POLYMORPHIC SHELLCODE

Radovan Plocek

Bachelor Degree Programme (3), FIT BUT

E-mail: xploce00@stud.fit.vutbr.cz

Supervised by: Jakub Křoustek

E-mail: ikroustek@fit.vutbr.cz

Abstract: Exploits, which are using polymorphic shellcodes, are serious problem for the current intrusion detection systems and antivirus programs based on signature detection, because it is complicated to detect them. This paper describes existing methods of creating the polymorphic shellcodes and their improvements. The application, which is based on these methods, can be used for testing the performance of the detection systems.

Keywords: Exploit, Shellcode, Polymorphism

1. ÚVOD

S ohledem na velký růst vlivu internetu na současnou společnost nabývá počítačová bezpečnost čím dál většího významu. Způsobů, kterých lze použít k útoku na uživatele v prostředí internetu, existuje velké množství. Mimo všeobecně známé viry a červy je to například právě zneužití zranitelnosti programu pomocí shell-kódů. Podstatným milníkem při vývoji metod jejich tvorby se stalo převzetí techniky polymorfismu, známé z vývoje počítačových virů. Ta dovoluje vytvářet shell-kódy se stejnou funkcí, ale různou podobou, čímž znesnadňuje jejich odhalení systémy pro detekci průniku a antivirovými programy založenými na detekci vzorů. Tato práce popisuje současné metody tvorby polymorfních shell-kódů a jejich úpravy implementované ve vyvíjené aplikaci, která jejich využitím bude transformovat klasické shell-kódy na polymorfní.

2. SHELL-KÓDY

Ze všeho nejdříve je třeba si uvést některé obecné poznatky o klasických shell-kódech, které jsou z větší části platné i pro shell-kódy polymorfní (následující informace jsou pouze stručným přehledem, hlubší úvod do problematiky můžeme najít v [1] a [2]). Důležité je zejména vědět, co to vlastně shell-kód je. Jedná se o posloupnost bajtů, které reprezentují instrukce ve strojovém jazyce a případná data. V původním významu bylo jejich cílem zpřístupnit tvůrci shell, což je interpret příkazů, umožňující uživateli například využívat funkce jádra či volat programy. V současné době se význam trochu posunul a nyní shell-kódem označujeme jakýkoli byte-kód dělající to, co chce jeho autor.

Když už víme, co to shell-kód je, popíšeme si jeho použití, které je stejné jak pro jejich základní variantu, tak i pro variantu polymorfní. Abychom mohli shell-kód úspěšně použít, musíme v programu, na který chceme zaútočit, najít nejdříve bezpečnostní slabinu. Tou nejčastěji bývá přetečení zásobníku, které je způsobeno nedůsledností programátora a podceněním kontroly vstupů. Jakmile takovou slabinu nalezneme, můžeme pomocí ní do programu vložit náš shell-kód, který již provede námi požadovanou akci.

Nyní nám již pouze zbývá popsat si jeho podobu. Jelikož shell-kód do programu kopírujeme ve formě řetězce, je nutné (uvažujeme-li jazyk C), aby neobsahoval žádné nulové bajty. Ve vlastním shell-kódu pak můžeme rozlišit tři hlavní části: [NOP sled] [Operační kód] [Opakující se návratová adresa]. NOP sled je posloupnost instrukcí NOP (0x90), které neprovádějí žádnou operaci. Vyko-

návání programu po nich doputuje až k operačnímu kódu, který provede námi požadovanou akci. Poslední součástí je opakující se návratová adresa, ukazující někam dovnitř NOP sledu. Tou se přepíše návratová adresa funkce, ve které je obsažena zranitelnost přetečením zásobníku, takže se posléze začne vykonávat náš shell-kód.

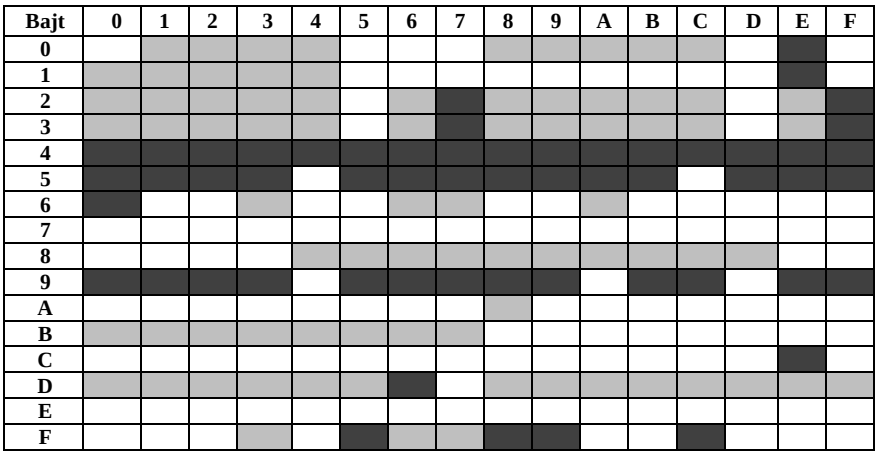
3. POLYMORFISMUS

Polymorfismus je jedním z pokročilých prvků počítačových útoků, který téměř znemožňuje odhalení útoku pomocí obranných systémů založených na detekci vzorů. Tento způsob detekce hledá známé vzory a v případě jejich rozpoznání je ohlásí. Dokáže rozpoznávat ale i vzory nové. To je ovšem velmi problematické právě u polymorfních shell-kódů. Vzhledem k různým podobám je nedokáží tyto systémy rozpoznat. Z toho důvodu se rozvíjí další druhy rozpoznávání, například detekce anomálií v síťovém provozu. Jednotlivé metody tvorby polymorfních shell-kódů jsou popsány v následujících podkapitolách, společně s jejich navrženými úpravami a zatím dosaženými výsledky.

3.1. FALEŠNÝ NOP SLED

Dlouhou posloupnost NOP instrukcí v klasických shell-kódech je velmi snadné objevit. Proto ji potřebujeme nahradit či zamaskovat. K tomu můžeme využít množství jiných instrukcí. Problémem ovšem je, že návratová adresa může ukazovat na kterýkoli bajt sledu, který ovšem vždy musí být začátkem platné instrukce, jinak by shell-kód nefungoval. Je tedy třeba, aby všechny nižší bajty vícebajtových instrukcí byly taktéž samostatnými instrukcemi. Z toho důvodu se v současných aplikacích pro tvorbu polymorfního shell-kódu používají většinou pouze jednobajtové instrukce a o použití vícebajtových instrukcí se zmiňují pouze v teoretické rovině.

Vyvíjená aplikace ovšem pracuje i s vícebajtovými kombinacemi. V současné době využívá 56 jednobajtových instrukcí a 1616 dvoubajtových kombinací. Zastoupení bajtů ve spektru znázorňuje **Tabulka 1**. Tmavé buňky ukazují zastoupené bajty v případě použití pouze jednobajtových instrukcí, světlé znázorňují zastoupení jednobajtových i dvoubajtových. Počet zastoupených bajtů se zvýšil z 56 na 135. V rámci dalšího vývoje je v plánu zvětšit počet zastoupených bajtů pomocí tří- a vícebajtových kombinací, což vyžaduje ještě další důkladné testování.



Tabulka 1: Výskyt bajtů ve falešném NOP sledu

3.2. ZAKÓDOVÁNÍ OPERAČNÍHO KÓDU

I nezakódovaný operační kód je snadné detekovat. Proto při generování polymorfního shell-kódu šifrujeme operační kód a zároveň s tím vytváříme dešifrovací posloupnost, která při provádění programu operační kód dešifruje do původní podoby. Nejčastějším způsobem šifrování je použití bito-

vé funkce XOR. Použít ale můžeme i další operace, jako jsou přičítání, odčítání, inkrementace, posun, rotace a další. V současné době se většinou dešifrovací posloupnost vkládá před zašifrovaný operační kód.

V rámci vyvíjené aplikace je v plánu implementovat i umístění dešifrovací posloupnosti až za operační kód či použití dvou dešifrovacích posloupností – jedné před operačním kódem a druhé za ním, jak je popsáno v [4]. Mezi jednotlivé bajty instrukcí dešifrovací posloupnosti jsou pro větší náhodnost vloženy další bezpečné instrukce, neovlivňující funkci dešifrování. Tyto úpravy zvyšují variabilitu výsledného shell-kódu.

3.3. ZMĚNA NÁVRATOVÉ ADRESY

Stejně jako u předchozích dvou částí nepolymorfních shell-kódů platí i u části s návratovými adresami, že jsou detekčními systémy velmi snadno odhalitelné. Z tohoto důvodu se mění dolní bity adresy tak, aby byla pokaždé různá, ale stále ukazovala dovnitř NOP sledu. Ve vyvíjené aplikaci je této metody využito bez dalších úprav.

3.4. VÝPLŇ

Jak bylo řečeno v úvodu této kapitoly, mimo detekci vzorů se vyvíjí i například detekce anomálií využívající spektrální analýzu provozu na síti. Pokud známe spektrum přenášených dat, můžeme odhadnout, zda daný tok je bezpečný, či nikoli. V rámci polymorfního shell-kódu můžeme dodatečně vložit další instrukce právě pro přizpůsobení se spektru provozu. Zároveň výplní můžeme zmenšit velikost části shell-kódu s opakující se návratovou adresou. Tato metoda byla popsána v [3]. Ve vyvíjené aplikaci je této metody využito zejména pro zmenšení části shell-kódu s návratovou adresou.

4. ZÁVĚR

Při zneužití zranitelnosti programu pomocí polymorfních shell-kódů se systémy pro detekci vzorů dostávají do nesnází. Sice některé jednodušší polymorfní techniky mohou zanechávat určité detekovatelné posloupnosti, ale při kvalitní implementaci pokročilých technik se dají tyto artefakty výrazně omezit až úplně odstranit. Vyvíjená aplikace umožňuje testování detekčních systémů a antivirových programů. Kombinováním různých technik polymorfismu a jejich úprav lze zjistit, jaké jsou limity současných obranných systémů.

PODĚKOVÁNÍ

Tento příspěvek vznikl za podpory výzkumného záměru MSM 21630528.

REFERENCE

[1] ERICSON, Jon. *Hacking: umění exploitace*. 2. upravené a doplněné vydání. Brno: Zoner Press, 2009. 544 s. ISBN 978-80-7413-022-9.

[2] ONE, Aleph. Smashing The Stack For Fun And Profit. *Phrack Magazine* [online]. 1996, 7 (49), [cit. 2011-03-04]. Dostupný z WWW: <http://www.phrack.com/issues.html?issue=49&id=14#article>.

[3] DESTRIAN, T., ULENSPIEGEL, T., MALCOM, Y., VON UNDERDUK, M. S. Polymorphic Shellcode Engine Using Spectrum Analysis. *Phrack Magazine* [online]. 2003, 9 (61), [cit. 2011-03-04]. Dostupný z WWW: <http://www.phrack.com/issues.html?issue=49&id=14#article>.

[4] KEROMYTIS, Angelos. *On the Infeasibility of Modeling Polymorphic Shellcode* [online]. Alexandria, Virginia, USA, 2007. 11 s. Oborová práce. Columbia University. Dostupné z WWW: <http://www.cs.columbia.edu/~angelos/Papers/2007/polymorph.pdf>.