root9B

SHELLTEA + POSLURP MALWARE

YARA RULES

ROOT9B
JUNE 2017

MEMORY-RESIDENT POINT-OF-SALE MALWARE

Retail Point-of-Sale (PoS) systems remain a top target for the financially-motivated hacker. Theft of payment card data in large volume exists not only as its own segment within financial crime, but also serves to facilitate other even more harmful motives of today's criminal elements. To the businesses targeted by cyber criminals, the negative effects are far reaching with impact on brand reputation, consumer and investor confidence, and business growth strategies. With such a lucrative target as payment card data, adversary groups continue to adapt Tactics, Techniques, and Procedures (TTPs) in response to defenders' change in security practices. One effective attacker TTP is to use so-called "fileless," or memory-resident malware, to carry out attacks against retailer PoS systems.

root9B discovered an advanced, targeted PoS intrusion focused on harvesting payment card information for exfiltration. The adversary's campaign has active and operational Command and Control (C2) servers. root9B's analysis determined that the adversary is using advanced memory-resident techniques to maintain persistence and avoid detection. The malware likely required a significant amount of time and knowledge to create. We typically see techniques at this level by well-resourced, well-funded, motivated adversaries.

This ongoing campaign has targeted numerous organizations and their PoS systems. root9B uncovered the TTPs utilized and describes them in a detailed analysis below. At a high-level, the adversary's methodology consists of the following steps:

- **Step 1:** Reconnaissance and targeting of a corporate individual with a spearphishing email attack employing an ActiveMIME document with an MS Office-enabled macro.

- **Step 2:** Email recipient opens the ActiveMIME document attachment and clicks to enable content, executing a PowerShell command initiating a surreptitious shellcode download.

- **Step 3:** A shellcode blob encapsulating a Dynamic Link Library (DLL) malware is dropped in the system registry and loaded into memory, conducting basic enumeration and sandbox detection on the target. This malware appears to be an updated version of "PowerSniff."

- **Step 4:** The malware continues reconnaissance of the target environment and contacts one of its five C2 domains with the results. If the environment meets the conditions the attacker is looking for, the attacker sends additional instructions.

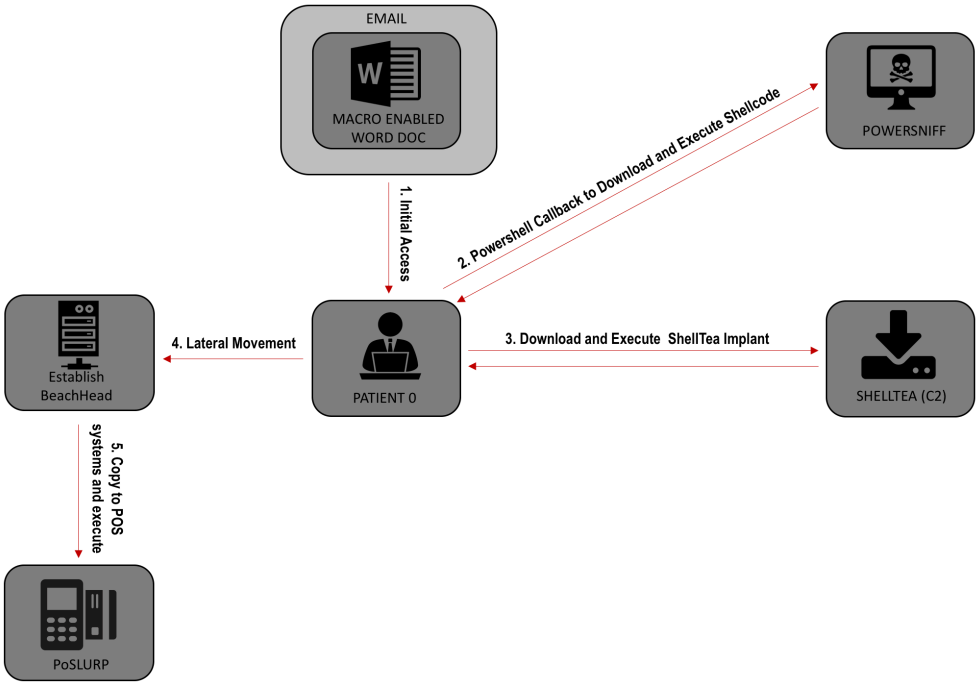
- **Step 5:** The attackers install a second fileless implant in another registry shellcode blob. This implant, which we have named ShellTea, has not been previously observed or reported. We have identified six hardcoded C2 domains utilized by this implant.

- **Step 6:** The attacker explores the network using compromised privileged credentials and establishes persistent staging servers for deploying malware and collecting data from PoS endpoints. Several staging servers are utilized by the attackers to spread the workload and provide redundancy to thwart defensive measures.

- **Step 7:** An advanced PoS RAM scraping malware, we have named PoSlurp, is deployed to the PoS endpoints. PoSlurp directly injects memory-resident code into a privileged user mode process. This capability has not been previously reported. The attacker can specify which PoS processes should be monitored for payment card transactions.

root9B has been able to deconstruct the four major components of the adversary's activities. Provided here is a detailed analysis of the initial access method, command-and-control methods, and the new ShellTea implant and PoSlurp POS RAM Scraper.

ATTACK OVERVIEW



MITIGATION RECOMMENDATIONS

- Apply macro restrictions in your environment to prevent users from inadvertently running malicious Office macros to help address this common initial access vector. For details, see: https://blogs.technet.microsoft.com/mmpc/2016/03/22/new-feature-in-office-2016-can-block-macros-and-help-prevent-infection/.

- Limit the exposure of privileged administrator credentials by following best practices such as the PAW model, audit your credential risks, and require multifactor authentication for privileged users. For details, see: https://docs.microsoft.com/en-us/windows-server/identity/securing-privileged-access/privileged-access-workstations.

- Implement application whitelisting on PoS systems utilizing Microsoft's built-in AppLocker or one of many commercial solutions. Much like the previous mitigation recommendation, PoS systems should be highly standardized and have no software running or installed that isn't part of required functionality. Audit application execution in a development environment to build an effective yet minimal whitelist.

- Develop and maintain a robust security monitoring program or contract an experienced security company. Tune your environment to collect relevant network and endpoint-based artifacts that allow you to detect adversary actions. Focus your analysis on critical network segments and employ active defense methodologies (HUNT) to proactively identify persistent threats.

- Create a whitelist or greylist of domains and IP addresses that your organization is allowed to reach via the network.

- Implement effective network segmentation controls. Prohibiting communication between distinct segments such as PoS and Store networks, except for required ports and protocols, and using different credentials in each network will greatly delay if not eliminate the attacker's ability to traverse the networks. Communication between systems in these critical networks should be far more predictable than the corporate network, enabling a security monitoring program to more easily identify abnormal activity.

INDICATORS OF COMPROMISE (IOCS)

POWERSNIFF C2 DOMAINS

vseflijkoindex.net

vortexclothings.biz

unkerdubsonics.org

popskentown.com

SHELLTEA C2 DOMAINS

neofilgestunin.org

verfgainling.net

straubeoldscles.org

olohvikoend.org

menoograskilllev.net

asojinoviesder.org

TOOLS

Yara Rules, Function Hash Resolution Tool, IDA Script, and Process Name CRC32 Code: https://gist.github.com/root9b/24b9b25f3b0b06a6939881e68d0bd2d0

YARA RULES

```
/*

  root9B Yara Rules for SHELLTEA + POSLURP MALWARE blog entry
  https://www.root9b.com/newsroom/shelltea-poslurp-malware

*/

rule PoSlurpFile : PoSlurp
{
    meta:
        copyright = "root9b, LLC"
        authors = "Matt Weeks, Dax Morrow"
        description = "ShellTea + PoSlurp PoS Malware on Disk PoSlurp executable"
        reference = "https://www.root9b.com/newsroom/shelltea-poslurp-malware"
        version = "1.0"
        last_modified = "2017-06-27"

  strings:
        $hex1 = { 81 C2 FF 5C F3 22 52 56 E8 } // outer layer custom function resolver

 condition:
        uint16(0) == 0x5A4D and uint32(uint32(0x3C)) == 0x00004550 and $hex1
}

rule inRegPowerSniff : PowerSniff
{
```

```
meta:
        copyright = "root9b, LLC"
        authors = "Matt Weeks, Dax Morrow"
        description = "ShellTea + PoSlurp PoS Malware in Registry PowerSniff"
reference = "https://www.root9b.com/newsroom/shelltea-poslurp-malware"
version = "1.0"
        last_modified = "2017-06-27"

strings:
        $hex1 = { 41 2B CF 81 38 BE BA AD AB 48 8B D0 75 09 81 78 04 0D F0 AD 8B } //
shellcode blob in registry

condition:
$hex1
}

rule inRegShellTea : ShellTea {

meta:
        copyright = "root9b, LLC"
  authors = "Matt Weeks, Dax Morrow"
        description = "ShellTea + PoSlurp PoS Malware in Registry ShellTea"
reference = "https://www.root9b.com/newsroom/shelltea-poslurp-malware"
version = "1.0"
        last_modified = "2017-06-27"


strings:
        $hex1 = { 48 83 EC 28 E8 F7 03 00 00 [1015] 48 89 5C 24 18 48 89 4C 24 08 55 56 57 41
54 41 } // Binary registry value with variable content for ShellTea config

condition:
$hex1
}

rule inMemPowerSniff : PowerSniff {

meta:
        copyright = "root9b, LLC"
        authors = "Matt Weeks, Dax Morrow"
        description = "ShellTea + PoSlurp in Memory PowerSniff"
        reference = "https://www.root9b.com/newsroom/shelltea-poslurp-malware"
version = "1.0"
        last_modified = "2017-06-27"


strings:
        $wide_string = "/%s?
user=%08x%08x%08x%08x&id=%u&ver=%u&os=%lu&os2=%lu&host=%u&k=%lu&type=%u" wide //
PowerSniff URL Pattern
        $wide_string2 = "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT %u.%u%s)" wide // PowerSniff
URL Pattern

condition:
all of them
}

rule inMemShellTea : ShellTea {

meta:
        copyright = "root9b, LLC"
        authors = "Matt Weeks, Dax Morrow"
        description = "ShellTea + PoSlurp PoS Malware in Memory ShellTea"
reference = "https://www.root9b.com/newsroom/shelltea-poslurp-malware"
version = "1.0"
        last_modified = "2017-06-27"


strings:
          $hex1 = { B9 1D C7 12 45 E8 } // opcodes for function hash
$hex2 = { B9 52 7E 10 E1 E8 } // opcodes for function hash
$hex3 = { B9 CC 11 67 D6 E8 } // opcodes for function hash

condition:
all of them
}

rule inMemPoSlurp : PoSlurp {

meta:
        copyright = "root9b, LLC"
        authors = "Matt Weeks, Dax Morrow"
        description = "ShellTea + PoSlurp PoS Malware in Memory PoSlurp"
reference = "https://www.root9b.com/newsroom/shelltea-poslurp-malware"
version = "1.0"
        last_modified = "2017-06-27"

        strings:
         $hex1 = { C6 45 ED 65 C6 45 EE 72 C6 45 EF 6E C6 45 F0 65 }  // Kernel32
obfuscated
         $hex2 = { E8 EE FD FF FF 68 88 13 00 00 FF D6 8D 44 24 18 50 FF D7 8D 44 24 10 50 8D 44 24 1C 50 FF D3 8B
44 24 10 2B 05 80 50 40 00 8B 4C 24 14 1B 0D 84 50 40 00 6A 00 68 80 96 98 00 51 50 E8 B7 05 00 00 6A 3C 33
D2 59 F7 F1 3B 05 2C 40 40 00 72 B0 } // opcodes f rom top-level scan memory basic block

        condition:
        all of them
}
```