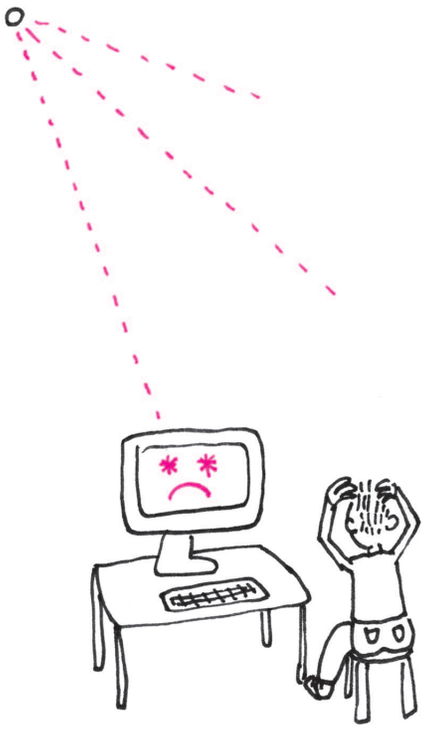
Promieniowanie kosmiczne a bezpieczeństwo komputerowe

*Krzysztof PIECUCH**

*student, Wydział Matematyki i Informatyki, Uniwersytet Wrocławski

Jak czarne dziury, odległe supernowe i inne wydarzenia kosmiczne mogą wpłynąć na bezpieczeństwo systemów komputerowych?



Czy możemy w cyfrowym świecie czuć się bezpiecznie? Czy jesteśmy pewni, że zabezpieczenia, których używamy, są w stu procentach niezawodne? Pewnym gwarantem jest dowód matematyczny bezpieczeństwa danego kryptosystemu. Jednak w takich dowodach zazwyczaj przyjmuje się, że komputery działają bezbłędnie i nic nie zakłóca obliczeń wykonywanych przez procesor. Niestety, komputer jest tylko maszyną i zdarza mu się czasem coś źle obliczyć.

Typowym błędem w obliczeniach wykonywanych za pomocą komputera jest tzw. przekłamanie bitu. Polega ono na zamianie któregoś bitu z wartości 0 na wartość 1 lub na odwrót. Przyczynić się do tego może usterka fabryczna albo np. przegrzanie się sprzętu. Najczęstszym powodem jest jednak promieniowanie kosmiczne.

Promieniowanie kosmiczne odkryto w 1912 roku. Gdy dochodzi ono do górnej części atmosfery, powstaje deszcz wysokoenergetycznych cząstek, które przedostają się do powierzchni Ziemi. Cząstka może wejść w interakcje, na przykład, z pamięcią RAM naszego komputera, powodując wyżej opisany błąd.

Pod koniec dwudziestego wieku firma IBM przeprowadziła serię badań na ten temat. Wynikało z nich, że liczba przekłamań zależy m.in. od położenia geograficznego. Zwiększa się, gdy zbliżamy się do biegunów ziemskich. W jaskiniach liczba błędów malała praktycznie do zera, a rosła, im wyżej wzbijaliśmy się w niebo (to poważny problem, jeśli chodzi o samoloty i statki kosmiczne). Średnia liczba błędów była równa jeden na miesiąc na każde 256 MB pamięci RAM. Należy pamiętać, że testy te zostały przeprowadzone pod koniec dwudziestego wieku. Obecnie jesteśmy nieco bardziej zaawansowani technicznie, ale z drugiej strony również pamięć RAM staje się coraz bardziej czuła na takie zakłócenia.

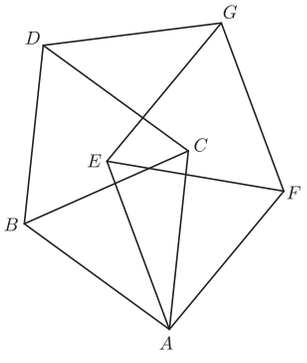
No dobrze, ale czy zmiana jednego bitu może mieć jakikolwiek wpływ na poziom bezpieczeństwa komputerowego?

W 2003 roku Sudhakar Govindavajhala i Andrew Appel przedstawili ciekawy program napisany na wirtualną maszynę Javy (patrz [2]). Jeśli w trakcie jego działania nastąpi przekłamanie dowolnego bitu, to z prawdopodobieństwem 70% możemy na maszynie wykonać dowolny, w szczególności złośliwy, kod. Prawdopodobieństwo wzrasta z każdym następnym przekłamaniem.

Idea programu jest następująca. Wypełniamy całą dostępną pamięć wskaźnikami na typ całkowity `int`. Jeśli teraz nastąpi przekłamanie pewnego bitu, to jest duża szansa, że wskutek tego któryś ze wskaźników (nazwijmy go A) zacznie wskazywać na inny, przypadkowy wskaźnik (nazwijmy go B). Nasz program próbuje wykryć taką sytuację. Ponieważ A powinien wskazywać na miejsce w pamięci, w którym jest zmienna typu `int`, to możemy wówczas w tym miejscu zapisać dowolną wartość liczbową (powiedzmy jakiś adres danych wirtualnej maszyny). Teraz patrzymy na B. Trzymamy tutaj adres zmiennej typu `int`. Jednakże ten adres przed chwilą zmieniliśmy na dowolny wybrany przez nas. Pod tym adresem możemy teraz zapisać dowolną wartość, nadpisując jakieś dane wirtualnej maszyny. Powtarzając ten proces, możemy krok po kroku zapisać w pamięci złośliwy kod. Następnie możemy nadpisać (w ten sam sposób) tablicę metod wirtualnych i uruchomić zapisany kod. W ten sposób jesteśmy w stanie obejść ograniczenia na uruchamiane programy, jakie narzuca wirtualna maszyna Javy.

Prawdopodobieństwo przekłamania pojedynczego bitu na konkretnym komputerze jest bardzo małe. Jednak prawdopodobieństwo przekłamania bitu

Rozwiązanie zadania M 1365.
Rozważmy konfigurację punktów A, B, C, D, E, F, G z rysunku, gdzie każdy odcinek ma długość 1.



Jeśli teza nie zachodzi dla żadnej pary punktów wybranej spośród A, B, C, to każdy z tych punktów jest innego koloru. Wtedy albo D jest tego koloru co B lub C, albo D jest tego samego koloru co A, powiedzmy, zielonego. Jeśli każdy z punktów A, E, F jest innego koloru, to albo G jest tego koloru co E lub F, albo G jest tego koloru co A, czyli zielonego. Ale wówczas oba punkty D i G są zielone, co kończy dowód.

Dan Boneh, Richard A. DeMillo, Richard J. Lipton, *On the importance of checking cryptographic protocols for faults*, J. Cryptology 14 (2), ss. 101–119, 2001.

na jakimś z wielu komputerów, podpiętych do sieci Internet, jest już znacznie większe. Skorzystał z tego Artem Dinaburg (patrz [1]). Przeprowadził on eksperyment, w którym zarejestrował kilka adresów stron różniących się o jeden bit od pewnych popularnych domen (np. `mic2osoft.com`, `a-azon.com` albo `fjcdn.net` zamiast, odpowiednio, `microsoft.com`, `amazon.com` i `fbcdn.net`). Wyniki były zaskakujące. W ciągu 6 miesięcy Dinaburg przechwycił 52 tysiące zapytań z 13 tysięcy różnych adresów IP. Podczas eksperymentu serwery zapisywały zapytania w bazie danych, a następnie wysyłały komunikat 404 (*Not Found*). Łatwo jednak wyobrazić sobie, jak zastosować tę metodę do niecnych celów. Kradzież ciasteczek, phishing, czyli wyłudzanie poufnych informacji osobistych, czy uruchamianie złośliwego oprogramowania to tylko niektóre przykłady.

Jednym z najpopularniejszych kryptosystemów jest RSA. Bazuje on na spostrzeżeniu, że łatwo jest przemnożyć dwie liczby pierwsze, natomiast bardzo trudno jest przeprowadzić operację odwrotną (czyli rozkład na czynniki). Algorytm tworzy parę kluczy. Jeden z nich jest prywatny – trzymamy go w tajemnicy. Natomiast drugi z nich to klucz publiczny, który udostępniamy pozostałym użytkownikom. Dowolny dokument możemy teraz podpisać swoim kluczem prywatnym. Za pomocą naszego klucza publicznego każdy może zweryfikować, że to faktycznie nasz podpis. Ponadto, z bardzo dużym prawdopodobieństwem, nikt nie jest w stanie podrobić naszego podpisu bez znajomości klucza prywatnego. Okazuje się, że jeśli podczas procesu podpisywania dokumentu w odpowiednim miejscu nastąpi przekłamanie bitu, to możliwe staje się odczytanie klucza prywatnego, a wystarczy do tego zastosować powszechnie znany algorytm Euklidesa (patrz praca podana na marginesie).

Czy możemy w jakiś sposób chronić się przed tego typu niebezpieczeństwem? Znane są kody korekcyjne CRC, które są w stanie wykryć przekłamania bitów. Produkuje się też specjalne pamięci ECC, potrafiące stwierdzić, które z bitów zostały przekłamane. Niestety, żadnej z tych technologii nie stosuje się w laptopach ani komputerach osobistych. Sprawdzanie za każdym razem, czy nie nastąpiło przekłamanie jakichś danych, jest bardzo kosztownym przedsięwzięciem. Czy ma to sens, skoro takie zdarzenia występują bardzo rzadko, a gdy już się zdarzają, to zazwyczaj nie czynią nam wielkiej szkody? Na ciekawe rozwiązanie wpadła tu firma Intel, która w 2007 roku opatentowała pomysł umieszczenia na każdym układzie scalonym detektora promieni kosmicznych. W razie wykrycia promieniowania układ może powtórzyć obliczenia, sprawdzić integralność danych albo poprosić o ponowny transfer danych.

Bardzo trudno w tak krótkim artykule wyczerpać problem przekłamania bitu. Na szczęście w Internecie można znaleźć dużo ciekawych materiałów na ten temat. Obok podajemy dwa, zdaniem autora najlepsze, artykuły dotyczące tego zagadnienia. Tak na dobry początek.

[1] Artem Dinaburg, *Bitsquatting. DNS hijacking without exploitation*, Black Hat Technical Security Conf., 2011.

[2] Sudhakar Govindavajhala, Andrew W. Appel, *Using memory errors to attack a virtual machine*, IEEE Sympos. on Security and Privacy, 2003.

Rozwiązanie zadania M 1363.
Niech

$$n = \overline{abcde} = 10^4a + 10^3b + 10^2c + 10d + e.$$

Wówczas $m = \overline{abde}$. Załóżmy, że $\frac{n}{m}$ jest liczbą całkowitą. Gdyby było $\frac{n}{m} \leqslant 9$, to przyjmując oznaczenia $u = 10^3a + 10^2b$ oraz $v = 10d + e$, mielibyśmy $n = 10u + 100c + v$ i $m = u + v$, a stąd sprzeczność:

$$0 \leqslant 9m - n = 9(u + v) - (10u + 100c + v) = 8v - u - 100c \leqslant 8 \cdot 99 - 1000 < 0.$$

Gdyby było $\frac{n}{m} \geqslant 11$, to

$$0 \leqslant n - 11m = 100c - u - 10v \leqslant 100 \cdot 9 - 1000 < 0$$

i znowu sprzeczność. Zatem $\frac{n}{m} = 10$, co daje $9v = 100c$. Stąd, gdyby $c \neq 0$, liczba $v \leqslant 99$ byłaby podzielna przez 100. Wobec tego $c = v = 0$. Wtedy mamy $n = \overline{ab000}$ – wówczas

$$\frac{n}{m} = \frac{\overline{ab000}}{\overline{ab00}} = 10.$$

Zatem liczby n postaci $\overline{ab000}$ są wszystkimi liczbami pięciocyfrowymi o własności z treści zadania.