erfolgen, wobei zufällige Zeitpunkte das System weniger vorhersehbar machen als feste. Man kann auch Umgebungsinformationen wie Intrusion Detection Alarme als Trigger für den Konfigurationswechsel hinzuziehen.

Sowohl der neue Zustand als auch die gewählte Adaption wird dann anhand der System Policies auf Validität geprüft. Schlägt diese Prüfung fehl, so wird eine neue Adaption bzw. ein neuer Zustand gewählt und die Prüfung wird erneut durchgeführt. Ansonsten wird sie ausgeführt. Abbildung 2 illustriert diesen Ablauf.

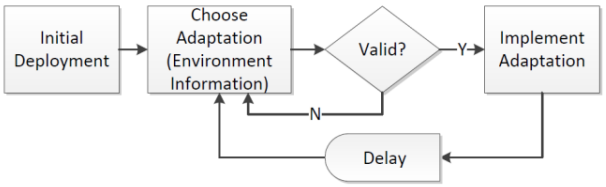

Abbildung 2: General MTD Process [1]

2.5 Randomisierung

Ziel der Randomisierung ist es, vollen Gebrauch der verfügbaren Konfigurationszustände und –parameter zu machen. Die Wahrscheinlichkeit, welcher Zustand als Nächstes gewählt wird, soll entweder gleichverteilt sein oder mit Hinzunahme von Alarmsystemen und zusätzlichen Informationen über potenzielle Angriffe und Verwundbarkeiten ein intelligenteres MTD System formen.

2.6 Probleme

2.6.1 MTD Problem

Eines der Kernprobleme ist es, welcher (valide) Konfigurations-zustand als Nächstes gewählt werden soll. Wie beim Punkt Randomisierung schon erwähnt, kann dies per zufälliger Auswahl erfolgen oder kombiniert mit intelligenteren Methoden wie Angriffserkennungssystemen (z.B. Intrusion Detection Systems) oder kostenbasierten Strategien.

2.6.2 Adaptionsauswahlproblem

Um zum nächsten Zustand zu kommen muss eine passende Adaption, welche aus einer Sequenz an Aktionen besteht gewählt werden. Der Streitpunkt besteht darin, dass es möglicherweise mehrere dieser Aktionssequenzen gibt und man die optimale Lösung finden muss. Hier sollen auch Zeit und Kosten berücksichtigt werden.

2.6.3 Timingproblem

Das richtige Timing, wann eine Adaption durchgeführt werden soll, ist ein besonders wichtiger Faktor für den Erfolg der MTD. Hierbei werden das Intervall zwischen den Adaptionen, die Zeit, die eine Adaption für einen Zustandswechsel benötigt, und die Zeit, die ein Angriff geschätzt benötigt, betrachtet. Die Entscheidung für den Zeitpunkt einer Adaption soll basierend auf dem Tradeoff zwischen Operation und Sicherheit getroffen werden. Je öfter eine Rekonfiguration stattfindet, desto höher sind die Performanceansprüche. Trotzdem sollte dies oft genug passieren, denn nur wenn sich die Konfigurationsparameter genau zwischen der Ausspähung durch den Angreifer und seinem Angriff ändern, kann das MTD System diesem entgehen.

doi: 10.2313/NET-2015-03-1_12

3. Weitere Konzepte

MTD ist, wie schon erwähnt, eine Technik für die Cybersicherheit und gilt als Beispiel für die Theorien der Systemagilität und des Effective Movement. Des Weiteren wird sie in diesem Abschnitt anhand eines spieltheoretischen Entwurfs modelliert.

3.1 Cybersicherheit und Diversität

Wir betrachten hier das Modell eines asynchronen Netzwerks mit n Knoten, welches z.B. Datenbankserver, Clienten und Router haben kann.

Nach Cybenko [2] bezieht sich MTD wie folgt auf die Cybersicherheit.

3.1.1 Kernziele der Cybersicherheit

Die Cybersicherheit lässt sich generell in drei Kernziele einteilen. Zuerst sollte ein Netzwerk Verfügbarkeit garantieren. Das bedeutet formal, dass mindestens einer von allen Knoten bzw. Systemen im Netzwerk nicht gefährdet ist. Des Weiteren sollte Integrität herrschen, was voraussetzt, dass die Mehrzahl, also höchstens die halbe Anzahl der Knoten, nicht gefährdet (oder bereits angegriffen) ist. Am besten jedoch sollte Vertraulichkeit erreicht werden. Das heißt, keiner der Knoten ist gefährdet. Gefährdet soll hier Zugang zu kritischen Informationen bedeuten.

Es gibt somit einen Tradeoff zwischen Vertraulichkeit und Verfügbarkeit. Im schlechtesten Fall trifft Verfügbarkeit, und im besten Fall Integrität und Vertraulichkeit zu.

3.1.2 Abhängigkeit

Abhängigkeit eines Systems bzw. Knotens von einem anderen resultiert, wenn eine Gefährdung des einen Knotens die Gefährdung des anderen zur Folge hat. Eine Zufallsvariable („time-to-compromise Variable“) misst die Zeit, die ein System von der Ausspähung des Exploration surface bis zum Angriff hat. Existiert Abhängigkeit zwischen zwei oder mehreren Knoten, so wird die Zufallsvariable für den Angriff auf den abhängigen Knoten maßgeblich von dem von ihm abhängigen Knoten beeinflusst. So werden nun zwei ähnliche bis gleiche Systeme in einem Netz mit hoher Wahrscheinlichkeit schneller mit nur kurzem Zeitabstand nacheinander angegriffen, wenn einer von beiden erfolgreich angegriffen wurde.

3.1.3 Diversität

Befinden sich im Netz nun nur äquivalente bzw. homogene Knoten, so spricht man von einer Monokultur. Angreifer können dort deutlich schneller weitere angreifen, wenn nur einer zum Opfer gefallen ist. Da hier aufgrund der Ähnlichkeit der Systeme zueinander die Exploration surfaces fast genau die gleichen sind, braucht der Angreifer keinen bis nur noch wenig Aufwand betreiben, um weitere Knoten auszuspähen, nachdem er einen einzigen erfolgreich erforscht hat. Dies ist z.B. eines der Konzepte von Botnets und Code Reuse. Monokulturen sind einfach zu skalieren und zu handhaben, büßen dafür aber Sicherheit ein. Die Zufallsvariablen für Angriffe der Knoten sind normalerweise nicht voneinander unabhängig.

Heterogene, also Knoten die verschieden voneinander sind, müssen alle einzeln mit erhöhtem Aufwand vom Angreifer ausgeforscht werden. Sie sind kostspieliger und schwieriger zu warten als Monokultursysteme, bieten aber erhöhte Sicherheit. Natürliche Diversität wird erreicht, wenn man ein heterogenes Netz hat, bei dem keine der Zufallsvariablen voneinander abhängig sind.

3.2 Systemagilität

3.2.1 Herausforderungen an die Systemagilität

Das ganze Konzept der MTD sollte so gut wie möglich vor dem Angreifer verheimlicht werden, also transparent sein. Das konkrete Verfahren sollte möglichst unvorhersehbar sein. Zur Prävention von Erraten durch den Gegner muss der Konfigurationsraum genügend groß gewählt werden.

Es ist außerdem eine Herausforderung, die Sicherheit aller Schichten des OSI-Modells aufrecht zu erhalten [3]. Die Konsistenz der Abhängigkeiten durch alle Schichten muss gewährleistet werden. Ein Beispiel wäre Multipath-Routing in einem Netzwerk. Bewegt man z.B. einen Dienst vom einen Knoten zum Nächsten, was natürlich von außen nicht sichtbar geschehen soll, muss man beachten, dass Angreifer TCP-Header mit Informationen über den neuen (und alten) Aufenthaltsort des Dienstes abfangen können.

Außerdem muss auch das Kostenmanagement berücksichtigt werden. Um Agilitätstechniken anzuwenden, müssen entweder zusätzliche Assets zu einem System installiert werden oder Performance im vorhandenen System eingebüßt werden.

3.2.2 Agilitätsmechanismen

Heutzutage sind noch die meisten Agilitätsmanöver reaktiv, was bedeutet, dass eine Verteidigungstechnik greift, nachdem ein Angriff festgestellt wurde. MTD soll proaktiv wirken, also präventiv Angriffe verhindern. Beispiele für Agilitäts-mechanismen sind Wrapperklassen, welche den Input für sicherheitsempfindliche Interfaces (z.B. System calls) umschreiben können und Verteilerdienste, die andere Dienste steuern (also allokieren, platzieren oder ausschalten) können. Weitere sind z.B. die Möglichkeit, Systemeigenschaften und –interfaces zur Laufzeit zu ändern und Täuschungstechniken, wo der Angreifer zur Erhöhung seines Aufwands gezwungen wird, indem man zuerst sein Verhalten beobachtet und ihn dann mit falschen Informationen füttert.

3.3 Effective Movement

MTD hat das Ziel, Systeme weniger deterministisch und statisch zu gestalten. Sind sie untereinander außerdem weniger homogen, wird der Aufwand, ein System erfolgreich anzugreifen, deutlich gehoben. Dies ist ein Beispiel für die Theorie des Effective Movement, deren Hauptleitfragen sich nach [4] folgendermaßen ergeben.

3.3.1 Umfang

Zunächst stellt sich die Frage, welche Elemente von der Technik abgedeckt werden sollen („Coverage“). Optimaler Weise sollte das gesamte Attack surface davon umschlossen werden und von der MTD Technik bewegt werden. Die Herausforderung, alle verwundbaren bzw. schützenswerten (also im Attack surface befindlichen) Komponenten im System zu finden, spielt dabei eine grundlegende Rolle.

doi: 10.2313/NET-2015-03-1_12

3.3.2 *Unvorhersehbarkeit*

Der Raum für die Bewegung muss hinreichend groß sein, um erfolgreiches Raten möglichst auszuschließen. Dieser wird durch die Kardinalität des Konfigurationszustandsraums bestimmt. Hierbei gilt: Je höher die Entropie einer Komponente, desto höher ist die Güte der Unvorhersehbarkeit. Die Entropie eines Knotens sinkt mit steigender Abhängigkeit der Knoten voneinander – je mehr Referenzen sie also untereinander haben.

3.3.3 *Rechtzeitigkeit*

Das Zeitmanagement der MTD spielt ebenso eine entscheidende Rolle für die Qualität der Cyberabwehr („Timeliness“). Die Konfigurationswechsel sollten am besten genau zwischen der Ausspähung des Systems durch den Angreifer und dem eigentlichen Angriff erfolgen. Es ist nützlich, viele Informationen über den Angreifer zu sammeln und daraus - beispielsweise durch einen Machine-Learning-Algorithmus - ein „Attacker model“ zu generieren, anhand dessen man seine Verteidigungsstrategie optimal anpasst. Hier stellt sich das Problem, dass diese Ausspähung von der Seite des Verteidigers aus schwierig zu erkennen ist. Fehlen solche Informationen, kann man den Zeitplan nicht an die Bedrohung anpassen und es liegt nahe, dass man die Zustandsbewegung möglichst oft vollzieht und dabei blind hofft, einem Angriff zuvorzukommen.

Weitere wichtige Kriterien für die Effektivität sind unter anderem der resultierende Overhead in Performanz, Speicher oder Netzwerk, die Kosten für z.B. Entwicklung, Einsatz, und Wartung, die Expertise, die auf menschlicher Ebene notwendig ist um mit dieser neuen Technik umzugehen und die Kompatibilität mit den bereits vorhandenen Komponenten.

3.4 MTD als spieltheoretischer Ansatz

3.4.1 *Modell*

Die Interaktion zwischen Angreifer und Verteidiger bei MTD kann man gut als Leader-Follower-Spiel mit zwei Spielern modellieren [5]. Abbildung 3 zeigt den Aufbau des Spiels. Das „Spielfeld“ sind die Plattformen, welche abwechselnd von Spielern „besetzt“ werden. Diese simulieren bei der MTD die Konfigurationszustände und deren Parameter haben jeweils unterschiedliche Werte. Es kann immer nur genau eine Plattform gespielt werden, was übertragen auf die Realtität bedeutet, dass ein Konfigurationszustand gerade auf einem System aktiv ist. Ein Knoten ist verwundbar/angreifbar, wenn er mindestens eine ausbeutbare Eigenschaft hat (rote Kreise im Bild). Besetzt der Verteidiger gerade einen verwundbaren Knoten und der Angreifer wählt diesen für seinen nächsten Spielzug, so wird er angegriffen.

Der Verteidiger ist der Leader im Spiel, also macht seinen Spielzug als Erster. Er weiß nicht, welche Knoten im Netz angreifbar sind und hat keine Möglichkeit herauszufinden, welche Knoten angegriffen werden - er sieht die Spielzüge des Angreifers also nicht. Er kann einmal befallene Knoten auch nicht wieder zurückgewinnen. Sein Ziel ist es, persistenter Bedrohung über die Dauer des Spiels vorzubeugen.

Der Angreifer ist der Follower und besitzt komplette Information darüber, welche Knoten angreifbar/verwundbar sind. Er hat eine begrenzte Menge an Exploits als Angriffsmittel, die jeweils für bestimmte Plattformen greifen und kann ausschließlich den momentan aktiven Knoten angreifen.

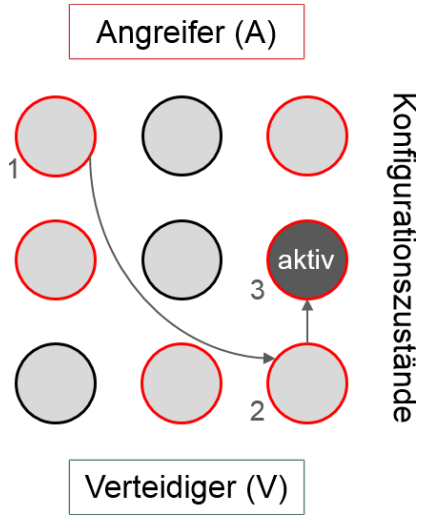

Abbildung 3: Spielfeld mit 3 Zügen nacheinander. Rote Felder: verwundbare Zustände, schwarze unverwundbare.

Spielt der Verteidiger über eine bestimmte Zeitdauer immer verwundbare Knoten, für die der Angreifer einen Exploit hat (und den auch anwendet), resultiert eine persistente Bedrohung und der Angreifer kann gewinnen. Die zeitliche Dauer kann z.B. in Minuten oder in der Anzahl der Spielzüge gemessen werden.

Das Spiel kann nun im Modus „Crash the Satellite“ oder im „Exfiltration Over a Difficult Channel“ gestartet werden. Hat der Angreifer bei „Crash the Satellite“ für eine bestimmte Dauer Kontrolle über das System (= persistente Bedrohung), so hat er gewonnen. Bei „Exfiltration Over a Difficult Channel“ muss der Angreifer über eine bestimmte Zeitdauer ein verwundbares System besetzen, damit sein Angriff erfolgreich ist. Ist das geschehen, so steigt seine Payoff-Funktion mit der Zeit.

3.4.2 *Wahl der Strategien*

Der Verteidiger geht intuitiv davon aus, dass der letzte Knoten, den er gespielt hat, angegriffen wurde und verwundbar war. Je ähnlicher ein System einem anderen ist, desto wahrscheinlicher teilen sie sich auch eine oder mehrere Verwundbarkeiten. Deshalb wird der Verteidiger als Nächstes einen möglichst unterschiedlichen spielen.

Der Angreifer kann auf zwei Arten operieren: im statischen Modus hat der Angreifer bereits am Anfang alle Fähigkeiten und gewinnt nichts durch das Beobachten des Systems. Hier ist das einzige Ziel des Verteidigers, die Wahrscheinlichkeit, dass der Angreifer eine verwundbare Plattform trifft, zu minimieren.

Im adaptiven Modus kann er über Zeit Gegenmaßnahmen entwickeln, je mehr Informationen er durch das Beobachten des Netzes, also die Züge des Verteidigers gewonnen hat. Als Beispiel könnte man hier einen Spam-Filter nennen, wo ein Angreifer nach jedem Senden einer Mail etwas über die Eigenschaften des Filters lernt und Antwortmails generieren kann. Der Verteidiger muss nun sowohl Angriffe abwehren als auch die Wahrscheinlichkeit, dass der aktuelle Knoten getroffen wird, minimieren.

doi: 10.2313/NET-2015-03-1_12

3.4.3 *Ergebnisse*

Die statistisch optimale Lösung nach der Auswertung von verschiedenen Gefahrenmodellen resultiert, wenn die Plattformwahl präferenziell geschieht - wohingegen rein zufällig gewählte Knoten nicht die klügsten Entscheidungen sind. Bezogen auf MTD heißt das, dass Verteidigung am besten klappt, wenn die Konfigurationen immer so verschieden zur vorherigen gewählt werden, wie es geht. Allerdings bleibt zu berücksichtigen, dass die optimalen Moving Target Strategien immer vom Modell abhängen, also möglichst individuell auf das Vorgehen des Angreifers eingegangen werden muss.

4. Beispiele

4.1 Address Space Layout Randomization

Address Space Layout Randomization (ASLR) ist laut [4] das derzeit verbreitetste Beispiel für MTD und wird z.B. eingesetzt, um der Bedrohung von Code Reuse und Code Injection Angriffen entgegenzuwirken. Wenn Angreifer Speicherorte von nützlichen Datenobjekten und Code einer Software herausfinden, können sie mit darauf zugeschnittenen Exploits jedes System schädigen, das diese Software benutzt, da die Speicheradressen überall die gleichen festen Offsets haben.

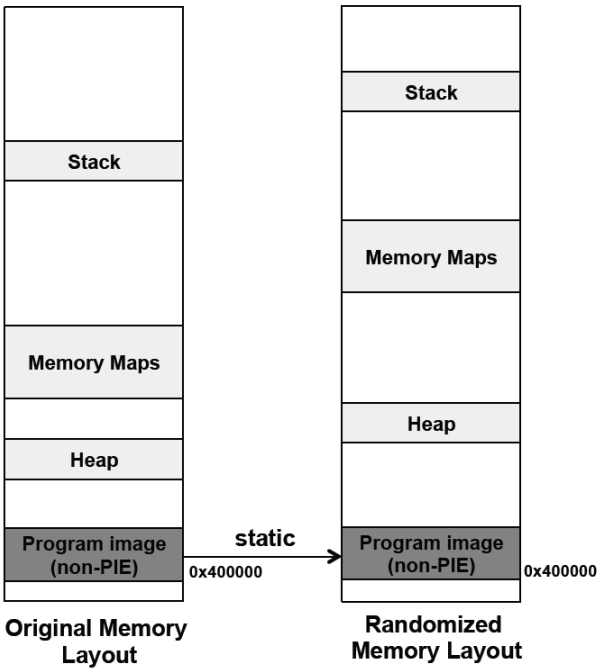

Abbildung 4: Speicherlayout ohne positionsunabhängigen Executables (PIE) [4]

Optimaler Weise sollen also auch die Program executables eine dynamische Speicheradresse bekommen. So wären alle Programmteile von der MTD-Technik abgedeckt und zusammen mit einer klugen Wahl der neuen virtuellen Speicheradressen wären keine davon leicht vorhersehbar.

ASLR löst die Programmteile von ihren statischen Adressen im Speicher, sodass sie dynamisch geändert werden bzw. an zufälligen Speicherorten stehen und das Speicherlayout also auf jedem System anders aussieht. Die meisten ASLR-Implementationen randomisieren den Stack, Memory Maps und den Heap (siehe Abbildung 4), aber das Program image ist oft noch positionsabhängig (PIE), da positionsunabhängige (non-PIE) Programme gesondert kompiliert werden müssen.

Ein Problem, das sich bei dieser Technik leider seit einiger Zeit ergibt, ist das *Spraying*, wo Schadcode ähnlich einer Spraydose großflächig in den Speicher gesprüht und dupliziert wird. So soll die Wahrscheinlichkeit steigen, dass trotzdem irgendwann ein Bibliotheksaufruf durch den eingeführten Code geschieht.

4.2 Spatio-temporal Address Mutation

Hierunter versteht man das dynamische Mapping von Hosts an IP-Adressen [6]. Als Konfigurationsparameter werden in diesem MTD-Konzept also IPs verwendet. Die Basis besteht darin, dass jeder Host, der mit einem anderen interagieren will, eine kurzlebige (engl. ephemeral) eIP zugewiesen bekommt, mit der er dann ausschließlich mit dem angegebenen Partner kommunizieren darf.

Ein sogenannter Controller übernimmt die Mappings der (realen) rIPs auf eIPs pro Host, bestimmt die Mutationsstrategie, autorisiert Zugriffe mit rIPs (welche nur von Administratoren ausgeführt werden dürfen) und schickt die Mapping-Tabelle an Gateways. Diese sind für die Transformation der IPs selbst verantwortlich – übernehmen also die „Arbeit“. Sie prüfen die IPs jedes Pakets, das passieren will, anhand der Mapping-Tabelle auf Legalität und transformieren die rIPs anschließend in eIPs und umgekehrt, oder verwehren ungültigen Paketen den Weg.

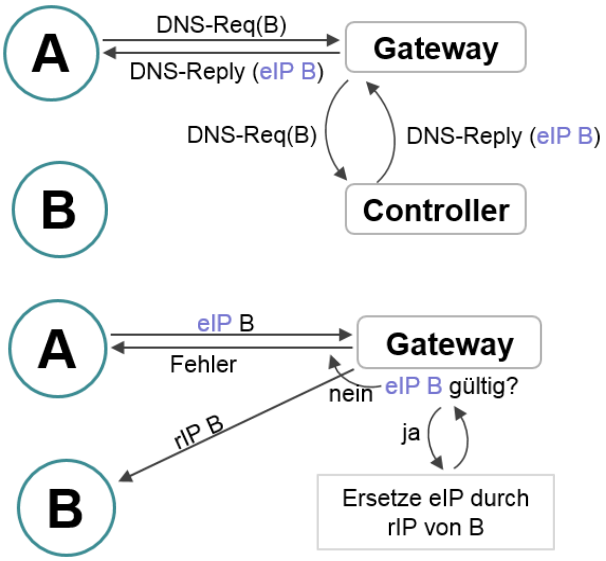

Abbildung 5: Kontaktaufnahme (oben) und Kommunikation (unten) bei STAM

Zur Kontaktaufnahme muss ein Host zunächst ein DNS-Request für den Empfänger starten. Der Controller erstellt nach dessen Eingang ein neues Mapping der Empfänger-rIP an eine eIP,

doi: 10.2313/NET-2015-03-1_12

welche mit der DNS-Reply zurück an den Anfragesteller-Host geht. Dieser verwendet daraufhin die neue eIP für jedes Paket, das er an diesen Partner schickt. Nachdem das Gateway die Gültigkeit der Pakete validiert und die Transformation der eIP zur rIP des Empfängers vorgenommen hat, schickt es sie weiter. Abbildung 5 visualisiert diese Vorgänge vereinfacht.
Die eIPs sind kurzlebig, weil sie vom Controller einen time-to-live (TTL) Wert als Verfallszähler bekommen, was die temporale Komponente des Konzepts darstellt. Die Wahl des TTL-Werts ist gemäß des Poisson-Erneuerungsprozesses. Die spatiale (räumliche) Komponente ist die Entscheidung, welche eIP als nächstes verwendet werden soll. Hierzu kann die Wahl entweder uniform oder täuschend sein. Bei der uniformen Wahl nimmt der Controller irgendeine aus den verbleibenden eIPs, wobei jede IP die gleiche Wahrscheinlichkeit hat. Bei der täuschenden Mutation nimmt der Controller die eIP, für welche er einen Angriff am unwahrscheinlichsten hält.

4.3 Field Programmable Gate Arrays

Field Programmable Gate Arrays (FPGAs) sind rekonfigurierbare Hardwarekomponenten bzw. Rechenbausteine, die dabei helfen, kritische Programmabschnitte auf Hardwareebene zu partitionieren (*Hardware Partitioning)* und somit zu beschützen. FPGAs sind weitaus primitiver gebaut und daher auch billiger als normale CPUs.
Dadurch, dass die FPGAs individuell konfigurierbar sind, kann der Softwaredesigner selbst entscheiden, welche Teile des Programms auf den FPGAs laufen sollen. Der wichtigste Vorteil hiervon ist die Verbesserung der Performanz des Systems. Implementiert man ein Sicherheitsprogramm z.B. für ein FPGA, so wird zu dessen Ausführung keine CPU-Zeit in Anspruch genommen, wodurch man also Parallelität hergestellt hat.
Besonders praktisch bezüglich Sicherheitsrisiken sind sie, wenn man im Adressraum von Programmen springen muss. Man kann das Programm in zwei Bereiche aufteilen, sodass die gefährdeten Teile des Programms auf einem FPGA und die sicheren Teile auf einer (oder mehreren) herkömmlichen CPU implementiert werden. Moving Target Defense ist hier also die Auslagerung bzw. Bewegung von Programmteilen innerhalb der Rechnerarchitektur [7].
Ein spezieller Ansatz für den Einsatz von FPGAs für Datensicherheit ist *CODESSEAL*. Hier befinden sich Programm und Daten zur Laufzeit verschlüsselt im Speicher. Zwischen dem Cache und dem Speicher befindet sich ein FPGA zur Ver- und Entschlüsselung der Daten, die von der CPU verarbeitet werden. Auf diese Weise befinden sich nur verschlüsselte Daten zwischen externer Hardware und Prozessor, also können Adressen und Datenleitungen nicht gesnifft werden.

5. Zusammenfassung

Man kann also folgern, dass MTD eine zukunftsorientierte Technik ist. Es soll nicht mehr still darauf gewartet werden, bis man von einer Cyberattacke angegriffen ist um sich dann erst zu versuchen zu verteidigen, sondern sich schon im Voraus durch geschickte Veränderungen der im Netz bzw. von Angreifern sichtbaren Systemressourcen so unkenntlich wie möglich zu machen.
Je höher der Grad der Diversifikation und der Randomisierung, desto geringer ist die Wahrscheinlichkeit, dass der Angreifer den nächsten Konfigurationszustand voraussagen kann. Dies kann durch zufällige Wahl oder besser noch kombiniert mit Angriffserkennungssystemen erreicht werden.

Als Schutz vor Buffer-Overflows wird heutzutage oft eine Technik eingesetzt, die einzelne Pages bzw. Datenobjekte im Speicher schützt und einen IDS-Alarm gibt, sobald geschützte Pages von User-Anfragen berührt werden [8]. Abbildung 6 zeigt ein solches Schema. Mit dieser Schutztechnik können nicht alle Pages gleichzeitig abgedeckt werden, da sich die Antwortzeit des Servers proportional zur Anzahl bewachter Pages verhält.
Hier wäre es nun eine MTD Idee, den Schutz der Pages dynamisch infolge von bereits erfolgten Angriffen anzupassen. So könnte man z.B. unter Einbezug von Machine Learning Algorithmen mehr Pages in der Nähe derer schützen, die schon einmal angetastet wurden.

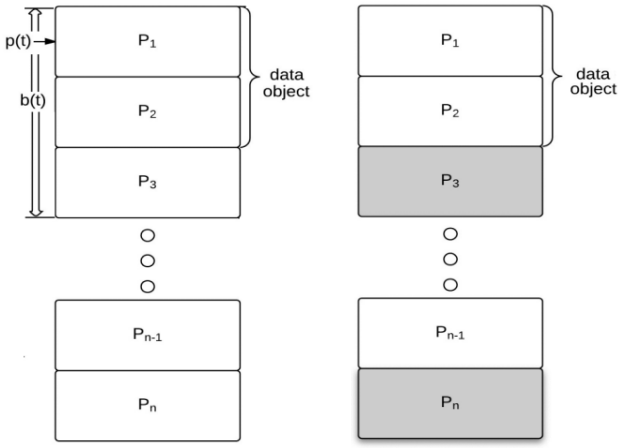

Abbildung 6: Speicherlayout ohne Page-Schutz (links) und mit Page-Schutz (rechts). b(t) = ausgelesene Pages durch das Payload-length-Feld; p(t) = erste zu lesende Page.

6. Referenzen

[1] R. Zhuang, S. A. DeLoach, X. Ou. Towards a Theory of Moving Target Defense, 2014.

[2] G. Cybenko, J. Hughes. No Free Lunch in Cyber Security, 2014.

[3] P. McDaniel, T. Jaeger, T. F. La Porta, N. Papernot. Security and Science of Agility, 2014

[4] T. Hobson, H. Okhravi, D. Bigelow. On the Challenges of Effective Movement, 2014.

[5] K. M. Carter, J. F. Riordan, H. Okhravi. A Game Theoretic Approach to Strategy Determination for Dynamic Platform Defenses, 2014.

[6] J. H. Jafarian, E. Al-Shaer, Q. Duan. Spatio-temporal Address Mutation for Proactive Cyber Agility against Sophisticated Attackers, 2014.

[7] T. R. Andel, L. N. Whitehurst, J. T. McDonald. Software Security and Randomization through Program Partitioning and Circuit Variation, 2014.

[8] M. Zhu, Z. Hu, P. Liu. Reinforcement Learning Algorithms for Adaptive Cyber Defense against Heartbleed, 2014

doi: 10.2313/NET-2015-03-1_12