DOI: http://dx.doi.org/10.5769/C2011004

Identificação de Artefatos Periciais do eMule

Rodrigo Lange[1], and Célia Ghedini Ralha[2]
(1) Departamento de Polícia Federal, Setor Técnico-Científico do Paraná, Curitiba, PR, Brasil, rodrigo.rl@dpf.gov.br
(2) Universidade de Brasília, Departamento de Ciências da Computação, Brasília, DF, Brasil, ghedini@cic.unb.br

***Resumo* — Houve um grande aumento da demanda pela realização de exames periciais para a identificação de atividades criminosas realizadas através do compartilhamento de arquivos por redes *peer-to-peer* (P2P), principalmente de material envolvendo a exploração sexual de crianças e adolescentes. O eMule é um dos programas P2P mais utilizados no Brasil e representa uma significativa parcela dos casos identificados de disponibilização e divulgação de material envolvendo pornografia infantojuvenil. Neste trabalho, são detalhados alguns vestígios deixados pelo aplicativo P2P eMule, os quais permitem a comprovação da materialidade delituosa. Com os resultados apresentados neste trabalho é possível o desenvolvimento de ferramentas para automatizar a tarefa de captura de dados do eMule, o que pode facilitar o procedimento pericial de comprovação de crimes cometidos através de compartilhamento de arquivos via rede P2P.**

Palavras-chave — eMule; KAD; eD2k; perícia em informática; informática forense; ponto-a-ponto; P2P

1. Introdução

O eMule é um aplicativo *peer-to-peer* (P2P) de compartilhamento de arquivos criado para ser executado em sistemas operacionais Microsoft Windows. Seu desenvolvimento foi iniciado em maio de 2002, por Hendrik Breitkreuz (também conhecido como Merkur), como uma alternativa ao programa eDonkey. Embora tenha sido baseado no eDonkey, foram implementadas tantas funcionalidades que é possível afirmar que se trata de um sistema totalmente novo [1].

O eDonkey, um aplicativo P2P criado pela empresa MetaMachine, foi lançado em 2000 e possuía algumas vantagens sobre os programas P2P da época, como o Napster. Para o desenvolvimento do eMule, foi realizado um processo de engenharia reversa no protocolo de rede utilizado pelo eDonkey, pois este protocolo era fechado e de propriedade da empresa MetaMachine. Em 2006, a empresa MetaMachine, em decorrência de uma ação judicial, concordou em parar de dar suporte à rede eD2k, utilizada pelo eDonkey, e pagar à *Recording Industry Association of America* (RIAA), entidade que representa grande parte da indústria fonográfica dos Estados Unidos, trinta milhões de dólares por infrações de direitos autorais.

O aplicativo utilizado para gerenciar a maioria dos servidores do eMule é o *eServer* . Este aplicativo, distribuído de forma gratuita, substituiu a versão desenvolvida pela empresa MetaMachine. O *eServer* possui versões compatíveis com diversos sistemas operacionais e diferentes arquiteturas. Atualmente, o sítio oficial do *eServer* [1] não está ativo.

A primeira versão do eMule (0.02) foi disponibilizada em julho de 2002, mas ainda não era operacional. A partir da versão 0.05a, lançada em agosto de 2002, este aplicativo passou a apresentar as funcionalidades necessárias para a troca de arquivos através de redes P2P.

Atualmente, o eMule é um *software* livre, distribuído sob a *GNU General Public License*, e seu código-fonte é disponibilizado no sítio oficial do aplicativo [2].

O código-fonte do eMule, escrito em Microsoft Visual C++, foi utilizado para o desenvolvimento de diversos outros aplicativos como o DreaMule, aMule, jMule, dentre outros.

A versão atual do eMule (0.50a), lançada em abril de 2010, apresenta diversas características, como o acesso às redes eD2k e Kademlia (também conhecida como KAD), compactação do conteúdo dos pacotes transferidos, sistema de créditos para recompensar os clientes que disponibilizam arquivos, utilização de técnicas de detecção de erros na transferência de arquivos e a possibilidade de recebimento simultâneo de partes do mesmo arquivo de clientes diferentes (*swarming*) [2].

O restante deste artigo será apresentado da seguinte forma: na Seção II discutimos aspectos de utilização do eMule; na Seção III apresentamos o funcionamento do eMule; na Seção IV descrevemos como é realizado o armazenamento de dados no eMule; e finalmente, na Seção V, apresentamos as conclusões e trabalhos futuros.

2. Utilização do emule

Até o início de 2011, haviam sido realizados mais de 500 milhões de downloads do eMule, segundo a lista do *SourceForge Top Downloads* [3], tornando-o um dos aplicativos P2P mais utilizados no mundo. O eMule foi responsável por aproximadamente 17% do tráfego P2P na América do Sul em 2008/2009, tendo milhões de usuários conectados diariamente, compartilhando aproximadamente um bilhão de arquivos [3].

O eMule foi o aplicativo P2P mais utilizado no Brasil em 2008, estando presente em 17,2% dos computadores [4].

[1] http://lugdunum2k.free.fr/kiten.html
[2] http://www.emule-project.net/
[3] http://sourceforge.net/top/topalltime.php?type=downloads

A utilização de aplicativos P2P para o compartilhamento de arquivos pode ser legítima como, por exemplo, a distribuição de *software* livre. Entretanto, também é possível que o emprego desse tipo de aplicativo seja ilícito, como no caso de distribuição de material envolvendo a exploração sexual de crianças e adolescentes. De acordo com [5, 6]with important consequences for child protection, policy making, and internet regulation. However, cur- rent knowledge of this activity remains very limited. We consider here a widely used P2P system, eDonkey, and focus on two key statistics: the fraction of paedophile queries entered in the system and the fraction of users who entered such queries. We collect hundreds of millions of queries, design and assess a paedophile query detection tool, and use it to estimate these statistics. We conclude that approximately 0.25% of queries are paedophile, and that more than 0.2% of users enter such queries. These statistics are by far the most precise and reliable ever obtained in this domain.", "author" : [{ "family" : "Latapy", "given" : "Matthieu" }, { "family" : "Magnien", "given" : "C." }, { "family" : "Fournier", "given" : "R." }], "container-title" : "IEEE Infocom Mini-Conference", "id" : "ITEM-1", "issued" : { "date-parts" : [["2011"]] }, "note" : "<m:note/>", "page" : "1-24", "title" : "Quantifying paedophile activity in a large P2P system", "type" : "paper-conference" }, "uris" : ["http://www.mendeley.com/documents/?uuid=91e18400-fcda-4819-843e-d841f3a7d492"] }, { "id" : "ITEM-2", "itemData" : { "DOI" : "10.1016/S1353-4858(10, o eMule é um dos principais meios de divulgação e transmissão de arquivos contendo pornografia infantojuvenil, uma vez que foram encontrados mais de 700.000 arquivos contendo pornografia infantojuvenil compartilhados com a utilização desse aplicativo.

Certamente, a ocorrência desse tipo de delito aumentou com a disseminação do uso da Internet, pois aplicativos P2P garantem a transferência de informações de forma fácil e eficiente, auxiliando as atividades criminosas [6].

A experiência brasileira não é diferente. Diversas organizações como a Ordem dos Advogados do Brasil (OAB) e a SaferNet, dentre outras, e entidades estatais, como o Ministério Público e a Polícia Federal, estão engajadas no combate ao compartilhamento de material envolvendo a exploração sexual de crianças e adolescentes. Para atingir este objetivo, são realizadas investigações que resultam em operações policiais com o cumprimento de dezenas (eventualmente centenas) de mandados de busca e apreensão [7].

Os artefatos periciais relacionados ao eMule (versão 0.50a), apresentados neste trabalho, foram identificados através da análise do código fonte do aplicativo, de exames no comportamento do aplicativo e de consultas à bibliografia relacionada ao tema. Durante nossa pesquisa, foi encontrado o estudo apresentado em [8], que descreve uma análise sobre alguns arquivos utilizados pelo eMule. Entretanto, o presente trabalho aprofunda o referido estudo, abrangendo um número maior de arquivos e, portanto, acreditamos que seja útil o compartilhamento dos resultados com a comunidade da área. O resultado do presente estudo também foi aplicado em ocorrências reais de perícias relacionadas ao compartilhamento e disponibilização de pornografia infantojuvenil através de redes P2P, para a verificação de uso do aplicativo no contexto específico.

3. Funcionamento do eMule

Nesta seção serão apresentados os conceitos básicos relacionados ao funcionamento do eMule.

A. Identificação do usuário do eMule (User ID)

Durante a primeira execução do eMule, é criado um identificador do usuário denominado *User ID*. Esse identificador, também chamado de *User Hash*, possui 16 bytes de tamanho e é composto de caracteres aleatórios, com exceção do 6º e o 15º bytes que são, respectivamente, 14 (0x0E) [4] e 111 (0x6F).

B. Identificação da conexão do cliente do eMule (Client ID)

O identificador do cliente (*Client ID*) possui quatro bytes e é fornecido, em cada sessão, pelo servidor ao qual o cliente do eMule se conecta. O *Client ID* é válido apenas enquanto for mantida a conexão TCP com o servidor do eMule, ao contrário do *User ID*, que é permanente. Sua função é indicar se a comunicação entre clientes pode ser realizada de modo adequado. Para tanto, o servidor tenta criar uma conexão TCP com o cliente.

O *Client ID* pode ser dividido em identificação alta (*High ID*) ou identificação baixa (*Low ID*). Um cliente recebe um *High ID* se ele consegue receber conexões TCP de entrada (*incoming connections*). Se o servidor não consegue estabelecer uma conexão TCP com o cliente, é atribuído um *Low ID*, criando várias restrições à utilização do eMule, pois esse cliente não poderá ser contatado por outros clientes para a transferência de arquivos. Dessa forma, clientes com *Low ID* somente poderão receber dados de clientes com *High ID*, pois um cliente com *Low ID* não conseguirá estabelecer uma conexão TCP para transferência de arquivos com outro cliente com *Low ID*.

Caso o cliente possa aceitar conexões TCP, a forma de cálculo do *Client ID*, dado um endereço IP *A.B.C.D*, é a seguinte: $A + (2^8 * B) + (2^{16} * C) + (2^{24} * D)$.

Por ser baseado no endereço IP e calculado de forma determinística, o *High ID* mantém-se o mesmo, caso o cliente possua o mesmo endereço IP na próxima vez em que se conectar, independentemente de qual servidor do eMule for utilizado.

O *Low ID*, por sua vez, sempre será um número menor que 16777216 (0x1000000), e é atribuído de forma sequencial pelo

[4] A notação 0x indica um valor hexadecimal.

servidor. Assim, na próxima vez que este cliente se conectar a um servidor, provavelmente irá receber outro *Client ID*, mesmo que mantenha o mesmo endereço IP.

C. Identificação dos arquivos (File ID)

Para identificação dos arquivos, o eMule utiliza um código *hash* baseado no conteúdo do arquivo, e não em seu nome, como era feito pelo Napster. Essa mudança facilita a identificação de alterações nos nomes dos arquivos realizadas por usuários mal intencionados, pois o mesmo identificador apontaria para diversos nomes diferentes.

Cada identificador de arquivos possui 16 bytes e é baseado no algoritmo MD4. Os arquivos são divididos em pedaços (*chunks*) de 9728000 bytes (9500KB ou 9,28MB) cada (caso o tamanho não seja um múltiplo exato, o último pedaço é menor que 9,28MB). Se o tamanho do arquivo for menor que 9,28MB, o identificador é o próprio *hash* MD4 do arquivo. Caso seja maior, é calculado o *hash* MD4 de cada parte (*Part Hash*) e os *hashes* são concatenados em uma sequência denominada *Hashset* (também chamado de *Hash List*), sobre a qual é calculado novamente o *hash* MD4, resultando no identificador do arquivo.

D. Links do eMule (eD2k Links)

Para facilitar a distribuição de arquivos com seus identificadores, o eMule utiliza atalhos (*links*) especialmente criados para tal fim, denominados *eD2k Links*. Existem diversos formatos desses atalhos, todos iniciando com a sequência de caracteres "*eD2k://*|". Abaixo são apresentados os principais:

a) *Formato básico*: são informados o tipo do atalho (para arquivos o tipo é "*file*"), o nome do arquivo, tamanho e identificador do arquivo (*File ID*):
ed2k://|file|<nome>|<tamanho>|<identificador>|/

b) *Formato com o Hashset*: além dos itens anteriores, também é informado o *Hashset* (após os caracteres "*p*=" e com os *hashes* das partes (*chunks*) separados por dois pontos), facilitando o processo de detecção de corrupção de dados:
ed2k://|file|<nome>|<tamanho>|<identificador>|p=<Hashset>|/

c) *Formato com o Root Hash*: contém o *Root Hash* (após os caracteres "*h*="), utilizado pelo sistema de detecção de corrupção AICH, descrito adiante:
ed2k://|file|<nome>|<tamanho>|<identificador>|h=<Root_Hash>|/

d) *Formato informando as fontes (endereço IP)*: contém o endereço IP e porta de comunicação do eMule de uma ou mais fontes do arquivo, facilitando o processo de recebimento:
ed2k://|file|<nome>|<tamanho>|<identificador>|/|sources,<IP:porta,IP:porta,IP:porta,...>|/

e) *Formato informando as fontes (nome do host)*: contém o nome do host, útil para computadores que não possuem endereço IP fixo:
ed2k://|file|<nome>|<tamanho>|<identificador>|/|sources,<nome_do_host:porta,nome_do_host:porta>|/

f) *Formato informando as fontes (endereço HTTP)*: é informado um endereço HTTP que contém as fontes do arquivo (após os caracteres "*s*="):
ed2k://|file|<nome>|<tamanho>|<identificador>|s=<http://sitio_web.com.br/nome_do_arquivo>|/

g) *Formato informando servidores do eMule*: também são utilizados links eD2k para indicar servidores. Neste caso, é comunicado que o link é do tipo servidor ("|server|"), em seguida são indicados o endereço IP e a porta em que o servidor está sendo executado:
ed2k://|server|<IP>|<porta>|/

h) *Formato para executar uma busca*: este formato é utilizado principalmente em navegadores da Internet. Faz com que o eMule inicie uma busca pelo "*termo_busca*":
ed2k://|search|<termo_busca>|/

Um exemplo de atalho de um arquivo que contém o *File ID*, o *Hashset* e o *Root Hash*, é apresentado abaixo. Nele, o arquivo de nome "*nome_do_arquivo*" possui 10526543 bytes de tamanho, sendo dividido em duas partes – a primeira, com 9728000 bytes, produz como *hash* MD4 a sequência hexadecimal "BA8FF884DF69D835713ABB5842F91234", e a segunda, com 798543 bytes, produz a sequência hexadecimal "B86D0A270DD948BA40DD38D6C9151E83". O cálculo do *hash* MD4 sobre a concatenação dessas sequências hexadecimais produz o identificador do arquivo "A6A726B45ECD9C88B49116FADFDC39F4" e o *Root Hash* "4BEFWIRSL2NU5EVDL4FBZC3UDMUZZI2O".

ed2k://|file|nome_do_arquivo|10526543|A6A726B45ECD9C88B49116FADFDC39F4|p=BA8FF884DF69D835713ABB5842F91234:B86D0A270DD948BA40DD38D6C9151E83|h=4BEFWIRSL2NU5EVDL4FBZC3UDMUZZI2O|/

E. Detecção de arquivos corrompidos

O eMule emprega dois sistemas de identificação de arquivos corrompidos: (1) *Intelligent Corruption Handling* (ICH) e (2) *Advanced Intelligent Corruption Handling* (AICH).

a) *Intelligent Corruption Handling (ICH)*: sempre que o eMule recebe uma parte de um arquivo (*chunk*), é calculado o identificador dessa parte. Caso esteja correto, essa parte é disponibilizada, de forma automática, para outros clientes através da rede. Se o identificador estiver incorreto, significa que a parte está corrompida, sendo necessário baixá-la novamente. Para otimizar este processo, o eMule baixa blocos de 180KB e calcula o *hash* novamente. Esse processo continua até que o *hash* esteja correto. A vantagem é óbvia – no melhor caso, somente 180KB são baixados e, no pior caso, 9,28MB. No entanto, o ICH possui alguns problemas: (1) não consegue verificar blocos pequenos, somente partes de 9,28MB; (2) é ineficaz nos casos em que mais de um segmento da mesma parte estiverem corrompidos.

b) *Advanced Intelligent Corruption Handling (AICH)*: para solucionar os problemas do ICH, foi desenvolvido o AICH. Neste sistema, cada parte de 9,28MB é subdividida em 53 blocos (52 de 180KB e 1 de 140KB), sendo calculado sobre cada bloco, através do algoritmo SHA1, o *hash*, denominado *Block Hash*. Esse conjunto de *hashes* forma o nível mais baixo de um *Hashset* AICH. Em seguida, é calculado o *hash* de dois *Block Hashes* adjacentes, sendo gerado um *hash* de verificação num nível acima do nível atual, até chegar ao *hash* AICH da parte. Esse processo continua, calculando os *hashes* de verificação, até chegar-se a um único *hash*, denominado *Root Hash*.

O número de *hashes* de verificação por arquivo é proporcional ao número de partes (*chunks*) desse arquivo, sendo necessário calcular 108 *hashes* para cada parte de 9,28MB.

Caso esteja disponível toda a estrutura de *hashes* (*Hashset AICH*), é possível a verificação de cada bloco para identificar qual bloco está corrompido. Caso não esteja disponível o *Root Hash*, o eMule escolhe aleatoriamente 10 clientes que estão compartilhando esse arquivo e solicita o *Root Hash*. Se pelo menos 92% dos clientes enviarem o mesmo *Root Hash*, o eMule irá considerá-lo confiável e irá utilizá-lo para a identificação de corrupção do arquivo recebido. O ideal é que o *Root Hash* seja enviado no próprio *eD2k link* do arquivo.

Quando ocorre a detecção de corrupção de uma parte, o eMule envia um pedido de um pacote de recuperação (*recovery packet*) para outro cliente do eMule e que esteja compartilhando esse arquivo. A escolha é realizada aleatoriamente e é exigido que o cliente escolhido possua o *Hashset* AICH completo, ou seja, os *hashes* de todos os blocos, mais os *hashes* de verificação, até chegar ao *Root Hash*.

O pacote de recuperação contém os *hashes* dos blocos da parte solicitada, bem como os *hashes* de verificação de toda a árvore de *hashes*. Um diagrama de um pacote de recuperação para um arquivo com quatro partes é apresentada na Figura 1.

Para aceitar o pacote de recuperação, é realizada uma comparação do *Root Hash* recebido no pacote com o *Root Hash* que o eMule havia considerado confiável. Se forem idênticos, o eMule calcula os *hashes* dos 53 blocos para identificar qual deles está corrompido e seleciona apenas esse bloco para ser baixado novamente.

F. Compartilhamento de arquivos

O eMule utiliza duas pastas para o recebimento de arquivos. A primeira, denominada Pasta de Arquivos Temporários, armazena os arquivos enquanto ainda não foram totalmente recebidos. Quando os arquivos são completamente recebidos, são movidos, de forma automática, para a Pasta de Arquivos Completos.

Além de compartilhar o arquivo quando ele for completamente recebido, o eMule compartilha, de forma automática, as partes (*chunks*) do arquivo após terem sido recebidas e verificadas se não estão corrompidas. Assim, o conteúdo das pastas de Arquivos Temporários e de Arquivos Completos é compartilhado de forma automática.

Além dessas duas pastas, o usuário também pode selecionar, de forma manual, outras pastas para compartilhamento.

G. Sistema de créditos do eMule

O sistema de créditos do eMule foi criado como forma de incentivo para que os usuários compartilhem seus arquivos. A primeira versão do eMule a ter implementado esse sistema foi a 0.19a, lançada em setembro de 2002.

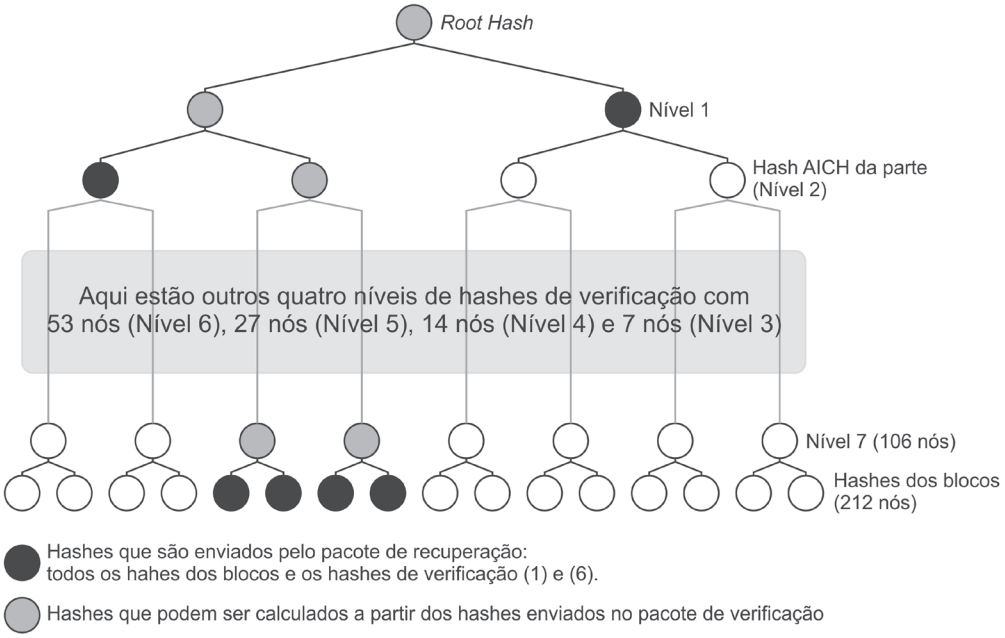


Figura 1 - Exemplo de um pacote de recuperação de um arquivo com quatro partes. Adaptado da documentação oficial do eMule (http://www.emule-project.net/).

O sistema de créditos propicia uma maior velocidade para iniciar o recebimento de arquivos, pois permite avançar na fila de espera mais rapidamente. Seu objetivo é desestimular a existência de usuários que apenas baixam arquivos sem compartilhar nada (*free riders*) [9].

O eMule divide a banda de envio de arquivos (*upload*) em *X* canais (*slots*), onde são atendidos os *X* clientes com maior pontuação. Essa pontuação é calculada através da divisão do tempo que o cliente está esperando na fila (em segundos) por 100 e a multiplicação desse resultado por um modificador (crédito).

O modificador, por sua vez, é calculado com base na quantidade de dados enviados pelo cliente que está solicitando o arquivo e quanto esse cliente recebeu de dados do usuário do eMule. Quanto maior for a taxa, mais rápido será o avanço na fila de envio desse cliente.

São calculadas duas taxas (modificadores de crédito), medidas em MB:

$$Taxa_1 = \frac{\text{Total Enviado pelo cliente} \times 2}{\text{Total Recebido pelo cliente}} \quad (1)$$

$$Taxa_2 = \sqrt[2]{\text{Total Enviado pelo cliente} + 2} \quad (2)$$

O menor valor entre $Taxa_1$ e $Taxa_2$ é utilizado como modificador da fila de envio de dados. Existem alguns limites para o modificador: (1) se o Total Enviado for menor que 1MB, o modificador é 1; (2) se foram recebidos dados desse cliente, mas nada foi enviado para ele, o modificador é 10. Se o cálculo acima resultar em um modificador menor que 1, ele é definido como 1, e se for maior que 10, é definido como 10.

O *User ID* é um elemento essencial no sistema de créditos do eMule. Para evitar fraudes, é empregado um sistema de autenticação baseado em chaves pública e privada.

Os créditos dos clientes são armazenados localmente, não sendo propagados para outros usuários do eMule. Dessa forma, os próprios créditos não ficam registrados localmente, mas sim nos outros clientes do eMule. Se o arquivo que armazena os créditos for apagado, todos os clientes que enviaram dados perdem o respectivo crédito.

4. Armazenamento de dados utilizado pelo emule

O eMule emprega o formato *little endian* para armazenamento das informações. Os campos de data/hora são gravados como *Unix time*, que emprega quatro bytes para registrar o número de segundos desde 01/01/1970.

O eMule utiliza diversos arquivos para armazenar configurações e dados sobre seu funcionamento. Esses arquivos, descritos em detalhes nesta seção, podem conter informações em posições fixas ou podem apresentar variações na posição em que os dados são gravados.

Para delimitar os dados nas posições variáveis, é empregado um sistema baseado na estrutura TLV (*Type*, *Length*, *Value*) para armazenar informações. De acordo com o código fonte do eMule, os tipos utilizados são apresentados na Tabela I.

Tabela I - Tipos Utilizados Pelo eMule

Nome	Código	Tipo	Tamanho
HASH	0x01	Matriz de bytes	16 bytes
STRING	0x02	Caractere	Variável
UINT8	0x09	Inteiro (8 bits)	1 byte
UINT16	0x08	Inteiro (16 bits)	2 bytes
UINT32	0x03	Inteiro (32 bits)	4 bytes
UINT64	0x0B	Inteiro (64 bits)	8 bytes
FLOAT32	0x04	Flutuante (32 bits)	4 bytes
BOOL	0x05	Booleano	1 bit
BOOLARRAY	0x06	Matriz de bytes	Variável
BLOB	0x07	Dados binários	Variável
BSOB	0x0A	Matriz de bytes	Variável

No sistema empregado pelo eMule para identificar os dados em posições variáveis, são utilizados códigos denominados *TAGS*. Esses códigos podem ser específicos para cada conjunto de dados (TAGs para servidores, para arquivos que já foram baixados, entre outros) ou gerais. TAGs podem ser identificadas por uma string ou por um código de identificação (TAG especial).

Os dados são armazenados em TAGS da seguinte forma:

i. Tipo da TAG (1 byte), conforme a Tabela I.
ii. Tamanho do nome da TAG (2 bytes) – se o tamanho for igual a 1, indica que será utilizado o código da TAG.
iii. Nome ou o código da TAG.
iv. Tamanho do conteúdo da TAG (2 bytes) se for um campo de comprimento variável. Caso contrário, será o tamanho de acordo com o tipo definido.
v. Conteúdo da TAG.

Um exemplo de uma TAG de nome de arquivo é apresentado na Figura 2.

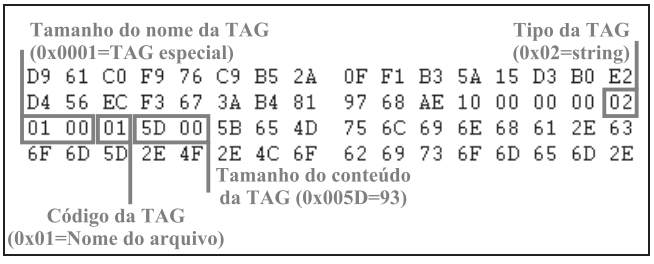


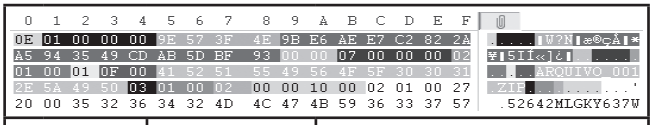
Figura 2 - Exemplo de uma TAG de nome de arquivo.

A relação dos códigos das TAGS, conforme encontrado no arquivo *Opcodes.h*, parte do código fonte da versão 0.50a do eMule, é apresentado na Tabela IV, localizada na Seção VII - Anexo.

Serão apresentados os principais arquivos do eMule que contém informações de interesse pericial.

a) *Known.met*: *é o arquivo com o maior número de informações com interesse* pericial. Aqui são gravadas informações sobre todos os arquivos que foram compartilhados ou baixados pelo eMule. Dentre essas informações, pode-se citar o nome do arquivo, o tamanho, o identificador do arquivo (*File ID*), a data e hora da última modificação, a quantidade de bytes transferida para outros usuários, entre outras informações.

O primeiro byte do arquivo *Known.met* indica a versão desse arquivo, podendo ser 14 (0x0E), que não possui suporte a TAGs de 64 bits, limitanto o tamanho máximo de arquivos compartilhados a 4 GB, ou 15 (0x0F), que suporta TAGs de 64 bits e amplia o tamanho máximo dos arquivos. Do segundo até o quinto byte, está armazenado o número de registros presentes no arquivo. Um exemplo da análise parcial de um arquivo *Known.met é* apresentado na Figura 3.



Deslocamento		Valor	Descrição
Posição fixa	0	0x0E	Cabeçalho do arquivo *Known.met* (versão)
	1 – 4	0x00000001	Número de registros presentes no arquivo *Known.met* (1)
	5 – 8	0x9E573F4E	Data da última modificação (08/08/2011 03:27:26 UTC)
	9 – 24	0x9BE6AEE7C2 822AA5943549 CDAB5DBF93	Identificador do arquivo (*File ID*)
	25 – 26	0x0000	Número de partes (*chunks*) em que é dividido o arquivo
	27 – 30	0x00000007	Número de TAGs referentes a este registro (7)
1ª Tag	31	0x02	Tipo da TAG (2=string)
	32 – 33	0x0001	Tamanho do nome da TAG (0x01=código)
	34	0x01	Nome da TAG (0x01=nome do arquivo)
	35 – 36	0x000F	Tamanho do conteúdo da TAG (0x0F=15 bytes)
	37 – 51	ARQUIVO_001.ZIP	Conteúdo da TAG Nome do Arquivo (ARQUIVO_001.ZIP)
2ª Tag	52	0x03	Tipo da TAG (0x03=inteiro 32 bits)
	53 – 54	0x0001	Tamanho do nome da TAG (0x01=código da TAG)
	55	0x02	Nome da TAG (0x02=tamanho do arquivo)
	56 – 59	0x00100000	Conteúdo da TAG Tamanho do Arquivo (1048576 bytes)

Figura 3 - Exemplo da análise parcial de um arquivo *Known.met*.

Tendo em vista que o eMule armazena no arquivo *Known.met* os arquivos que foram baixados pelo aplicativo e os que foram compartilhados pelo usuário, é necessário criar uma forma de diferenciar essas situações.

Uma forma é apresentado em [8] e consiste na busca pela TAG 0x12 (Nome do arquivo Part). Se existe essa TAG para um determinado arquivo, significa que foi criado um arquivo temporário (extensão .part) durante o recebimento pelo eMule. Entretanto, a ausência dessa TAG não significa, com certeza, que o arquivo não tenha sido obtido através do eMule, pois o arquivo *Known.met* pode ter sido apagado ou se corrompido e o eMule, ao recriá-lo, irá adicionar todos os arquivos que estão nas pastas compartilhadas, sem a TAG 0x12.

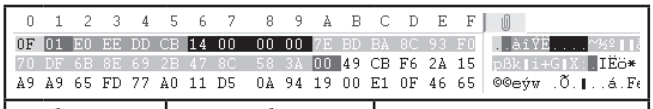
Caso a opção "Lembrar os arquivos baixados" (*Remember downloaded files*) seja desmarcada, o eMule não irá atualizar o arquivo *Known.met*.

b) *Known2_64.met*: os *hashes* AICH dos arquivos que estão sendo compartilhados são armazenados aqui. A atualização deste arquivo também pode ser desabilitada na opção "Lembrar os arquivos baixados".

c) *Cancelled.met*: neste arquivo são registrados os arquivos que estavam sendo baixados e, antes de terem sido completamente recebidos, foram cancelados. A atualização desse arquivo pode ser desabilitada pelo usuário, caso a opção "Lembrar os arquivos cancelados" (*Remember cancelled files*) seja desabilitada.

Até a versão 0.48a BETA1 do eMule, lançada em 29/04/2007, era gravado o identificador (*File ID*) do arquivo. A partir dessa versão, para aumentar a privacidade, o eMule passou a gravar o hash do *File ID*, dificultando, dessa forma, a identificação dos arquivos cancelados.

O cabeçalho atual (que mantém apenas o hash do *File ID*) é 0x0F. O cabeçalho do arquivo que mantém o *File ID* ao invés do hash é 0x0E. A estrutura atual desse arquivo é apresentada na Figura 4:



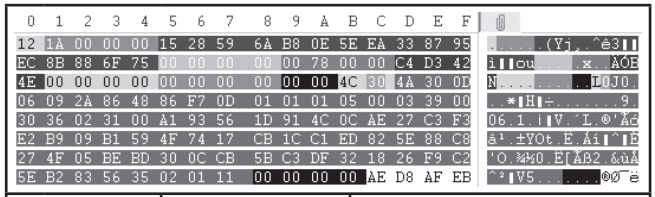
Deslocamento		Valor	Descrição
Posição fixa	0	0x0F	Cabeçalho do arquivo *Cancelled.met*
	1	0x01	É a versão do arquivo
	2 – 5	0XCBDDEEE0	É a semente (*seed*) para evitar o uso de tabelas de hash pré-calculadas (3420319456)
	6 – 9	0x00000014	Número de registros presentes no arquivo (20)
1º Registro	10 – 25	0x7EBDBA8C93 F070DF6B8E69 2B478C583A	Hash do identificador do arquivo (File ID) que foi cancelado
	26	0x00	Tag (atualmente não são utilizadas, apresentando sempre o valor 0x00)

Figura 4 - Exemplo da análise parcial de um arquivo *Cancelled.met*.

d) *Clients.met* e *clients.met.bak*: informações de outros clientes com quem o usuário teve contato são armazenados nestes arquivos. O arquivo com extensão *.bak* é uma cópia de segurança do arquivo *Clients.met*. Dentre as informações armazenadas, pode-se citar o identificador do cliente (*User ID*), a quantidade de bytes enviados e recebidos do cliente, a chave pública

do cliente (para fins de autenticação) e data e hora da última vez em que foi visto. A estrutura desse arquivo é apresentada na Figura 5.

Como o identificador do cliente (*User ID*), armazenado no arquivo *Clients.met*, *não guarda relação com o endereço IP utilizado pelo cliente, a utilidade desta informação é limitada.* Entretanto, é possível a realização de pesquisas na rede eD2k tendo por objetivo a localização do cliente. Caso o cliente seja encontrado, pode-se identificar o respectivo *Client ID*, que é baseado no endereço IP (se a conexão for *High ID*) e que pode, por sua vez, possibilitar a localização *física do* computador onde está sendo executado o cliente do eMule.



Deslocamento		Valor	Descrição
Posição fixa	0	0x12	Cabeçalho do arquivo *Clients.met* (0x12=possui suporte para o sistema de créditos; 0x11 não possui suporte para o sistema de créditos)
	1 – 4	0x0000001A	Número de clientes no arquivo (26)
1º Cliente	5 – 20	0x1528596AB80E 5EEA338795EC8B 886F75	Identificador do cliente (*User ID*)
	21 – 24	0x00000000	Quantidade de bytes enviados para o cliente (0)
	25 – 28	0x00007800	Quantidade de bytes recebidos do cliente (30720)
	29 – 32	0xC4D3424E	Data e hora em que o cliente foi visto pela última vez (10/08/2011 18:53:56)
	33 – 36	0x00000000	UploadHi (utilizado para armazenar 64 bits de dados enviados ao usuário)
	37 – 40	0x00000000	DownloadHi (utilizado para armazenar 64 bits de dados recebidos do usuário)
	41 – 42	0x0000	Reservado
	43	0x4C	Tamanho da chave pública do cliente (76). O máximo é 80.
	44 – 119	0x 304A300D[...]	Chave pública do cliente
	120	0x00000000	Preenchimento até chegar a 80

Figura 5 - Exemplo da análise parcial de um arquivo *Clients.met*.

e) *Downloads.txt* e *Downloads.bak*: no arquivo *downloads.txt* estão armazenadas a data de utilização e a pasta de Arquivos Temporários. Além disso, também são registrados todos os nomes dos arquivos temporários com extensão *.part* (utilizado durante o recebimento de dados) e os respectivos links eD2k. O arquivo com extensão *.bak* é uma cópia de segurança do arquivo com extensão *.txt*. Um exemplo de arquivo *downloads.txt* é apresentado na Figura 6.

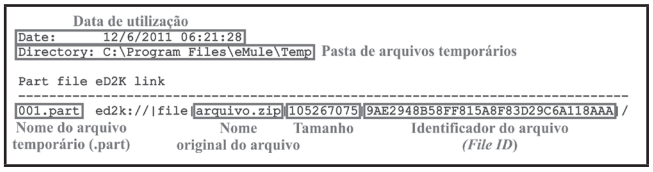


Figura 6 – Exemplo de um arquivo *downloads.txt*.

f) *Emfriends.met*: o eMule possibilita ao usuário adicionar clientes a uma lista denominada Amigos (*Friends*). Os clientes que estão presentes na lista de amigos possuem alguns privilégios como, por exemplo, ter um canal (*slot*) específico para a transferência de dados, sem a necessidade de esperar na fila geral de envio. As informações sobre os clientes presentes na lista de amigos como, por exemplo, o identificador do usuário (*User ID*), último endereço IP e porta utilizados, data e hora da última conversa (*chat*) com esse cliente, dentre outras, são armazenadas neste arquivo.

g) *Preferences.ini*: neste arquivo o eMule salva diversas configurações relacionadas ao seu funcionamento. Assim, como exemplo, a variável *AppVersion=0.50a* indica que a versão do eMule é *0.50a*. Se a configuração for do tipo Ligado/Desligado, é atribuído o valor 1 para ligado e 0 para desligado. A Tabela II apresenta algumas das informações relevantes encontradas neste arquivo.

Tabela II – Exemplos de Informações relevantes encontradas no arquivo *Preferences.ini*

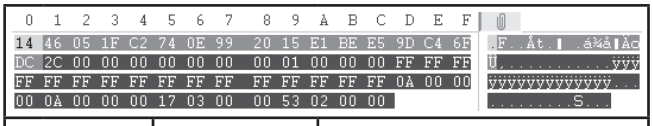
Variável	Descrição
AppVersion	Versão do eMule
IncomingDir	Pasta de arquivos completos (local para onde são movidos os arquivos quando forem completamente recebidos)
TempDir	Pasta de arquivos temporários (local de armazenamento dos arquivos durante o recebimento)
AutoStart *	Se habilitado, o eMule será executado juntamente com a inicialização do sistema operacional
Autoconnect *	Se habilitado, o eMule irá conectar-se de forma automática
Reconnect *	Se habilitado, o eMule irá se reconectar de forma automática
UseAutocompletion *	Se habilitado, o eMule irá registrar no arquivo A*C_SearchStrings.dat* os termos utilizados para a busca por arquivos
RememberCancelledFiles *	Se habilitado, o arquivo *Cancelled.met* será atualizado quando arquivos forem cancelados
RememberDownloadedFiles *	Se habilitado, o arquivo *Known.met* será atualizado para os arquivos que forem compartilhados

* Indicam variáveis do tipo ligado/desligado (0=Opção desabilitada; 1=Opção habilitada)

h) *Ipfilter.dat*: este arquivo *é utilizado como* um filtro de conexões *não desejadas*. As requisições de clientes cujos endereços IP estão relacionados neste arquivo não serão respondidas pelo eMule, nem serão permitidas conexões para estes clientes. O eMule permite que seja informada uma URL para a atualização automática

desse arquivo. Dentre os IPs bloqueados encontram-se, normalmente, endereços de órgãos relacionados à segurança pública e de entidades privadas, como a Polícia Federal e a Associação de Defesa da Propriedade Intelectual. Em cada linha deste arquivo é informado o endereço IP (ou conjunto de endereços IP) e uma breve identificação desses endereços.

i) *Preferences.dat*: o identificador do usuário (*User ID*) e informações sobre o posicionamento e tamanho das janelas do eMule são armazenados neste arquivo. A Figura 7 apresenta a análise de um arquivo *Preferences.dat*.



Deslocamento		Valor	Descrição
Posição fixa	0	0x14	Versão do arquivo *Preferences.dat*
	1 – 16	0x46051FC274 0E992015E1BE E59DC46FDC	Identificador do usuário (*User ID*)
	17 – 60	0x2C00[...]	Configurações sobre o posicionamento das janelas do eMule

Figura 7 – Exemplo da análise de um arquivo *Preferences.dat*.

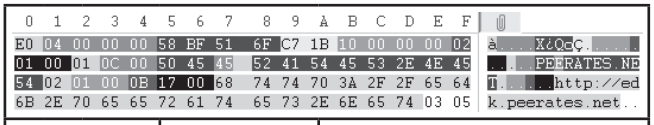
j) *Sharedir.dat*: os nomes das pastas selecionadas pelo usuário para serem compartilhadas são armazenados neste arquivo, em formato Unicode. A ordem em que as pastas compartilhadas aparecem no arquivo não guarda relação com o momento em que foram compartilhadas, pois o Emule ordena as pastas de forma alfabética.

k) *Server.met*: a relação dos servidores utilizados pelo eMule encontra-se neste arquivo. A relevância pericial está na identificação do país em que está localizado o servidor, pois se houver o envio de material ilícito ao exterior, tal parâmetro poderá determinar em que esfera o fato será investigado. Um exemplo de análise desse arquivo é apresentado na Figura 8. Na Tabela V apresentada na Seção VII – Anexo encontramos os códigos das TAGs localizadas no arquivo *Server.met*.

l) *Staticservers.dat*: aqui são registrados os servidores do eMule cadastrados pelo usuário e que possuem endereço IP estático ou que possuem nomes de domínio que podem ser resolvidos por servidores DNS. É possível que o eMule seja configurado para acessar somente os servidores presentes nesta lista.

m) *Addresses.dat*: o eMule permite que sejam informadas URLs para obtenção, de forma automática, de arquivos *server.met* atualizados quando o eMule é executado. Esses endereços são armazenados neste arquivo e, embora seja possível informar mais de um, apenas o primeiro endereço que fornecer um arquivo *server.met válido será utilizado.*

n) *AC_SearchStrings.dat*: cada palavra utilizada pelo usuário na busca por arquivos *é armazenada neste arquivo*, desde que as opções "Auto-conclusão (função de histórico)" (*Auto completion (history function)*) estejam habilitadas (variável "*UseAutocompletion*" do arquivo *Preferences.ini*).



Deslocamento			Valor	Descrição
Posição fixa		0	0xE0	Cabeçalho do arquivo *Server.met* (versão)
		1 – 4	0x00000004	Número de registros presentes no arquivo *Server.met*
1º Servidor	Posição fixa	5 – 8	0x58BF516F	Endereço IP do servidor (88.191.81.111)
		9 – 10	0x1BC7	Porta utilizada para comunicação (7111)
		11 – 14	0x00000010	Número de TAGs deste registro (16)
	1ª Tag	15	0x02	Tipo da TAG (0x02=string)
		16 – 17	0x0001	Tamanho do nome da TAG (0x01=código)
		18	0x01	Nome da TAG (0x01=nome do servidor)
		19 – 20	0x000C	Tamanho do conteúdo da TAG (12 bytes)
		21 – 32	PEERATES.NET	Nome do servidor
	2ª Tag	33	0x02	Tipo da TAG (0x02=string)
		34 – 35	0x0001	Tamanho do nome da TAG (0x01=código)
		36	0x0B	Nome da TAG (0x0B=descrição)
		37 – 38	0x0017	Tamanho do conteúdo da TAG (23 bytes)
		39	http://edk.peerates.net	Descrição do servidor

Figura 8 - Exemplo da análise parcial de um arquivo *Server.met*.

o) *AC_ServerMetURLs.dat*: o eMule permite que sejam informadas URLs para obtenção de arquivos *Server.met* atualizados. Os endereços digitados pelo usuário para a obtenção de arquivos *Server.met são armazenados no arquivo AC_ServerMetURLs.dat*.

p) *Storedsearches.met*: arquivo utilizado para o armazenamento do resultado de buscas. *É* atualizado caso *a opção* "Lembrar pesquisas abertas entre sessões" (*Remember open searches between restarts*) esteja habilitada.

q) *Cryptkey.dat*: armazena a chave privada RSA de 384 bits do usu*ário. É utilizada na autenticação com outros usuários* no sistema de créditos do eMule, para evitar que um cliente malicioso tente aproveitar-se de créditos alheios.

r) *Statistics.ini*: *é o local onde* as estatísticas de funcionamento do eMule são armazenadas. A Tabela III apresenta algumas informações que podem ser encontradas neste arquivo.

Tabela III – Exemplos de informações relevantes presentes no arquivo *Statistics.ini*

Variável	Descrição
TotalDownloadedBytes	Quantidade total de bytes recebidos (*download*)

Variável	Descrição
TotalUploadedBytes	Quantidade total de bytes enviados (*upload*)
DownCompletedFiles	Quantidade de arquivos que foram inteiramente recebidos
ConnRunTime	Tempo (em segundos) em que o cliente do eMule ficou conectado
ConnUploadTime	Tempo (em segundos) em que o cliente do eMule enviou dados (*upload*)
ConnDownloadTime	Tempo (em segundos) em que o cliente do eMule recebeu dados (*download*)
statsDateTimeLastReset	Data e hora (em formato *Unix time*) da última vez em que as estatísticas de uso foram reiniciadas

s) *Arquivos com extensão .part*: os arquivos, enquanto estão sendo recebidos pelo eMule, ficam armazenados na pasta de Arquivos Temporários. Os nomes dos arquivos são substituídos por uma sequência numérica e possuem extensão *.part*. Quando o recebimento do arquivo é concluído, ele é renomeado para o nome original, movido para a pasta de arquivos completos e seu conteúdo é compartilhado. Quando uma parte do arquivo é recebida, o eMule calcula o identificador da parte e verifica se está corrompida. Caso o identificador esteja correto, a parte é compartilhada de forma automática. O compartilhamento de arquivos que estão sendo recebidos não pode ser desabilitado [5].

t) *Arquivos com extensão .part.met* e *.part.met.bak*: armazenam as informações relacionadas à transferência do arquivo com extensão .part (descrito acima). O arquivo *.part.met.bak* é uma cópia de segurança, criada pelo eMule, do arquivo *.part.met*. A análise de um exemplo desse arquivo é apresentada na Figura 9. Na Tabela VI, apresentada na Seção VII – Anexo, encontramos os códigos das TAGs dos arquivos com extensão *.part.met*.

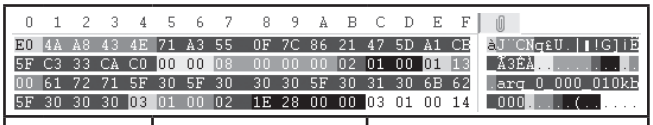
u) *src_index.dat, preferencesKad.dat, nodes.dat, key_index.dat, load_index.dat*: as informações relacionadas à rede KAD são armazenadas nestes arquivos. A rede KAD funciona de forma descentralizada e pode ser acessada concomitantemente com a rede eD2k. A principal utilização dessa rede é a busca de fontes de arquivos.

5. CONCLUSÃO

Como apresentado em [10], a utilização de técnicas periciais problemáticas pode levar a conclusões incorretas. Dessa forma, é necessário que os peritos criminais tenham grande conhecimento sobre os artefatos deixados pelos programas que estão sendo analisados.

Erros na avaliação podem acarretar desperdício de recursos e na redução da confiança da sociedade no trabalho policial [11].

[5] Perguntas e respostas do eMule. Disponível em: http://emule-project.net/home/perl/help.cgi?l=1&rm=show_to pic&topic_id=311#unshare. Acesso em: 28/jul/2011.



	Deslocamento	Valor	Descrição
Posição Fixa	0	0xE0	Cabeçalho do arquivo com extensão *.part.met* (0xE0=normal; 0xE2=arquivos com mais de 4GB)
	1 – 4	0x4E43A84A	Data da última gravação (11/08/2011 10:0:42 UTC)
	5 – 20	0x71A3550F7C86 21475DA1CB5FC3 33CAC0	Identificador do arquivo (*File ID*)
	21 – 22	0x0000	Número de partes (*chunks*)
	23 – 26	0x00000008	Número de TAGS (8)
1ª Tag	27	0x02	Tipo da TAG (0x02=string)
	28 – 29	0x0001	Tamanho do nome da TAG (0x01=código)
	30	0x01	Nome da TAG (0x01=nome do arquivo)
	31 – 32	0x0013	Tamanho do conteúdo da TAG (19 bytes)
	33 – 51	arq_0_000_010kb_000	Nome do arquivo
2ª Tag	52	0x03	Tipo da TAG (0x03=int32)
	53 – 54	0x0001	Tamanho do nome da TAG (0x01=código)
	55	0x02	Nome da TAG (0x02=tamanho do arquivo)
	56 – 59	0x000028E1	Tamanho do arquivo

Figura 9 - Exemplo da análise parcial de um arquivo com extensão *.part.met*.

Este trabalho apresentou alguns dos principais vestígios deixados pelo aplicativo P2P eMule. A identificação, localização e interpretação correta dos dados obtidos pela perícia são de fundamental importância para a determinação de autoria e materialidade criminosa.

Com as informações apresentadas no presente artigo, é possível o desenvolvimento de ferramentas que possam automatizar a tarefa de obtenção de dados relacionados ao eMule. Também pode servir como ponto de início para uma abordagem em tempo real (*live Forensics*). Outra possibilidade seria a localização de artefatos com relevância pericial em fluxos de rede, possivelmente com o emprego de técnicas de Inteligência Artificial para a identificação dos pacotes relativos ao aplicativo eMule, tendo em vista a grande quantidade de informações que são transferidas nas atuais conexões de rede.

Referências

[1] L. Caviglione and F. Davoli, "Traffic volume analysis of a nation-wide eMule community," *Computer Communications*, vol. 31, no. 10. pp. 2485-2495, Jun-2008.

[2] L. Sheng, X. Dong, J. Song, and K. Xie, "Traffic Locality in the eMule System," *2010 First International Conference on Networking and Distributed Computing*, pp. 387-391, Oct. 2010.

[3] H. Schulze and K. Mochalski, "Internet Study 2008/2009," *IPOQUE Report*. 2009.

[4] J. R. S. de Oliveira and E. E. da Silva, "EspiaMule e Wyoming ToolKit: Ferramentas de Repressão à Exploração Sexual Infanto-Juvenil em

Redes Peer-to-Peer," in *The Fourth International Conference on Forensic Computer Science*, 2009, pp. 108-113.
[5] M. Latapy, C. Magnien, and R. Fournier, "Quantifying paedophile activity in a large P2P system," in *IEEE Infocom Mini-Conference*, 2011, pp. 1-24.
[6] M. Taylor, J. Haggerty, D. Gresty, and P. Fergus, "Forensic investigation of peer-to-peer networks," *Network Security*, vol. 2010, no. 9, pp. 12-15, Sep. 2010.
[7] P. Fagundes, "Fighting Internet Child Pornography - The Brazilian Experience," *The Police Chief*, vol. LXXVI, no. 9, 2009.
[8] S. Christensen, *An IT-Forensic Examination of P2P Clients*. 2009, p. 145.
[9] Y. Li and D. Gruenbacher, "Analysis of P2P file sharing network's credit system for fairness management," in *2010 IEEE Network Operations and Management Symposium - NOMS 2010*, 2010, pp. 88-95.
[10] M. Piatek, T. Kohno, and A. Krishnamurthy, "Challenges and directions for monitoring P2P file sharing networks -or- why my printer received a DMCA takedown notice," in *Proceedings of the 3rd conference on Hot topics in security*, 2008, pp. 1–7.
[11] M. Liberatore, R. Erdely, T. Kerle, B. N. Levine, and C. Shields, "Forensic Investigation of Peer-to-Peer File Sharing Networks," *Computer*, no. October, pp. 1-11, 2009.

ANEXO

Tabela IV – Principais TAGs Encontradas no Arquivo *Known.Met*

Principais TAGs encontradas no arquivo *KNOWN.MET*		
Nome e código da TAG	**Tipo**	**Descrição**
Filename (0x01)	string	Nome do arquivo
Filesize (0x02)	uint32 uint64	Tamanho do arquivo. Se for utilizado o tipo de 32 bits, o limite de tamanho de arquivo é 4 GB. O tipo de 64 bits suporta arquivos maiores de 4 GB.
Filetype (0x03)	string	Tipo do arquivo. Valores possíveis: "Audio", "Video", "Image", "Doc" e "Pro".
Fileformat (0x04)	string	Formato do arquivo
Lastseencomplete (0x05)	uint32	Última vez que foi visto completo (*Unix time*)
Transferred (0x08)	uint32 uint64	Quantidade de bytes que foi transferido. Se for utilizado o tipo de 32 bits, o limite de tamanho de arquivo é 4 GB. O tipo de 64 bits suporta arquivos maiores de 4 GB.
Partfilename (0x12)	string	Nome do arquivo .part (nome temporário utilizado enquanto é realizado o recebimento do arquivo). Só existe para arquivos que foram recebidos através do eMule.
Dlpriority (0x18)	uint8	Prioridade de *download*. Valores possíveis: 0=Baixa; 1=Normal; 2=Alta; 3=Lançamento; 4=Muito Baixa; 5=Automática
Ulpriority (0x19)	uint8	Prioridade de *upload*. Valores possíveis: 0=Baixa; 1=Normal; 2=Alta; 3=Release; 4=Muito Baixa; 5=Automática
Kadlastpublishsrc (0x21)	uint32	Data e hora em que foi compartilhado pela última vez na rede KAD
Dl_active_time (0x23)	uint32	Tempo de *download* (em segundos)
Aich_hash (0x27)	hash	Hash AICH
Filehash (0x28)	hash	Identificador do arquivo (*File ID*)

Principais TAGs encontradas no arquivo *KNOWN.MET*		
Nome e código da TAG	**Tipo**	**Descrição**
Lastshared (0x34)	uint32	Última vez que o arquivo foi compartilhado
Aichhashset (0x35)	uint32	Hashset AICH
Filesize_hi (0x3a)	uint32	Utilizada em conjunto com a TAG de tamanho do arquivo de 32 bits para representar o tamanho de arquivos em 64 bits. Empregada apenas quando o cliente não suportar Tags de 64 bits.
Attransferred (0x50)	uint32	Quantidade de bytes enviados (*upload*)
Atrequested (0x51)	uint32	Número de pedidos de transferência de arquivos
Ataccepted (0x52)	uint32	Número de pedidos aceitos de transferência de arquivos
Category (0x53)	uint32	Categoria
Attransferredhi (0x54)	uint32	Utilizada em conjunto com a TAG *Attransferred* para representar o tempo de transferência de arquivos.
Maxsources (0x55)	uint32	Número máximo de fontes (pode ser configurado pelo usuário)
Media_artist (0xd0)	string	Nome do artista da mídia
Media_album (0xd1)	string	Nome do álbum da mídia
Media_title (0xd2)	string	Título da mídia
Media_length (0xd3)	uint32	Tempo de duração da mídia (em segundos)
Media_bitrate (0xd4)	uint32	*Bitrate* da mídia
Media_codec (0xd5)	string	Codec da mídia

Tabela V –TAGs Encontradas no Arquivo *Server.Met*

TAGs do arquivo *SERVER.MET*		
Nome e código da TAG	**Tipo**	**Descrição**
Servername (0x01)	string	Nome do servidor
Description (0x0b)	string	Breve descrição do servidor
Ping (0x0c)	uint32	Tempo (em milissegundos) necessário para se comunicar com o servidor (ping)
Fail (0x0d)	uint32	Número de vezes em que não foi possível comunicar-se com o servidor
Preference (0x0e)	uint32	Prioridade dada a este servidor (0=Normal, 1=Alta, 2=Baixa)
Port (0x0f)	uint32	Porta principal utilizada
Ip (0x10)	uint32	Endereço IP do servidor
Dynip (0x85)	string	Nome de domínio do servidor
Maxusers (0x87)	uint32	Número máximo de usuários simultâneos que o servidor suporta
Softfiles (0x88)	uint32	Número de "*soft files*"

TAGs do arquivo *SERVER.MET*		
Nome e código da TAG	**Tipo**	**Descrição**
Hardfiles (0x89)	uint32	Número de "*hard files*"
Lastping (0x90)	uint32	Última vez que o servidor respondeu ao pacote *ping*
Version (0x91)	string uint32	Versão ou nome do servidor (pode ser numérico ou sequência de caracteres)
Udpflags (0x92)	uint32	Informa se o servidor aceita conexões através do protocolo UDP
Auxportslist (0x93)	string	Portas adicionais (além da porta padrão) utilizadas pelo servidor para comunicação com os clientes do eMule
Lowidusers (0x94)	uint32	Número de usuários conectados com *Low ID*
Udpkey (0x95)	uint32	Utilizado para ofuscar a comunicação através de pacotes UDP
Udpkeyip (0x96)	uint32	Utilizado para ofuscar a comunicação através de pacotes UDP
Tcpportobfuscation (0x97)	uint16	Porta TCP utilizada pelo servidor para ofuscação
Udpportobfuscation (0x98)	uint16	Porta UDP utilizada pelo servidor para ofuscação

Tabela VI – Principais TAGs Encontradas nos Arquivos com Extensão *.PART.MET*

Principais TAGs encontradas nos arquivos com extensão *.PART.MET*		
Nome e código da TAG	**Tipo**	**Descrição**
Filename (0x01)	string	Nome do arquivo
Filesize (0x02)	uint32 uint64	Tamanho do arquivo. Se for utilizado o tipo de 32 bits, o limite de tamanho de arquivo é 4 GB. O tipo de 64 bits suporta arquivos maiores de 4 GB.
Filetype (0x03)	string	Tipo do arquivo. Valores possíveis: "Audio", "Video", "Image", "Doc" e "Pro".
Lastseencomplete (0x05)	uint32	Última vez que foi visto completo (*Unix time*)
Transferred (0x08)	uint32 uint64	Quantidade de bytes que foi transferido. Se for utilizado o tipo de 32 bits, o limite de tamanho de arquivo é 4 GB. O tipo de 64 bits suporta arquivos maiores de 4 GB.
Gapstart (0x09)	uint32 uint64	Início do intervalo que ainda não foi recebido. Se for utilizado o tipo de 32 bits, o limite de tamanho de arquivo é 4 GB. O tipo de 64 bits suporta arquivos maiores de 4 GB.
Gapend (0x0a)	uint32 uint64	Final do intervalo que ainda não foi recebido. Se for utilizado o tipo de 32 bits, o limite de tamanho de arquivo é 4 GB. O tipo de 64 bits suporta arquivos maiores de 4 GB.
Partfilename (0x12)	string	Nome do arquivo .part (nome temporário utilizado enquanto é realizado o recebimento do arquivo).
Status (0x14)	uint32	Status da transferência (1=pausado)

Principais TAGs encontradas nos arquivos com extensão *.PART.MET*		
Nome e código da TAG	**Tipo**	**Descrição**
Dlpriority (0x18)	uint8	Prioridade de *download*. Valores possíveis: 0=Baixa; 1=Normal; 2=Alta; 3=Release; 4=Muito Baixa; 5=Automática
Ulpriority (0x19)	uint8	Prioridade de *upload*. Valores possíveis: 0=Baixa; 1=Normal; 2=Alta; 3=Lançamento; 4=Muito Baixa; 5=Automática
Compression (0x1a)	uint32 uint64	Quantidade de bytes economizados com a compressão
Corrupted (0x1b)	uint32 uint64	Quantidade de bytes desperdiçados com a corrupção de dados
Kadlastpublishsrc (0x21)	uint32	Data e hora em que foi compartilhado pela última vez na rede KAD
Dl_active_time (0x23)	uint32	Tempo de *download* (em segundos)
Aich_hash (0x27)		Hash AICH
Attransferred (0x50)	uint32	Quantidade de bytes enviados (*upload*)
Atrequested (0x51)	uint32	Número de pedidos de transferência de arquivos
Ataccepted (0x52)	uint32	Número de pedidos aceitos de transferência de arquivos
Category (0x53)	uint32	Categoria
Attransferredhi (0x54)	uint32	Utilizada em conjunto com a TAG Attransferred para representar o tempo de transferência de arquivos.