Hash Collision Attack Vectors on the eD2k P2P Network

Uzi Tuvian Lital Porat
Interdisciplinary Center Herzliya
E-mail: {tuvian.uzi,porat.lital}[at]idc.ac.il

Abstract

In this paper we discuss the implications of MD4 collision attacks on the integrity of the eD2k P2P network. Using such attacks it is possible to generate two different files which share the same MD4 hash value and therefore the same signature in the eD2k network. Leveraging this vulnerability enables a covert attack in which a selected subset of the hosts receive malicious versions of a file while the rest of the network receives a harmless one.

We cover the trust relations that can be voided as a consequence of this attack and describe a utility developed by the authors that can be used for a rapid deployment of this technique. Additionally, we present novel attack vectors that arise from this vulnerability, and suggest modifications to the protocol that can circumvent such attacks.

1. Introduction

File sharing peer-to-peer networks have affected the evolution of internet, booming household connectivity rates and demand for ever-increasing bandwidth. File sharing networks provided accessibility to rich media content on the internet before the availability of legit online stores (like iTunes) and as such became widely popular around the globe. According to a recent research[13], up to 40 billion files were 'illegally file shared' in the year of 2008. As a hub of un-regulated, non-monitored high traffic file swapping activity, P2P networks pose an ideal candidate for distribution of malicious executables.

In recent years, efficient collision attacks have been that target the MD4 and MD5 family of hash algorithms have been discovered. These attacks enable the rapid generation of pseudo-random colliding blocks. Although not as useful as first and second preimage attacks, a collision attacks is suffice to generate colliding executables – two different executables which share the same hash value.

In this paper we present attack vectors which use such colliding executables into an elaborate attacks on users of the eD2k network which uses the MD4 hash algorithm in its generation of unique file identifiers. By voiding the 'uniqueness' of the identifiers, the attacks enable selective distribution – distributing a specially generated harmless file as a decoy to garner popularity among hosts in the network, and then leveraging this popularity to send malicious executables to a specific sub-group. Unlike conventional distribution of files over P2P networks, the attacks described give the attacker relatively high control of the targets of the attack, and even lets the attacker terminate the distribution of the malicious executable at any given stage.

While the discussed vulnerability has been known among the eMule developers community [10,11] no real research/discussion of it have been conducted and no modifications have been done to date to countermeasure it.

2. Background

The eDonkey2000 file sharing network is a decentralized peer to peer network originally designed and released by MetaMachine as proprietary client and server. The network allows search and retrieval of files, and unlike other P2P network at the time allowed multi-source downloads – downloading the same file from multiple sources and benefiting from the joint bandwidth of all available sources. In order to maintain the integrity of files in the network, a scheme incorporating a 128-bit MD4 checksum is deployed to generate a unique identifier of each file in the network[1]. This identifier, in conjunction with file size, is used to identify unique search results and as a main identifier of files in the eD2k URI scheme, mostly used in websites dedicated for file sharing.

Five years after the introduction of the eDonkey2000 network, MetaMachine discontinued support for the network after receiving a 'cease and desist' letter from the RIAA[2]. The network has since been 'taken over' by a few alternative clients and servers (most notably the open-source eMule project) that implemented the eD2k protocol using reverse engineering techniques and extended it to support new features and a server-free network structure.

3. Trust Relations in the eD2k Network

In order to locate a file in the eD2k network a user can either import an eD2k URI from an external source - usually a website, or perform a search through the client (either a server based or a distributed search). As is the case with most modern file sharing networks, a user which intends on locating a specific file must use certain techniques in order to identify true results and avoid 'false positives' such as viruses and fake results.

One such widely used technique in the eD2k network is searching and importing verified URIs from websites and community forums where members of the community share URIs to files that have already been verified. This techniques facilitates a trust relationship between the multiple members of the community – each time a different member of the community invests time and efforts into locating, validating and publishing a file s.t the risk and efforts are distributed between the members.

Another highly popular technique is using the popularity of a file as a measure to its 'validity'; The user deduces from the number of the users hosting the file whether the file is 'worth downloading' by following the hypothesis that most users would remove a fake one (once spotted). Additionally, this strategy promises the user, up to some probability, that the download would go faster than more rare files due to the multi-source nature of the network.

Both these techniques are built upon the uniqueness of the file hash, as incorporated in the URI scheme and search results, to verify that the file downloaded is indeed the one matching the filtering criteria used to identify the fitting files. The attack vectors discussed in this article will leverage these trust relations by deploying different files sharing the same MD4 hash result and, hence, the same eD2k URI.

4. Hash Collision Attacks

A collision attack on a hash function is a process which tries to locate two arbitrary inputs resulting in the same hash value. Such operation is unfeasible in an ideal hash function where, following to the 'Birthday Problem', a successful collision attack on a given hash of n bits will require up to $2^{n/2}$ hash function evaluations. By using cryptanalysis to identify weaknesses in the hash generation process, researchers are able to define efficient collision attacks that enable much faster generation of collisions.

The MD4 hash function has been first shown to be vulnerable to such collision attacks in [3] dating back to 1996. In [4] Wang et al. described an efficient collision attack against the MD4 hash function (among other functions of the same family); Their technique was later improved by Sasaki et al. and described at [5]. The results of these researches allow rapid generation of collision at a very low cost (a few microseconds of CPU time).

An implementation of Wang et al.'s efficient collision attacks on MD4 and MD5 is freely available for download as open source software from Patrick Stach's website at [6]. This utility was used as a basis for the experiments described in this article.

5. Description of the attack

The result of an MD4 hash generation is affected by two parameters : **1.** An Initialization Vector (IV) and **2.** The data block being digested. The IV is the initial value used as input for the first round of the hash generation and the data block is the data on which the algorithm iterates during the digest generation process. Since Wang et al.'s attack supports arbitrary IVs, we can build an executable which incorporates the generated colliding blocks within it (each version containing a different block), and uses the different blocks to differentiate its behavior. The hash result of the binary data preceding the colliding blocks would be used as an IV to the collision generator, and since the resulting binaries are identical apart from the generated blocks, the hash values of the complete binaries would be the same.

Another possible (and more covert) scheme would be to incorporate a cyphered block within the executable. This block will contain the hidden code of the executable and de-cyphering it will be possible either directly – using one version of the colliding block as a key, or in-directly, using the block to de-cypher a longer key (which enables a stronger encryption of the hidden code). This scheme significantly lessens the chances of the harmless executable being detected as malicious by anti-virus applications.

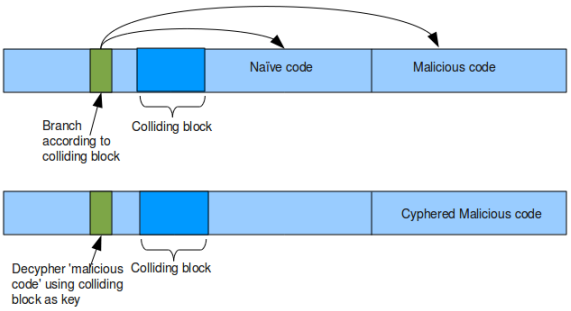

Fig 1. Possible layouts of attacking executables : branching according to colliding block version (top) and cyphering the malicious code using the colliding block as key (bottom).

6. Methodology and Results

For our experiments we have adapted the 'evilize' open source library [7] created by Peter Selinger of Dalhousie University's Department of Mathematics and Statistics to work alongside with Patrick Stach's implementation of Wang et al.'s MD4 collision attack algorithm. The resulting utility generates colliding executables along the lines of the earlier technique described above (using the colliding blocks to branch between two different functions contained in the executable). This process allows the generation of same-size, MD4-identical, differently behaving executables at a marginal cost.

Using the modified tool, two colliding executables were generated. Generation of the executable took less than a second on a Core 2 Duo processor. The eMule [8] client was then used to generate 2 URIs for the colliding files. Upon inspection, the URIs proved to be identical. Additionally, eMule recognized both files to be the same – grouping them together in the application's UI. Then, both versions of the file were put in the shared folder of a machine running the client (one version at a time) and the URI was manually entered to a client running on another machine connected to the network. Both versions were located and transferred this way across the network, either by using the server-based search feature or searching through the Kademlia distributed network. The same behavior has also been observed using the aMule [9] client.

The experiment was then repeated, using an extended version of the URI, incorporating an AICH (Advanced Intelligent Corruption Handling) field, which is an extension of the original protocol used for file corruption handling. Since AICH is based on a SHA-1 hash tree, it could offer a limited countermeasure to the attacks discussed in this article, but since the current use of AICH by the clients is limited to cases where a corruption has been detected, it did not come into affect during the experiments and had no effect on the results.

Filename	MD4 hash
good	88ede0373d0502705f09c472fed62379
evil	88ede0373d0502705f09c472fed62379
Filename	**AICH value**
good	VDFD35DNOIMNP2UZ7LX6YAH66GIKMGXB
evil	CEZGAWJKHBMEEEP2EKQDTUHPKKRM5BRA

Fig 2. MD4 (top) & AICH results (bottom) as extracted from the eD2k URIs of the generated colliding files. Although AICH values of the files differed, same MD4 was sufficient to facilitate the attack.

The results of the experiments prove that it is indeed possible to distribute differently behaving files pretending to be the same on the eD2k network. These results enable a few potential malicious attack vectors, all of which are enabled by the fact that the attacker can send different versions of the executable to different hosts.

In our experiments we have concentrated on the generation of colliding executables, but the implications of this issue is by no mean limited to executable files. Any file format that support behavior branching (either by design or as a side effect) can potentially be used to facilitate this attack. Additionally, this article refers to files of less then 9500kB. The scheme used to generate the eD2k file hash of files larger then 9500kB is not a 'straight forward' MD4 hash, but nonetheless vulnerable to the same attack using minimal modifications (which won't be discussed in this paper).

7. Attack Vectors

Most of the potential attack vectors would raise their success ratio by first distributing the harmless copy of the file in order to gain 'reputation' and get a higher 'rank' among the potential search results. Additionally, the attacker might post the URI on a community website and let the 'legit' version be verified by the community members.

After this 'seeding period' the attacker might either start distributing the malicious version and attack as many hosts as possible, or distribute different versions to different hosts according to various filtering parameters.

One such filtering parameter is the host's IP address. Using this parameter, the attack can be limited to specific countries/regions (based on legal issues,

political agenda, cyber warfare) or specific companies/organizations (sending malware, hiding illegal content distribution from known RIAA and law enforcement hosts, installing trojan horses on government computers, sending legit copies to AV companies) etc.

Another potential filtering parameter is by time: which allows the attacker to limit the distribution of the malware (i.e sending the hostile version every other day, or whenever a certain threshold is reached) in order to lower the chances of being investigated by AV companies/being noticed by network administrators or even shutting down the distribution of the malware completely upon a given date.

Other possible parameters include OS (either deduced from the client version or using OS fingerprinting techniques), prior knowledge on the specific user, the files hosted by the host (in case the feature is enabled) etc. These criteria can be combined in order to reach optimized results, and a 'dedicated' attacker might use machine learning algorithms in conjunction with a database of various hosts characteristics in order to optimize the success rates over several iterations of the attack.

8. Possible Optimizations

Due to the multi-source nature of the network, the attacking host is only required to provide the affected chunk of the file (the one containing the colliding block) in order to affect the behavior of the executable on the receiving host – thus rendering the attack highly traffic-efficient.

The multi-source technique also presents a 'race condition' to the attack where the critical chunk (the one containing the colliding block) might be distributed to a host either by the attacker, a host which already finished the download of the file or a host currently downloading the file (which already finished the transfer of the relevant chunk). This 'race condition' limits the attacker's control over the distribution of the versions, but a few measures might raise the chances of the attacker to be the source of the chunk – having fast connectivity to the network, giving top priority to the transfer of the critical chunk, allowing a large number of concurrent transfers and connections to other hosts etc. Additionally, the attacker may leverage a feature in the protocol which allows the addition of a pre-defined source to the eD2k URI. This feature may be used when the URI is distributed manually through forums or other means of URI distribution.

Another beneficial optimization can be to remove or alter the shared file once it is executed (in case it is executed on the machine running the client). This optimization can be highly efficient in cases where a limited distribution is desired and it'll significantly lower the chances of a host getting the 'limited version' by chance. Such tactic can be used in addition to giving 'non-interesting hosts' a lower priority in the attacker's transfer queue; since there are higher chances that the 'default chunk' distributed by other hosts in the network is the 'legit' one - the chances of a 'false attack' are low. This tactic can also be reversed in case a higher distribution of the malicious version is desired.

9. Potential Countermeasures

One possible countermeasure presented in the eMule developers forums [12] suggests running an AICH check upon successful download of any file (and not only in case of corruption). This countermeasure can prove effective in case a validated AICH hash is available beforehand (such as the case of a URI acquired through a website), but due to the 'swarm voting' mechanism used to determine the AICH hash in all other cases, it's possible to 'poison' the voting and send an AICH that fits the version sent to the host.

Another possible countermeasure is to change the default hashing algorithm used to a more modern algorithm such as a SHA-2 variant or SHA-3 (once it's standardized). This modification would break the backwards-compatibility of the protocol, but a gradual move might draw the impact to a minimum, and in the long-run, it seems like an inevitable move the developers would have to make.

10. Conclusions

In this paper we have examined MD4 hash collision attacks and the possible attack vectors that it presents to the eD2k peer to peer network. By publishing this paper, we hope to trigger discussion of these issues among the security community and make way to future research of hash derived weaknesses in today's network protocols.

11. Acknowledgments

The authors would like to thank Dr. Anat Bremler-Barr for her support of this paper and Mr. Yoav Steinberg for the original idea leading us to this research.

12. References

[1] Ed2k link – Amule Project FAQ. http://wiki.amule.org/index.php/Ed2k_link.

[2] Macovschi Alexandru, "eDonkey Shuts Down", Softpedia. http://news.softpedia.com/news/eDonkey-Shuts-Down-9528.shtml.

[3] Hans Dobbertin, "Cryptanalysis of MD4", Fast Software Encryption 1996: 53–69.

[4] Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, Xiuyuan Yu, "Cryptanalysis of the Hash Functions MD4 and RIPEMD", Eurocrypt 2005: 1–18.

[5] Yu Sasaki, Lei Wang, Kazuo Ohta, Noboru Kunihiro, "New Message Difference for MD4", Fast Software Encryption 2007: 329–348.

[6] Stach & Liu Research – MD5 and MD4 Collision Generators. http://www.stachliu.com/research_collisions.html.

[7] Peter Selinger – MD5 Collision Demo. http://www.mscs.dal.ca/~selinger/md5collision/.

[8] eMule-Project.net. http://www.emule-project.net/.

[9] aMule. http://www.amule.org/.

[10] Possible Danger to Ed2k Network – Official eMule-Board. http://forum.emule-project.net/index.php?showtopic=56643.

[11] Md5 Broken – Official eMule-Board. http://forum.emule-project.net/index.php?showtopic=59474.

[12] Upgrade Hashing – Official eMule-Board. http://forum.emule-project.net/index.php?showtopic=99686&st=20.

[13] IFPI Digital Music Report 2009. http://www.ifpi.org/content/library/DMR2009.pdf.