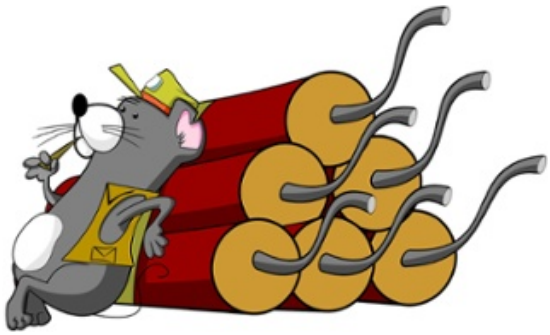
ENSILO
SILO YOUR DATA FROM THREAT ACTORS

MENU

OCTOBER 27, 2016 BY TAL LIBERMAN

AtomBombing: A Code Injection that Bypasses Current Security Solutions



Our research team has uncovered new way to leverage mechanisms of the underlying Windows operating system in order to inject malicious code. Threat actors can use this technique, which exists by design of the operating system, to bypass current security solutions that attempt to prevent infection. We named this technique AtomBombing based on the name of the underlying mechanism that this technique exploits.

AtomBombing affects all Windows version. In particular, we tested this against Windows 10.

Unfortunately, this issue cannot be patched since it doesn't rely on broken or flawed code – rather on how these operating system mechanisms are designed.

Code Injection 101

The issue we revealed presents a way for threat actors to inject code. Attackers use code injection to add malicious code into legitimate processes, making it easier to bypass security products, hide from the user, and extract sensitive information that would otherwise be unattainable.

For example, let's say an attacker was able to persuade a user to run a malicious executable, *evil.exe*. Any kind of decent application level firewall installed on the computer would block that executable's communication. To overcome this issue, *evil.exe* would have to find a way to manipulate a legitimate program, such as a web browser, so that the legitimate program would carry out communication on behalf of *evil.exe*.

This manipulation technique is known as code injection.

Code Injection: An Important Tool in the Attacker's Toolbox

There are quite a few reasons why code injection is useful. An attacker may use code injection, for example, to:

1. ***Bypass process level restrictions:*** Many security products employ a white list of trusted processes. If the attacker is able to inject malicious code into one of those trusted processes, the security product can

easily be bypassed.

2. ***Access to context-specific data.*** Some data is only accessible to certain processes, while inaccessible to others. For example:
   a. ***Taking screenshots.*** A process that takes a screenshot of the user's screen, must run within the context of the user's desktop. However, more often than not malware will be loaded into the services desktop, not the user's, preventing the malware from taking a screenshot of the user's desktop. Using code injection, a malware can inject code into a process that's already running in the user's desktop, take a picture and send it back to the malware in the services desktop.
   b. ***Performing Man in the Browser (MitB) attacks.*** By injecting code into a web browser an attacker can modify the content shown to the user. For example, in a banking transaction process, the customer will always be shown the exact payment information as the customer intended via confirmation screens. However, the attacker modifies the data so that the bank receives false transaction information in favor of the attacker, i.e. a different destination account number and possibly amount. In a MitB attack, the customers are unaware of the money being funneled out of their account until it's too late.
   c. ***Accessing encrypted passwords.*** Google Chrome encrypts the user's stored passwords by using Windows Data Protection API (DPAPI). This API uses data derived from the current user to encrypt/decrypt the data and access the passwords. In this scenario, a malware that is not running in the context of the user will not be able to access the passwords. However, if the malware injects code into a process that's already running in the context of the current user, the plain-text passwords can be easily accessed.

Behind the Scenes of AtomBombing

The underlying Windows mechanism which AtomBombing exploits is called atom tables. These tables are provided by the operating system to allow applications to store and access data. These atom tables can also be used to share data between applications.

What we found is that a threat actor can write malicious code into an atom table and force a legitimate program to retrieve the malicious code from the table. We also found that the legitimate program, now containing the malicious code, can be manipulated to execute that code.

For the technology deep dive, please the researcher's post here: https://breakingmalware.com/injection-techniques/atombombing-brand-new-code-injection-for-windows/

Code Injections in the Past

Currently there are just a handful of known code injection techniques. A list of several of these can be found here: http://resources.infosecinstitute.com/code-injection-techniques/

Additionally, last summer our research team found a new code injection technique called PowerLoaderEx. PowerLoaderEx enables an attacker to inject code without needing to actually write code or data to the injected process.

Once a code injection technique is well-known, security products focused on preventing attackers from compromising the endpoints (such as anti-virus and host intrusion prevention systems), typically update their signatures accordingly. So once the injection is known, it can be detected and mitigated by the security products.

Being a new code injection technique, AtomBombing bypasses AV, NGAV and other endpoint infiltration prevention solutions.

Mitigation

AtomBombing is performed just by using the underlying Windows mechanisms. There is no need to exploit operating system bugs or vulnerabilities.

Since the issue cannot be fixed, there is no notion of a patch for this. Thus, the direct mitigation answer would be to tech-dive into the API calls and monitor those for malicious activity.

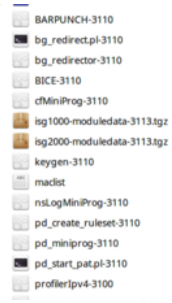
It's important though at this point to take a step back. AtomBombing is one more technique in the attacker's toolbox. Threat actors will continuously take out a tool – used or new - to ensure that they bypass anti-infiltration technologies (such as AV, NGAV, HIPS, etc).

Obviously we need to find a different way to deal with threat actors. Under the assumption that threat actors will always exploit known and unknown techniques, we need to build our defenses in a way that prevents the consequences of the attack once the threat actor has already compromised the environment.

READ NEXT

HEALTHCARE - A CYBERATTACK TARGET THAT'S ONLY GETTING BIGGER

THE ZERO-DOWNTIME ORGANIZATION



Cyber-Security in 120 S
Brokers Free-for-All Lo

POST TAGS RESEARCH

Comments for this thread are now closed.

18 Comments **Ensilo Blog** **Login**

Recommend 4 **Share** **Sort by Best**

MonkeyBoy • 4 months ago

Got here from: http://thehackernews.com/2016/...

A lot of hype from that site.
#1 you say "For example, let's say an attacker was able to persuade a user to run a malicious executable, evil.exe"

Yea! Hello? If you get any "evil.exe" run on your system you're in trouble.
Not by this technique alone, as the executable can install a number of components (rootkits) in userland and/or kernel space.

That page goes on to hype it up more talking about "encryption" key hijacking, etc., which is not particular to this technique..

2 ^ | v • Share ›

Tal_Liberman ➚ MonkeyBoy • 4 months ago

This technique can be used post infection to inject code from one process to another. For example, on a computer protected by modern security software a malicious executable such as evil.exe will not be able to communicate because it would be blocked by a firewall. It would need to inject code into a browser in order to communicate which will also be blocked by any standard AV. By using AtomBombing evil.exe can inject code into a web browser undetected by said AV and then have the web browser communicate on its behalf undetected by said firewall.

1 ^ | v • Share ›

Sebastian Cato ➚ Tal_Liberman • 4 months ago

Why not just use CreateRemoteThread or NtQueueApcThread calling something like LoadLibrary? Using atom tables just sounds like it would achieve the same goal with added hassle, and you wouldn't gain anything, since it still requires NtQueueApcThread, which AFAIK most AV hooks at runtime. It's been used for a while now, see e.g., [1]

[1]: http://www.codeproject.com/Art...

^ | v • Share ›

Tal_Liberman ➚ Sebastian Cato • 4 months ago

Use CreateRemoteThread or NtQueueApcThread to call LoadLibrary and any AV will catch you. Use AtomBombing and you'll bypass most AV products.

Some AVs do hook NtQueueApcThread but they'll only flag your malicious behavior if you use NtQueueApcThread to call LoadLibrary or some code that was allocated dynamically with VirtualAllocEx. Also if you use NtQueueApcThread to call LoadLibrary you must place a DLL file on the disk, which will be picked up by the AV and sent to its cloud service for reputation analysis. AtomBombing allows you to inject shellcode without having to touch the disk.

2 ^ | v • Share ›

Hacked_OFF • 4 months ago

This explains how we have been able to be hacked for the last 18 months no matter what we do in terms of security features in Windows 10. This is actually a design feature inserted by Microsoft as a back door for the security services, as we have discovered. We strongly suspected this and now it have been proven. Bloody Hell !!!!, Microsoft software update policy is also designed to allow the security services to compromise updates... so watch out !!!

1 ^ | v • Share ›

level1 ➚ Hacked_OFF • 4 months ago

Get the tin hats out...

^ | v • Share ›

ExoticWaves • 4 months ago

Is Mac OSX susceptible to this attack vector, "Atom bombing"???

^ | v • Share ›

Tal_Liberman ➚ ExoticWaves • 4 months ago

No, AtomBombing only affects Windows.

^ | v • Share ›

أبو نزار • 4 months ago

Is this gap infect Linux Operating Systems !!!

Share ›

Dylan Taylor ➚ أبو نزار • 4 months ago

Linux is not affected by this, no.

Share ›

Cees Timmerman • 4 months ago

Threat signatures are always outdated. Use signed hashes to ensure the integrity of your system's authorized software.

Share ›

Robert • 4 months ago

Have you spoken to anyone at Microsoft about this (i.e. MSRC)?

Share ›

Tal_Liberman ➚ Robert • 4 months ago

This code injection technique is not dependent on a vulnerability in Microsoft. Rather it leverages legitimate building blocks of Windows.

1 Share ›

bro • 4 months ago

So, how to apply this? AtomTable allows only Strings and Integers (based on MS documentation). So, only affected applications are apps that uses GlobalAtomTable in such a way that allows bypass some security principles (like storing code in form of string, or path to dll in form of string etc.). Still it sounds like a problem with applications not with Windows API. Apps should consider data coming from GlobalAtomTable as untrusted source. Can you name application that is vulnerable to this flow?

Share ›

Tal_Liberman ➚ bro • 4 months ago

I suggest you read the detailed write-up on our technical blog:
http://breakingmalware.com/inj...

Share ›

Mitja Kolsek • 4 months ago

Sounds like a very Interesting research, thank you for sharing it.

I was hoping for more technical details but I guess you have reasons not to reveal them at this point. Nevertheless, I'm puzzled by your claim that "... the legitimate program, now containing the malicious code, can be manipulated to execute that code" in conjunction with "the issue cannot be fixed". Does this mean that telling an application to accept code from an atom table and execute it is a by-design documented feature? If so, it sounds like a gaping hole that Microsoft will have to close, but if not, then it sounds like you have found a way to trick it to do so (which also implies a need for a patch). I'll appreciate more information - thanks!

Share ›

Tal_Liberman ➚ Mitja Kolsek • 4 months ago

Hi Mitja,
I believe the detailed write-up on our technical blog should answer all of your questions:
http://breakingmalware.com/inj...

Subscribe to enSilo's Blog and Stay on Top of the Latest Security Research and Industry News

Email*

SUBSCRIBE

Recent Posts

- RSA in 120 Secs: In Case You Missed Us at the Show...
- Cyber-Security in 120 Secs: UPDATE - Cost of a Data Breach
- Customer Advisory Warning: The Comeback of the Hancitor Campaign
- Cyber-Security in 120 Secs: The Data Breach Effect and The Cost
- Cyber-Security in 120 Secs: Cyber-Attacks on Banks
- Cyber-Security in 120 Secs: Drafting a Cyber Security Executive Order
- Cyber-Security in 120 Secs: Compromise is Inevitable
- Cyber-Security in 120 Secs: The Shadow Brokers Free-for-All Loot
- WhatsApp With That: One Says Backdoor, the Other Says Feature
- Cyber-Security in 120 Secs: 'Mungous Ransomware on MongoDB

Posts by Topic

- Weekly Security News (68)
- Industry (19)
- Research (16)
- Business (9)

Archive by Month

- May 2016 (9)
- December 2016 (9)
- October 2015 (7)
- February 2016 (7)
- July 2016 (7)
- November 2016 (7)
- December 2015 (6)
- January 2016 (6)
- September 2016 (6)
- February 2017 (6)
- November 2015 (5)
- August 2016 (5)

see all