McAfee®

Protect what you value.

The McAfee Network Security Platform Strategy on Evasive Peer-to-Peer Traffic Management

A McAfee Avert® Labs publication

Table of Contents

McAfee®

Background

For years, peer-to-peer (P2P) traffic has been an increasing challenge for network administrators. Existing P2P clients and protocols like BitTorrent are becoming more popular, and newer P2P protocols are created every year. Predominantly used for piracy (downloading ripped-off movies, music, or application software), P2P traffic takes up anywhere from 50 to 90 percent of Internet traffic, according to a 2007 P2P traffic survey from ipoque.[1] If left uncontrolled, it could take up almost the entire bandwidth of most enterprise or Internet service provider (ISP) networks, eating into bandwidth availability, increasing costs, and degrading the quality of other real-time protocols like Voice over IP (VoIP). The solution thus far has been to use traffic shaping. This involves classifying network flows based on their type (HTTP, P2P, FTP) and assigning them different bandwidth limits and/or quality-of-service (QoS) treatments. There are numerous products on the market that provide traffic management using this approach. The classification of any given flow boils down to explicitly whitelisting a flow as "known good," or blacklisting a flow as "known bad." This is done predominantly using network-level signatures similar to the way anti-virus products detect malware binaries using byte patterns.

Recent Advances and Pain Points

P2P protocol developers have been working on anti-traffic-shaping measures since as early as 2005, and the techniques have gotten very robust since then. To get an idea of the level of evasiveness we are dealing with, let's look at two very popular P2P protocols—BitTorrent and Skype. (The use of Skype may be considered benign or malign depending on site policy; in this paper, the Skype protocol obfuscation is described only to illustrate the state of the art in obfuscation technology, and not to suggest that it is a harmful protocol.)

Case #1: Open-source, open-protocol

BitTorrent has a truly decentralized architecture. In the most general case, there is no central server for registrations, logins, or storing information about which nodes are sharing what files. A distributed hash table (DHT) approach allows a requesting node to look up what nodes are sharing a particular file, even as these nodes are entering or leaving the Internet. From a network-detection point of view, this is a huge issue, since there is absolutely no list of IP addresses or specific ports one can blacklist—it has to be done via an analysis of each and every network flow. Further, once a node gets the IP-port pair of a remote node that is sharing the requested file, the BitTorrent protocol supports what is called message stream encryption (MSE) or protocol header encryption (PHE) to encrypt the entire file-transfer TCP session. This involves a Diffie-Hellman key exchange that was designed to set up a shared secret between two parties communicating for the very first time in the presence of eavesdroppers. In other words, the Diffie-Hellman key exchange will look like completely random bytes (inside packets that have random lengths, and non-deterministic IP addresses and ports) to an intrusion prevention system or a traffic-shaping device. Once a shared secret key is set up between these two nodes, they proceed to use a completely open protocol, but now encrypted using this key, to start sharing the file requested. So we would have an open-source BitTorrent client like Azureus using the open-protocol BitTorrent, and a traffic-shaping device would be helpless against it.

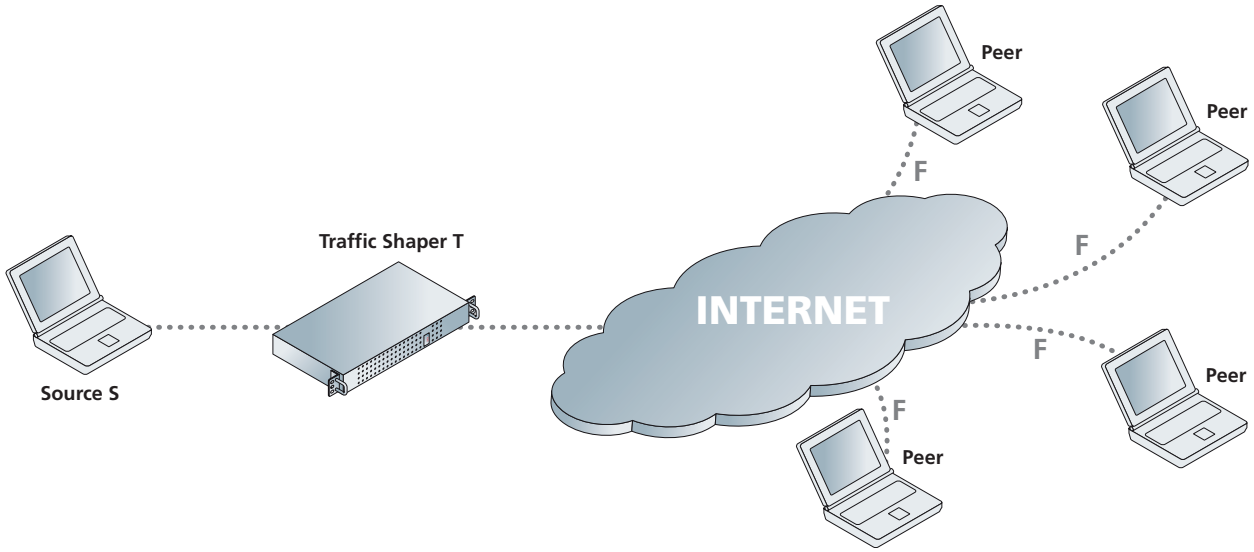

Figure 1: Typical traffic-shaping scenario for a P2P file-download from a local source

Case #2: Heavily guarded secret source and protocol
Skype's approach is at the other extreme. It is completely closed-source. It is also heavily obfuscated and armored with anti-reverse-engineering measures. Unlike BitTorrent, Skype is designed to deter the development of clones of the Skype client. This is understandable since Skype makes money from voice calls and wouldn't want clones to piggyback onto the Skype cloud and eat into the company's revenues. To achieve this, Skype clients need to have a shared secret that is hard to reverse engineer. One approach would be to use a shared secret key stored in each Skype binary, but such approaches are easiest to reverse. Instead, Skype clients use a secret algorithm to generate the key for encrypting or decrypting Skype traffic. This algorithm is then wrapped with layers upon layers of obfuscation that take many man-hours to reverse, but trivial amounts of processing time to execute. The algorithm is also upgraded every now and then with newer builds. What is the result from a network-detection point of view? First, a traffic-shaping device would need to be programmed with an algorithm to detect Skype traffic based on reverse engineering the secret Skype algorithm. Further, even if the number of man-hours spent on that effort could keep pace with the updates to the algorithm, the device would still need to perform this mathematical transpose function on each and every network flow, perhaps severely limiting its performance in an ISP or enterprise environment.

Both of these contrasting approaches are technologically brilliant. They make P2P protocols practically invisible to traffic shapers.

(**Note**: As if that weren't scary enough, we are also seeing more and more metamorphic malware wherein malware served from a web URL or a P2P node will be a totally new, recompiled version each time. So the malware has no static byte pattern that can be used by signature-based anti-virus solutions. Soon we will see such malware spreading over invisible P2P networks.)

The McAfee Network Security Platform Approach to Combat Evasive P2P Traffic

Through our P2P research here at McAfee, we are not only rapidly adapting to these P2P evasiveness measures, but also looking ahead at long-term solutions. We have engineered several patent-pending techniques that identify generic behavioral characteristics of evasive P2P file-sharing protocols. These are characteristics that stem from the very evasive, file-sharing, and distributed nature of these P2P systems. The more aggressive a P2P system's design to defeat traditional traffic shapers based on individual protocol-identification signatures, the easier it becomes for our system to detect them. Likewise, due to the sheer lack of control of such P2P systems over the characteristics the detection-algorithm identifies, it is impossible for the P2P protocols to evade the algorithm without becoming easy candidates for byte-pattern signature-based detection. We describe the key components of this groundbreaking approach to rate-limiting evasive bulk transfers in the following sections.

The process of classifying evasive P2P file-sharing can be split into three phases:

Phase I: Graylisting—Classifying the unclassified
First, we attempt to classify protocols based on "known good" (whitelists) and "known bad" (blacklists).

McAfee® Network Security Platform (*formerly McAfee IntruShield®*) comes with an industry-grade network-to-application-layer protocol classification engine. Currently, we have detection coverage for hundreds of protocols[2] including the TCP/IP stack (for example, TCP and ICMP), numerous application-level protocols (such as FTP, TFTP, and HTTP) and a large number of P2P and instant messenger protocols (for example, eDonkey, BitTorrent, and Gnutella). Further, the coverage for these protocols is kept constantly up to date, and coverage for newer protocols and variants is added on a continuous basis.

This coverage is therefore used to whitelist and blacklist network flows. For example, protocols such as FTP and SSH are "known-good" protocols, whereas BitTorrent, and Gnutella variants will be part of the "known-bad" set.

Any network flow that does not fall into these exhaustive whitelists and blacklists will be flagged as a graylisted (or unknown) protocol.

Phase II: Heuristics to zone in on bulk-data transfer flows
Now that we have a list of unknown or graylisted flows, we use some carefully crafted heuristics that identify flows as either bulk-data or not-bulk-data transfers. This way, we further drill down into the graylist to eliminate any interactive sessions or low-bandwidth flows that do not need to be traffic-shaped. Some of the heuristics we use include looking for:

- **Flows with non-trivial volumes of data transfer**—Modern, efficient P2P systems are designed so that the smallest incomplete "chunk" of a file that a server-node can have is at least a few hundred kilobytes (256 KB per chunk in BitTorrent). Hence, we focus on flows that have already transferred a certain predefined minimum amount of data (say, a few kilobytes of data) and we ignore all data-flows that only transfer low volumes of data.

- **Flows that have data-traffic and not interactive sessions**—Interactive sessions naturally have packets with tiny payloads exchanged in both directions intermittently. Data transfers, on the other hand, involve mostly a unidirectional flow of large packets with packet sizes equal to the network medium's capacity or the negotiated maximum segment size (MSS) of the session. For example, if a network flow has hundreds of packets in the server-to-client direction, and most of these packets are of, for example, 1412 bytes, then it's very likely the flow is a server-to-client bulk-data transfer and 1412 bytes is either the maximum transmission unit (MTU) of the network route, or the MSS negotiated between the two peers. Further, given the large number of packets exchanged, discovering this path MTU or MSS for the session is as easy as noting the size of the largest packet seen in that direction so far.

Figure 2 is a screenshot that illustrates this idea. It shows an Ethereal network trace of a BitTorrent file-download session. Specifically, it shows a typical 16 Kb file-chunk transfer happening as a burst of large, equal, MTU/MSS-sized packets.

- **Non-printable/binary responses**—Obfuscated or encrypted protocols naturally have non-printable (non-ASCII) characters in them. Since traditional traffic shapers typically look for signatures at the start of network flows, evasive P2P protocols typically encrypt or obfuscate bytes from the very start of the flow itself. Thus the start of network flows becomes the best place for us to look for binary characters.

Result I

Phases I and II above identify flows as graylisted/unknown, non-printable/binary, long-lasting (non-trivial data), and non-interactive/data-stream flows with a high degree of certainty, flagging them as evasive file-transfer protocols. Network Security Platform provides a protocol family called "bulkdata" for all such network flows.[3]

The first option at this point is to tag flows like these and rate-limit them with a priority lower than that given to "known-good" protocols and perhaps higher than "known-bad" protocols. For example, HTTP might be explicitly whitelisted, and the Emule P2P protocol might be explicitly blacklisted. In the rate-limiting priority scheme, such flows would get less priority than HTTP, but more than Emule under heavy network loads. In this way, the algorithm allows network administrators to provide differentiated quality to network traffic based on whether it falls in whitelisted, blacklisted, or the graylisted (previously "invisible") area.

Phase III: Multi-flow correlation of evasive bulk-data flows over time

This phase focuses on the distributed nature of the P2P system. Modern P2P systems split a file into multiple chunks. At any point in time, there could be multiple, different P2P nodes that are downloading and/or sharing these chunks. For decentralization purposes, by design, P2P clients requesting a file will request different chunks from different nodes. This essentially shows up as a series of network flows initiated by the requesting node connecting to a bunch of random target nodes. If the P2P system is designed to be evasive to firewalls and traffic shapers, the only parameter that will be common to these flows will be the source IP address. The destination IPs, destination ports, source ports, and so on will all appear to be random. Also, the flow content itself will be obfuscated or encrypted.

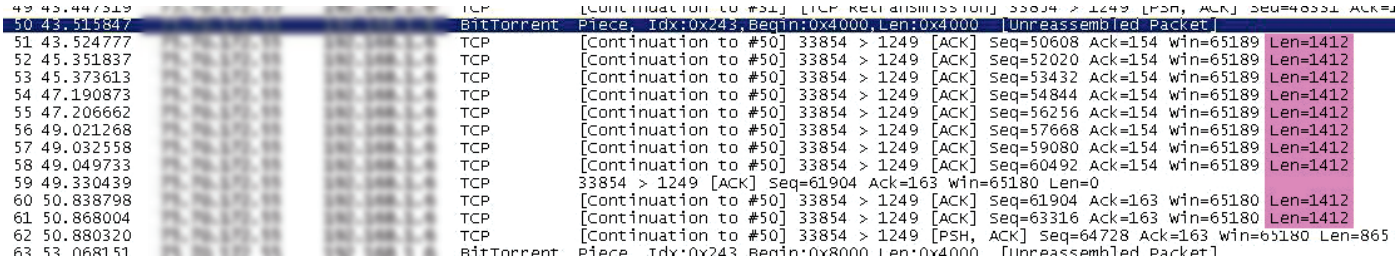

Figure 2: An Ethereal network trace of a P2P (BitTorrent) file-download session

Because of the obfuscated nature of the flows, Phases I and II of the algorithm catch them easily. Phase III does the following time-based correlation between these "component" detections:

- **Source-based correlation**—We identify multiple flows that are identified by Phases I and II of the algorithm (unknown, binary, long-lasting, data-flows) originating from the same source IP.

- **Sweep nature/multiple different destinations**—We count multiple flows connecting to the same destination from a given source IP as one flow. This avoids false alarms where a source makes frequent connections to the same target node (for example, HTTP browser sessions, HTTP/FTP site-mirroring, and so on).

- **User-configurable threshold for the sweep**—The network administrator can define a threshold number (N) of such flows that should be generated by a source to multiple different destinations over a given time period (P) for it to qualify for a P2P file-transfer chunk-request scatter. That is, if the scatter crosses (N) in (P) period, that source, with a corresponding degree of certainty, can be tagged as a P2P file-sharing node.

All flows flagged by Phases I and II that originate from a node that is tagged as a Phase III can be considered to be P2P file-transfer flows with an even higher certainty level. These can then be subject to even more aggressive rate-limiting.

Result II

Phase III of the algorithm provides a network administrator with a threshold-configurable way to identify P2P scatter and further drill down on P2P file-sharing sources. It also provides an even more granular rate-limiting of the protocols that fall into the gray area identified by Phases I and II.

We have tested this unique approach towards detecting and traffic-shaping evasive P2P protocols on large, real-world networks including an enterprise and an open university network environment. We made the following notable observations:

- **Correctness of implementation**—First, we handpicked several network flows that appeared to be unidentifiable bulk-data transfers based on quick manual inspection. We replayed all of these flows with the algorithm and found that it caught 100 percent of these flows. This essentially validates the correctness of the algorithm implementation—it detects all flows that match the definition of our "evasive, long-lasting bulk-data transfer."

All Alerts=All Alerts/Source IP=169.229.119.162

Total Rows # 24
Alert Snapshot at : 2007-05-15 16:06:14.000 PDT

Time	Attack	Source IP	Src...	Destination IP	De...	Do...	Se...	Int...	Applic...
2007-05-14 23:47:15.000 ...	P2P: Unknown Long Binary Response Data-Str...		7229		4166	My Co...	jerry40...	1A	----
2007-05-14 22:59:01.000 ...	P2P: Unknown Long Binary Response Data-Str...		7229		4166	My Co...	jerry40...	1A	----
2007-05-14 23:33:57.000 ...	P2P: Unknown Long Binary Response Data-Str...		7229		4166	My Co...	jerry40...	1A	----
2007-05-14 23:49:03.000 ...	P2P: Unknown Long Binary Response Data-Str...		----		----	My Co...	jerry40...	1A	----
2007-05-14 23:46:33.000 ...	P2P: Unknown Long Binary Response Data-Str...		7229		4166	My Co...	jerry40...	1A	----
2007-05-14 22:59:01.000 ...	P2P: Unknown Long Binary Response Data-Str...		7229		4166	My Co...	jerry40...	1A	----
2007-05-15 07:46:53.000 ...	P2P: Unknown Long Binary Response Data-Str...		3984		80	My Co...	jerry40...	1A	pktsearch
2007-05-15 16:06:14.000 ...	SSL: Bad State Transition		3074		3074	My Co...	jerry40...	1A	http
2007-05-14 23:04:57.000 ...	SSL: Bad State Transition		7229		4166	My Co...	jerry40...	1A	http
2007-05-15 01:03:59.000 ...	P2P: Unknown Long Binary Response Data-Str...		----		----	My Co...	jerry40...	1A	pktsearch
2007-05-15 01:01:21.000 ...	P2P: Unknown Long Binary Response Data-Str...		4417		443	My Co...	jerry40...	1A	pktsearch
2007-05-15 00:59:21.000 ...	P2P: Unknown Long Binary Response Data-Str...		4412		443	My Co...	jerry40...	1A	pktsearch
2007-05-14 22:59:38.000 ...	P2P: Unknown Long Binary Response Data-Str...		----		----	My Co...	jerry40...	1A	----
2007-05-14 22:57:38.000 ...	P2P: Unknown Long Binary Response Data-Str...		----		----	My Co...	jerry40...	1A	----
2007-05-14 22:55:38.000 ...	P2P: Unknown Long Binary Response Data-Str...		----		----	My Co...	jerry40...	1A	----
2007-05-14 22:53:30.000 ...	P2P: Unknown Long Binary Response Data-Str...		4191		443	My Co...	jerry40...	1A	pktsearch
2007-05-14 22:47:32.000 ...	P2P: Unknown Long Binary Response Data-Str...		4167		443	My Co...	jerry40...	1A	pktsearch
2007-05-15 01:21:11.000 ...	P2P: Unknown Long Binary Response Data-Str...		7229		50604	My Co...	jerry40...	1A	----
2007-05-15 10:25:01.000 ...	P2P: Unknown Long Binary Response Data-Str...		54267		443	My Co...	jerry40...	1A	pktsearch
2007-05-15 10:24:13.000 ...	SSL: Packet With No Connection		54259		443	My Co...	jerry40...	1A	ssl
2007-05-14 23:59:03.000 ...	P2P: P2P Distributed File Download Traffic Det...		7229	----	4166	My Co...	jerry40...	1A	----
2007-05-14 23:49:06.000 ...	P2P: P2P Distributed File Download Traffic Det...		7229	----	4166	My Co...	jerry40...	1A	----
2007-05-14 23:12:14.000 ...	P2P: P2P Distributed File Download Traffic Det...		7229	----	4166	My Co...	jerry40...	1A	----
2007-05-14 23:00:58.000 ...	P2P: P2P Distributed File Download Traffic Det...		7229	----	4166	My Co...	jerry40...	1A	----

Figure 3: Network Security Platform's Alert Viewer showing a source detected as a P2P (Joost TV) client

- **No false alarms**—We let the algorithm run live on the university environment for a day, and zoned in on the source hosts caught by the Phase III (distributed P2P evasive bulk-data transfer sweep). Based on detailed forensic analyses, we confirmed that all of these source hosts were involved in P2P activity using protocols we currently did not have coverage for (100 percent true-positives or zero percent false-positives). For example, the screenshot in Figure 3 shows a P2P Joost-TV session detected on the university network.

- The rate of graylisted/unknown flow detection was of the order of thousands per hour when averaged over a week.

- **Significant evasive bulk-data activity**—Of these gray flows, 40 to 50 percent were identified as "evasive bulk-data" based on Phases I and II of the algorithm. In other words, every hour, on that specific network, up to approximately 500 bulk-data flows were previously completely invisible to us and were getting away unmanaged.

- **Whitelisting/blacklisting possibility**—Interestingly, about 30 percent of the gray flows were identified as non-evasive but bulk-data flows. This meant they had only printable characters at all the locations in the flow we inspected. If these printable bytes remained static across flows, network administrators could write user-defined signatures to blacklist or whitelist them, to further improve the traffic-shaping effectiveness.

- **High detection rate**—Finally, we picked a few evasive P2P protocols like BitTorrent. We intentionally removed coverage for these from our blacklists to see how we caught them generically. Using Phases I and II, we caught about 84 to 92 percent of these long-lasting bulk-data flows without a single signature to identify them. Since the P2P protocols we chose (BitTorrent, eMule/eDonkey, etc.) can be considered good representative samples of the P2P traffic out there,[3] we just showed that this graylisting algorithm caught 84 to 92 percent of P2P bulk-data transfer flows that were previously completely invisible to the device.

Looking at the evasiveness of the P2P protocols today, we are convinced that a graylisting approach combined with network behavioral analysis is going to be the only way to detect and manage P2P traffic moving forward. McAfee Network Security Platform is the first product that has adopted this patent-pending approach, and we are already seeing some great results from it. As illustrated above, this approach allowed us to provide different traffic-shaping treatments to the web traffic, identified P2P or other undesirable traffic, and the unknown bulk-data traffic on our network—definitely the kind of traffic management control and visibility that has been missing all this time!

References

1 *http://www.ipoque.com/media/news/ipoques_2007_p2p_survey_to_be_presented_at_technology_reviews_emerging_technologies_conference_at_mit.html*

2 *https://knowledge.mcafee.com/docs/Product-Documentation/INTRSNSR/4.1/Network Security Platform_4.1.3.7-4.1.1.75_Release_Notes.pdf* (Section 7, "Supported Protocols")

3 *http://www.ipoque.com/userfiles/file/Internet_study_2007_abstract_en.pdf* ("Most Popular P2P Protocols")

McAfee, Inc.
3965 Freedom Circle
Santa Clara, CA 95054
888.847.8766
www.mcafee.com