Criptoanálise distribuída de alto desempenho

Antonio Marcos de Oliveira Candia, Marcelo Pasin

Laboratório de Sistemas de Computação-LSC/UFSM, Santa Maria, RS
{candia,pasin}@inf.ufsm.br

Resumo

A criptoanálise computacional pode ser definida como o uso do computador para o ataque a um texto criptografado na tentativa de decifrá-lo [SCH 95]. Quebra-pedra é um arcabouço Java para criptoanálise computacional que oferece as técnicas de ataque por força-bruta e busca dirigida por dicionário e foi projetado em torno de dois princípios básicos: independência de plataforma de execução e independência de algoritmos de criptografia [CAN 04].

Os sistemas de criptoanálise atuais não são facilmente adaptáveis a algoritmos diferentes daqueles para os quais foram projetados. Além disso, a linguagem Java, até a versão 1.4.2, é freqüentemente citada por seu baixo desempenho em aplicações computacionalmente intensivas. No arcabouço, estes problemas foram atacados com o uso de dois conceitos disponíveis na plataforma Java: o código reflexivo e a interface nativa Java (*Java Native Interface* - JNI). O uso destes mecanismos permite a carga dinâmica de código nativo sem a necessidade de recompilação do sistema de criptoanálise. A tabela 1 mostra alguns dos resultados obtidos com o uso desta técnica em três sistemas diferentes. Nela pode se ver que o impacto do uso de código misto Java/C foi relativamente baixo e, em alguns casos, desprezível, mostrando a viabilidade do uso deste método.

Porém, mesmo com o uso de código otimizado, um ataque pode ser extremamente custoso devido ao volume de computação requerido [OEC 03, WIE 96]. Por outro lado, constata-se o fato de que estes ataques podem ser categorizados como tarefas trivialmente paralelizáveis. Foi realizada, então, uma implementação paralela dos mesmos, do tipo mestre-escravo e utilizando o sistema de *Sockets* presente em Java para a troca de mensagens entre os nós. Este mecanismo foi escolhido devido às características do problema em questão: necessidade de troca de mensagens pequenas e em pouca quantidade e independência nas tarefas delegadas aos nós. Para os ataques de dicionário assumiu-se que cada escravo contaria com uma lista de palavras previamente instalada. Estas restrições possibilitam o uso de troca de mensagens do tipo datagrama UDP/IP. Para o balanceamento de carga entre os nós que realizam a computação paralela, optou-se por um esquema simples de pré-definir estaticamente os grãos de processamento. Medições de tempo realizadas mostraram que a diferença média em relação a um *speedup* ideal foi menor que 0,1% para até 8 processadores idênticos.

Esta implementação, porém, possui alguns problemas. Por exemplo, cada cliente a ser instalado nos nós deve conter implementações dos algoritmos criptográficos para utilização nos ataques. Qualquer novo algoritmo força a instalação, no mínimo, de um novo adaptador e de uma biblioteca de carga dinâmica nestes clientes. Em um ambiente controlado, com poucos nós, isto é possível, porém, em um ambiente típico de uso isto causa um transtorno considerável, virtualmente inviabilizando o uso do sistema. Outra

Tabela 1: Número médio de iterações por segundo (i/s) das implementações de referência.

	PIV, 2.4GHz, J2SE 1.4.1	PIII, 1GHz, J2SE 1.4.2	Athlon, 1.4GHz, J2SE 1.4.2
DES em C	93640 i/s	57340 i/s	52600 i/s
MD5 em C	1620 i/s	890 i/s	1385 i/s
DES Java+C	88340 i/s	47390 i/s	51070 i/s
MD5 Java+C	1590 i/s	885 i/s	1370 i/s

constatação feita é que, pelo volume de computação necessário, este sistema poderia utilizar a capacidade de processadores eventualmente ociosos em uma rede. Alguns exemplos de sistemas que utilizam esta abordagem de compartilhamento de ciclos ociosos de UCP são o CADEO [CER 04], o SETI@Home e suas variantes. Esta arquitetura de processamento concorrente é chamada, atualmente, de Computação Colaborativa por permitir que usuários - ou seus processadores - colaborem para alcançar a solução de um problema.

No entanto, cabe ressaltar que este modelo de computação introduz alguns problemas novos. Por exemplo, por ser um empréstimo não-controlado de capacidade de processamento, a qualquer momento um usuário pode decidir retirar seu processador deste sistema. Portanto, deve haver uma maior preocupação com características como tolerância a falhas e migração de tarefas não-concluídas.

A programação da versão colaborativa do Quebra-pedra está sendo realizada através da biblioteca ProActive. Essa biblioteca faz parte do projeto *ObjectWeb Middleware* [CAR 98]. Resultados preliminares mostram que o ganho de desempenho é semelhante aos resultados obtidos com o uso de *Sockets*. As vantagens de utilizar o ProActive incluem: um mecanismo transparente de migração de tarefas de processamento, programação simplificada, e o fato do mesmo ser escrito totalmente em Java, permitindo sua portabilidade. Maiores estudos são necessários, entretanto, para verificar se as características de desempenho deste sistema se mantém quando há um grande número de nós sendo utilizados para a computação distribuída dos ataques criptoanalíticos.

Referências

[CER 04] CERA, M. C.; PASIN, M. Suporte em Java para alocação dinâmica de processadores. Anais da 4a Escola Regional de Alto Desempenho, ERAD2004, p. 171–172, Jan. 2004

[CAN 04] CANDIA, A. M. de O. Criptoanálise em Java. Brasília:DPF, Anais da I Conferência Internacional de Perícias em Crimes Cibernéticos, p.125–129, Set. 2004.

[CAR 98] CAROMEL, D. et al. Towards seamless computing and metacomputing in Java. Wiley & Sons, Ltd., v. 10, n. 11–13, p. 1043–1061, 1998.

[WIE 96] WIENER, M. J. Efficient DES Key Search, Technical Report TR-244, Carleton University. IEEE Computer Society Press, 1996.

[OEC 03] OECHSLIN, P. Making a Faster Cryptanalytic Time-Memory Trade-Off. Proceedings of Crypto'03, 2003.

[SCH 95] SCHNEIER, B. Applied cryptography (2nd ed.): protocols, algorithms, and source code in C. John Wiley & Sons, Inc. 1995.