From Technofeudalism to Technosocialism: FORTH unexpected roles in Decentralised Systems

Part A: Introduction

We began this presentation with a brief description of a FORTH like engine in Bitcoin (Part B) and how the invisible internet project (Part B), being a crucial alternative to the domain name system, was used as the backbone of Bitcoin but remains relatively obscure.

We then extended the concept of Bitcoin address, as the hash of public key, to Omnihash user identifier and Omnihash JSON, being a generalised representation of digital assets with Decentralised ownerships. (Part C)

We further propose to employ a FORTH like engine in shell libraries for JavaScript and Java applications (Part D: Trojan Horse Super-Shell), in order to consolidate fragmented user base of open source projects and small commercial social media applications to compete with and create one to one replacements for mainstream dominant social media platforms (Goal of Decentralised Systems).

From Technofeudalism to Technosocialism: FORTH unexpected roles in Decentralised Systems

Technofeudalism is a term coined by former Greek finance minister Yanis Varoufakis as the title of a book which compares top American technology companies to feudal lords, who literally enslaved individual users and free software programmers, exploiting their data and source code without fairly rewarding them.

Technosocialism has been used by many other authors but I use it to describe the very real but underreported progress made by free software programmers and free software, despite the lack of fair mechanisms for rewards so far. Free software programmers or technosocialists have made technofeudalism possible as a matter of fact, but they do not get to enjoy the rewards.

Bitcoin and related projects were supposed to address this problem, and are successful in their own way, so we are now half way towards technosocialism.

Part B:
FORTH-like Engine in Bitcoin
+ Invisible Internet Project

```
406 bool EvalScript(std::vector<std::vector<unsigned char> >& stack, const CScript& script, unsigned int fla
407 {
408     static const CScriptNum bnZero(0);
409     static const CScriptNum bnOne(1);
410     // static const CScriptNum bnFalse(0);
411     // static const CScriptNum bnTrue(1);
412     static const valtype vchFalse(0);
413     // static const valtype vchZero(0);
414     static const valtype vchTrue(1, 1);
415
416     // sigversion cannot be TAPROOT here, as it admits no script execution.
417     assert(sigversion == SigVersion::BASE || sigversion == SigVersion::WITNESS_V0 || sigversion == SigV
418
419     CScript::const_iterator pc = script.begin();
420     CScript::const_iterator pend = script.end();
421     CScript::const_iterator pbegincodehash = script.begin();
422     opcodetype opcode;
423     valtype vchPushValue;
424     ConditionStack vfExec;
```

```
406  bool EvalScript(std::vector<std::vector<unsigned char> >& stack, const CScript& script, unsigned int fl
435
436      try
437      {
438          for (; pc < pend; ++opcode_pos) {
439              bool fExec = vfExec.all_true();
440
441              //
442              // Read instruction
443              //
444              if (!script.GetOp(pc, opcode, vchPushValue))
445                  return set_error(serror, SCRIPT_ERR_BAD_OPCODE);
446              if (vchPushValue.size() > MAX_SCRIPT_ELEMENT_SIZE)
447                  return set_error(serror, SCRIPT_ERR_PUSH_SIZE);
448
449              if (sigversion == SigVersion::BASE || sigversion == SigVersion::WITNESS_V0) {
450                  // Note how OP_RESERVED does not count towards the opcode limit.
451                  if (opcode > OP_16 && ++nOpCount > MAX_OPS_PER_SCRIPT) {
452                      return set_error(serror, SCRIPT_ERR_OP_COUNT);
453                  }
```

```
406     bool EvalScript(std::vector<std::vector<unsigned char> >& stack, const CScript& script, unsigned int fl

    if (opcode == OP_CAT ||
        opcode == OP_SUBSTR ||
        opcode == OP_LEFT ||
        opcode == OP_RIGHT ||
        opcode == OP_INVERT ||
        opcode == OP_AND ||
        opcode == OP_OR ||
        opcode == OP_XOR ||
        opcode == OP_2MUL ||
        opcode == OP_2DIV ||
        opcode == OP_MUL ||
        opcode == OP_DIV ||
        opcode == OP_MOD ||
        opcode == OP_LSHIFT ||
        opcode == OP_RSHIFT)
        return set_error(serror, SCRIPT_ERR_DISABLED_OPCODE); // Disabled opcodes (CVE-2010-5137).

    // With SCRIPT_VERIFY_CONST_SCRIPTCODE, OP_CODESEPARATOR in non-segwit script is rejected even in an unexecu
```

```
406   bool EvalScript(std::vector<std::vector<unsigned char> >& stack, const CScript& script, unsigned int fl
483               switch (opcode)
484               {
485                   //
486                   // Push value
487                   //
488                   case OP_1NEGATE:
489                   case OP_1:
490                   case OP_2:
491                   case OP_3:
492                   case OP_4:
493                   case OP_5:
494                   case OP_6:
495                   case OP_7:
496                   case OP_8:
497                   case OP_9:
498                   case OP_10:
499                   case OP_11:
500                   case OP_12:
```

```
406  bool EvalScript(std::vector<std::vector<unsigned char> >& stack, const CScript& script, unsigned int fl
515                  //
516                  // Control
517                  //
518                  case OP_NOP:
519                      break;
520
521                  case OP_CHECKLOCKTIMEVERIFY:
522                  {
523                      if (!(flags & SCRIPT_VERIFY_CHECKLOCKTIMEVERIFY)) {
524                          // not enabled; treat as a NOP2
525                          break;
526                      }
527
528                      if (stack.size() < 1)
529                          return set_error(serror, SCRIPT_ERR_INVALID_STACK_OPERATION);
530
531                      // Note that elsewhere numeric opcodes are limited to
532                      // operands in the range -2**31+1 to 2**31-1, however it is
```

```
406  bool EvalScript(std::vector<std::vector<unsigned char> >& stack, const CScript& script, unsigned int fl
601
602                  case OP_IF:
603                  case OP_NOTIF:
604                  {
605                      // <expression> if [statements] [else [statements]] endif
606                      bool fValue = false;
607                      if (fExec)
608                      {
609                          if (stack.size() < 1)
610                              return set_error(serror, SCRIPT_ERR_UNBALANCED_CONDITIONAL);
611                          valtype& vch = stacktop(-1);
612                          // Tapscript requires minimal IF/NOTIF inputs as a consensus rule.
613                          if (sigversion == SigVersion::TAPSCRIPT) {
614                              // The input argument to the OP_IF and OP_NOTIF opcodes must be either
615                              // exactly 0 (the empty vector) or exactly 1 (the one-byte vector with valu
616                              if (vch.size() > 1 || (vch.size() == 1 && vch[0] != 1)) {
617                                  return set_error(serror, SCRIPT_ERR_TAPSCRIPT_MINIMALIF);
618                              }
619                          }
```

```
406 bool EvalScript(std::vector<std::vector<unsigned char> >& stack, const CScript& script, unsigned int fl
635
636                     case OP_ELSE:
637                     {
638                         if (vfExec.empty())
639                             return set_error(serror, SCRIPT_ERR_UNBALANCED_CONDITIONAL);
640                         vfExec.toggle_top();
641                     }
642                     break;
643
644                     case OP_ENDIF:
645                     {
646                         if (vfExec.empty())
647                             return set_error(serror, SCRIPT_ERR_UNBALANCED_CONDITIONAL);
648                         vfExec.pop_back();
649                     }
650                     break;
651
652                     case OP_VERIFY:
653                     {
```

```
406     bool EvalScript(std::vector<std::vector<unsigned char> >& stack, const CScript& script, unsigned int fl
673                 //
674                 // Stack ops
675                 //
676                 case OP_TOALTSTACK:
677                 {
678                     if (stack.size() < 1)
679                         return set_error(serror, SCRIPT_ERR_INVALID_STACK_OPERATION);
680                     altstack.push_back(stacktop(-1));
681                     popstack(stack);
682                 }
683                 break;
684
685                 case OP_FROMALTSTACK:
686                 {
687                     if (altstack.size() < 1)
688                         return set_error(serror, SCRIPT_ERR_INVALID_ALTSTACK_OPERATION);
689                     stack.push_back(altstacktop(-1));
690                     popstack(altstack);
```

```
406  bool EvalScript(std::vector<std::vector<unsigned char> >& stack, const CScript& script, unsigned int fl
693
694                  case OP_2DROP:
695                  {
696                      // (x1 x2 -- )
697                      if (stack.size() < 2)
698                          return set_error(serror, SCRIPT_ERR_INVALID_STACK_OPERATION);
699                      popstack(stack);
700                      popstack(stack);
701                  }
702                  break;
703
704                  case OP_2DUP:
705                  {
706                      // (x1 x2 -- x1 x2 x1 x2)
707                      if (stack.size() < 2)
708                          return set_error(serror, SCRIPT_ERR_INVALID_STACK_OPERATION);
709                      valtype vch1 = stacktop(-2);
710                      valtype vch2 = stacktop(-1);
711                      stack.push_back(vch1);
```

```
406  bool EvalScript(std::vector<std::vector<unsigned char> >& stack, const CScript& script, unsigned int fl
703
704                  case OP_2DUP:
705                  {
706                      // (x1 x2 -- x1 x2 x1 x2)
707                      if (stack.size() < 2)
708                          return set_error(serror, SCRIPT_ERR_INVALID_STACK_OPERATION);
709                      valtype vch1 = stacktop(-2);
710                      valtype vch2 = stacktop(-1);
711                      stack.push_back(vch1);
712                      stack.push_back(vch2);
713                  }
714                  break;
715
716                  case OP_3DUP:
717                  {
718                      // (x1 x2 x3 -- x1 x2 x3 x1 x2 x3)
719                      if (stack.size() < 3)
720                          return set_error(serror, SCRIPT_ERR_INVALID_STACK_OPERATION);
721                      valtype vch1 = stacktop(-3);
```

```
406     bool EvalScript(std::vector<std::vector<unsigned char> >& stack, const CScript& script, unsigned int fl
730                 case OP_2OVER:
731                 {
732                     // (x1 x2 x3 x4 -- x1 x2 x3 x4 x1 x2)
733                     if (stack.size() < 4)
734                         return set_error(serror, SCRIPT_ERR_INVALID_STACK_OPERATION);
735                     valtype vch1 = stacktop(-4);
736                     valtype vch2 = stacktop(-3);
737                     stack.push_back(vch1);
738                     stack.push_back(vch2);
739                 }
740                 break;
741
742                 case OP_2ROT:
743                 {
744                     // (x1 x2 x3 x4 x5 x6 -- x3 x4 x5 x6 x1 x2)
745                     if (stack.size() < 6)
746                         return set_error(serror, SCRIPT_ERR_INVALID_STACK_OPERATION);
747                     valtype vch1 = stacktop(-6);
```

```
406     bool EvalScript(std::vector<std::vector<unsigned char> >& stack, const CScript& script, unsigned int fl

755                 case OP_2SWAP:
756                 {
757                     // (x1 x2 x3 x4 -- x3 x4 x1 x2)
758                     if (stack.size() < 4)
759                         return set_error(serror, SCRIPT_ERR_INVALID_STACK_OPERATION);
760                     swap(stacktop(-4), stacktop(-2));
761                     swap(stacktop(-3), stacktop(-1));
762                 }
763                 break;
764
765                 case OP_IFDUP:
766                 {
767                     // (x - 0 | x x)
768                     if (stack.size() < 1)
769                         return set_error(serror, SCRIPT_ERR_INVALID_STACK_OPERATION);
770                     valtype vch = stacktop(-1);
771                     if (CastToBool(vch))
772                         stack.push_back(vch);
```

```
406  bool EvalScript(std::vector<std::vector<unsigned char> >& stack, const CScript& script, unsigned int fl
765                  case OP_IFDUP:
766                  {
767                      // (x - 0 | x x)
768                      if (stack.size() < 1)
769                          return set_error(serror, SCRIPT_ERR_INVALID_STACK_OPERATION);
770                      valtype vch = stacktop(-1);
771                      if (CastToBool(vch))
772                          stack.push_back(vch);
773                  }
774                  break;
775
776                  case OP_DEPTH:
777                  {
778                      // -- stacksize
779                      CScriptNum bn(stack.size());
780                      stack.push_back(bn.getvch());
781                  }
782                  break;
```

```
406  bool EvalScript(std::vector<std::vector<unsigned char> >& stack, const CScript& script, unsigned int fl
783
784                  case OP_DROP:
785                  {
786                      // (x -- )
787                      if (stack.size() < 1)
788                          return set_error(serror, SCRIPT_ERR_INVALID_STACK_OPERATION);
789                      popstack(stack);
790                  }
791                  break;
792
793                  case OP_DUP:
794                  {
795                      // (x -- x x)
796                      if (stack.size() < 1)
797                          return set_error(serror, SCRIPT_ERR_INVALID_STACK_OPERATION);
798                      valtype vch = stacktop(-1);
799                      stack.push_back(vch);
800                  }
801                  break;
```

```
406  bool EvalScript(std::vector<std::vector<unsigned char> >& stack, const CScript& script, unsigned int fl
803                  case OP_NIP:
804                  {
805                      // (x1 x2 -- x2)
806                      if (stack.size() < 2)
807                          return set_error(serror, SCRIPT_ERR_INVALID_STACK_OPERATION);
808                      stack.erase(stack.end() - 2);
809                  }
810                  break;
811
812                  case OP_OVER:
813                  {
814                      // (x1 x2 -- x1 x2 x1)
815                      if (stack.size() < 2)
816                          return set_error(serror, SCRIPT_ERR_INVALID_STACK_OPERATION);
817                      valtype vch = stacktop(-2);
818                      stack.push_back(vch);
819                  }
820                  break;
```

```
406  bool EvalScript(std::vector<std::vector<unsigned char> >& stack, const CScript& script, unsigned int fl
822                  case OP_PICK:
823                  case OP_ROLL:
824                  {
825                      // (xn ... x2 x1 x0 n - xn ... x2 x1 x0 xn)
826                      // (xn ... x2 x1 x0 n - ... x2 x1 x0 xn)
827                      if (stack.size() < 2)
828                          return set_error(serror, SCRIPT_ERR_INVALID_STACK_OPERATION);
829                      int n = CScriptNum(stacktop(-1), fRequireMinimal).getint();
830                      popstack(stack);
831                      if (n < 0 || n >= (int)stack.size())
832                          return set_error(serror, SCRIPT_ERR_INVALID_STACK_OPERATION);
833                      valtype vch = stacktop(-n-1);
834                      if (opcode == OP_ROLL)
835                          stack.erase(stack.end()-n-1);
836                      stack.push_back(vch);
837                  }
838                  break;
839
```

```
406  bool EvalScript(std::vector<std::vector<unsigned char> >& stack, const CScript& script, unsigned int fl
840                  case OP_ROT:
841                  {
842                      // (x1 x2 x3 -- x2 x3 x1)
843                      //  x2 x1 x3  after first swap
844                      //  x2 x3 x1  after second swap
845                      if (stack.size() < 3)
846                          return set_error(serror, SCRIPT_ERR_INVALID_STACK_OPERATION);
847                      swap(stacktop(-3), stacktop(-2));
848                      swap(stacktop(-2), stacktop(-1));
849                  }
850                  break;
851
852                  case OP_SWAP:
853                  {
854                      // (x1 x2 -- x2 x1)
855                      if (stack.size() < 2)
856                          return set_error(serror, SCRIPT_ERR_INVALID_STACK_OPERATION);
857                      swap(stacktop(-2), stacktop(-1));
858                  }
```

Part B: Invisible Internet Project

Section page, summary here, highlight words, give addresses

achow101 Merge #29833: i2p: fix and improve logs b27afb7 · 5 months ago

494 lines (414 loc) · 15.4 KB

Code Blame Raw

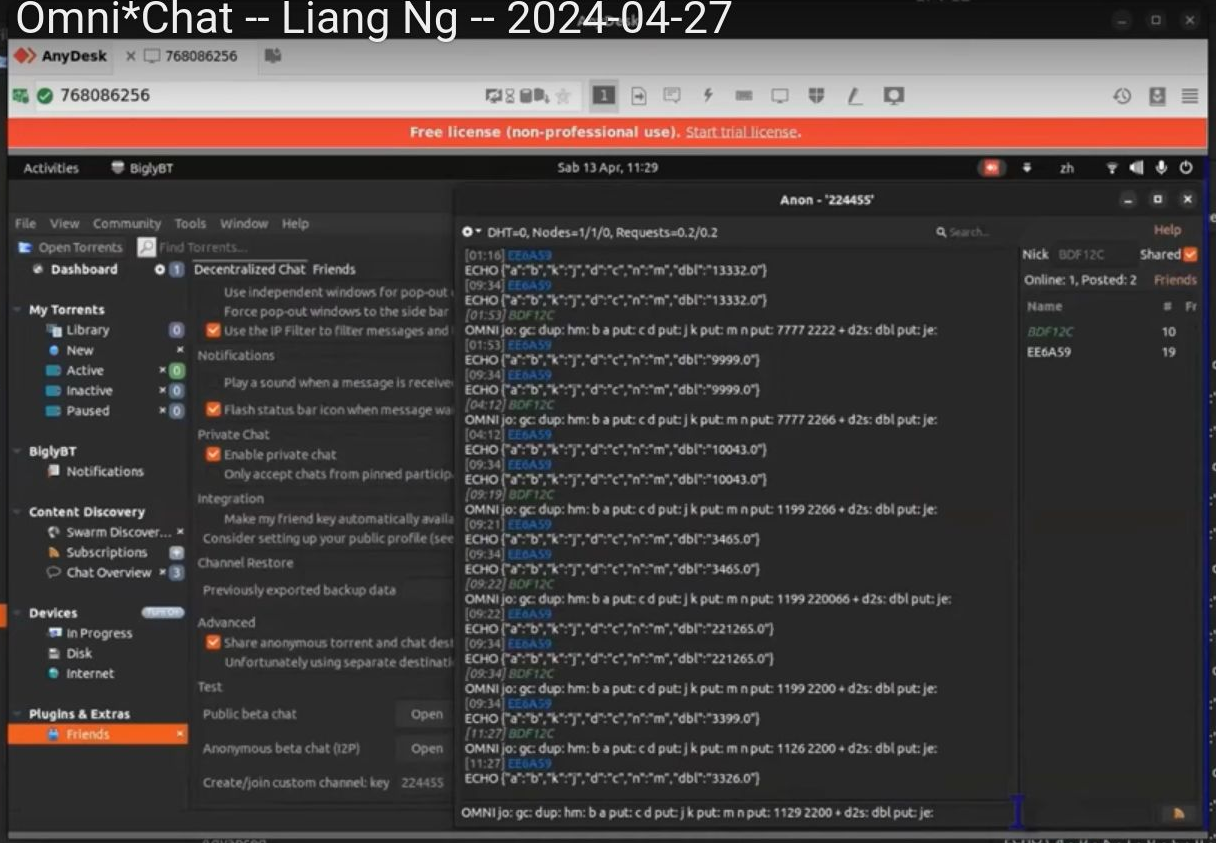
```
1   // Copyright (c) 2020-2022 The Bitcoin Core developers
2   // Distributed under the MIT software license, see the accompanying
3   // file COPYING or http://www.opensource.org/licenses/mit-license.php.
4
5   #include <chainparams.h>
```

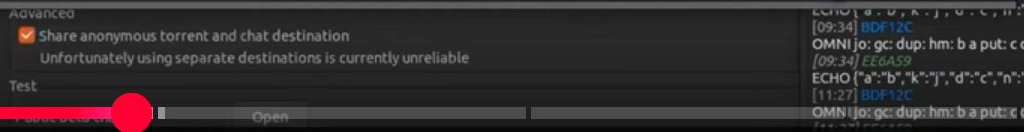
master bitcoin / src / i2p.cpp Top

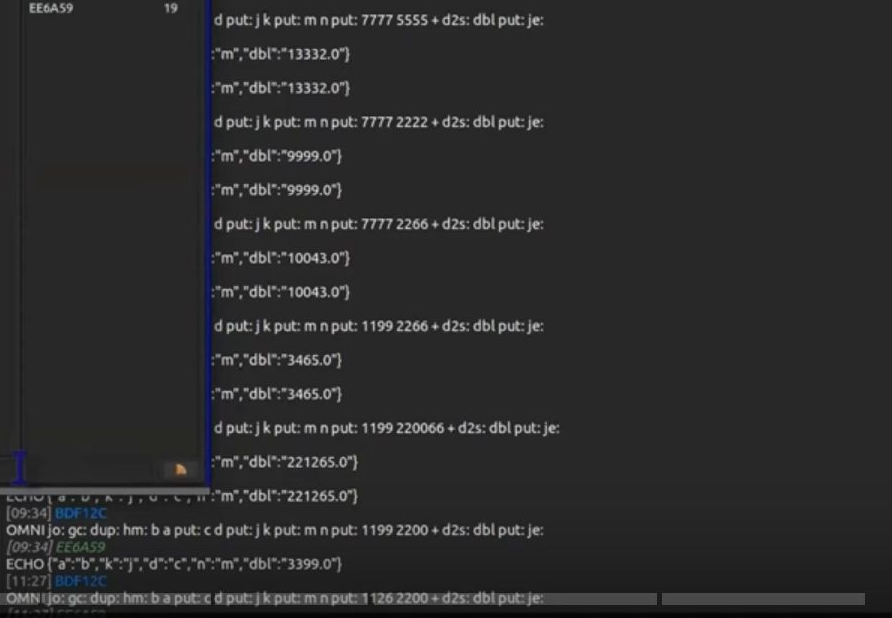
Code Blame Raw

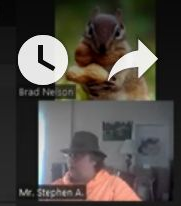
```
87      * @throw std::runtime_error if conversion fails
88      */
89     static CNetAddr DestBinToAddr(const Binary& dest)
90     {
91         CSHA256 hasher;
92         hasher.Write(dest.data(), dest.size());
93         unsigned char hash[CSHA256::OUTPUT_SIZE];
94         hasher.Finalize(hash);
95
96         CNetAddr addr;
97         const std::string addr_str = EncodeBase32(hash, false) + ".b32.i2p";
98         if (!addr.SetSpecial(addr_str)) {
99             throw std::runtime_error(strprintf("Cannot parse I2P address: \"%s\"", addr_str));
100        }
101
102        return addr;
```

Omni*Chat -- Liang Ng -- 2024-04-27









SSR-2024-04-13-11.29.04.mkv

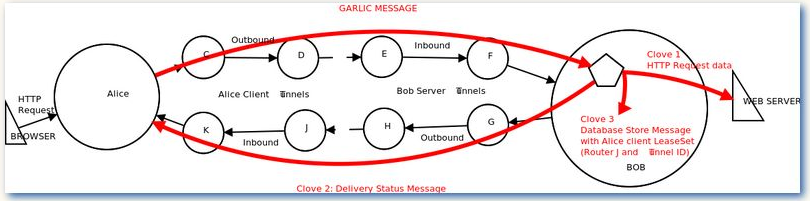
4:03 / 24:32 • Introduction ›

```
21  public class Hash extends SimpleDataStructure {
106     }
107
108     /**
109      *  For convenience.
110      *  @return "{52 chars}.b32.i2p" or null if data not set.
111      *  @since 0.9.25
112      */
113     public String toBase32() {
114         if (_data == null)
115             return null;
116         return Base32.encode(_data) + ".b32.i2p";
117     }
118
119     /**
120      *  @since 0.9.17
```

End-to-End Message Bundling

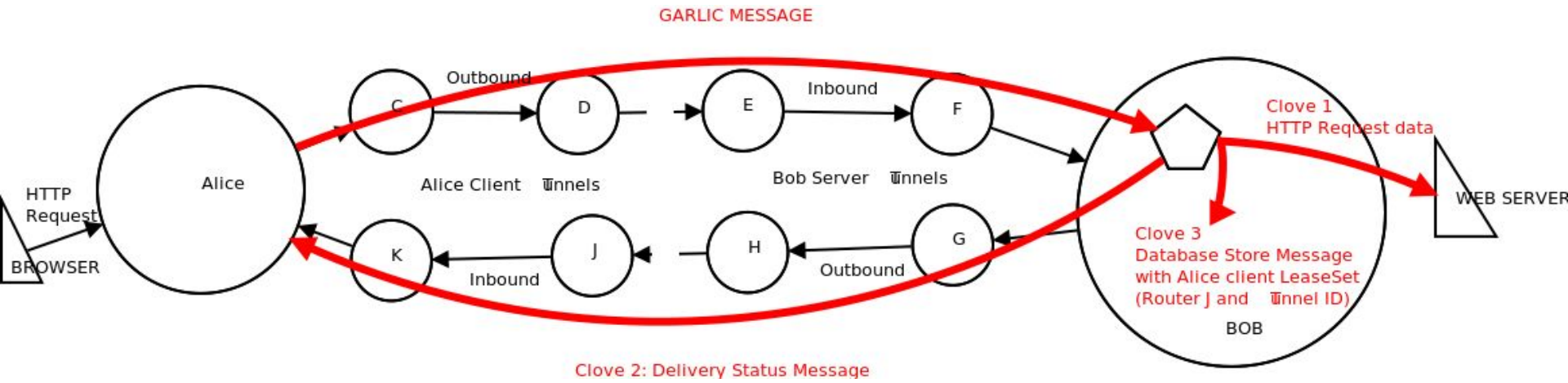
At the layer above tunnels, I2P delivers end-to-end messages between Destinations. Just as within a single tunnel, we use ElGamal/AES+SessionTag for the encryption. Each client message as delivered to the router through the I2CP interface becomes a single Garlic Clove with its own Delivery Instructions, inside a Garlic Message. Delivery Instructions may specify a Destination, Router, or Tunnel.

Generally, a Garlic Message will contain only one clove. However, the router will periodically bundle two additional cloves in the Garlic Message:



1. A Delivery Status Message, with Delivery Instructions specifying that it be sent back to the originating router as an acknowledgment. This is similar to the "reply block" or "reply onion" described in the references. It is used for determining the success or failure of end to end message delivery. The originating router may, upon failure to receive the Delivery Status Message within the

https://geti2p.net/en/docs/how/garlic-routing



I2P Hidden Services

Name	Type	Points at	Preview	Status	Control
A01 Omni*Web HTTPS	Standard server	127.0.0.1:443	No Preview		Stop
Destination: yo6sgmfq7pfvvp2e4kcuhjtfg7wfltt63igwcukhbmuqm6lu3a3a.b32.i2p **Description:** A01 Omni*Web HTTPS					
HTTP Phos tunnel	HTTP server	127.0.0.1:80	Preview		Stop
Destination: weor7pxsuxpkkbvh4sgta7hvoom5jrzemvy253y5norcamadujaq.b32.i2p **Description:** HTTP Phos tunnel					
I2P HTTPS Tunnel	HTTP server	127.0.0.1:7668	Preview		Stop
Destination: 6fjdbkxk7z3ykh5fauag3gopui7sx4sdzererl7jvsrdivsboqra.b32.i2p **Description:** I2P HTTPS Tunnel					
I2P webserver	HTTP server	127.0.0.1:7658	No Preview		Start
Hostname: mysite.i2p **Description:** My eepsite					

Part C: Omnihash & Omni*Shell

– Hash of Everything

https://en.bitcoin.it/wiki/Technical_background_of_version_1_Bitcoin_addresses

Technical background of version 1 Bitcoin addresses

Main page
Bitcoin FAQ
Editing help
Forums
Chatrooms
Recent changes
Page index

Tools

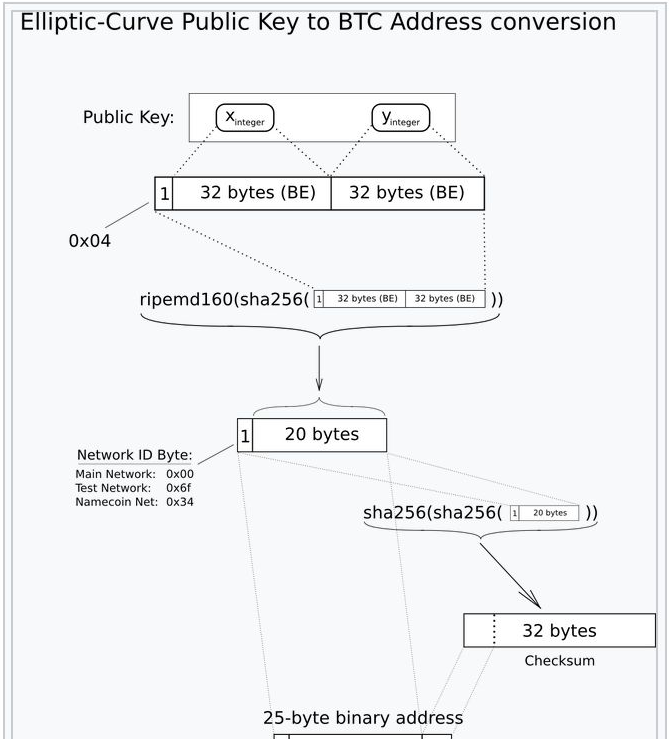
What links here
Related changes
Special pages
Printable version
Permanent link
Page information

Sister projects

This article may be too technical for some users. The more basic article on Bitcoin Addresses may be more appropriate.

A Bitcoin address is a 160-bit hash of the public portion of a public/private ECDSA keypair. Using public-key cryptography, you can "sign" data with your private key and anyone who knows your public key can verify that the signature is valid.

A new keypair is generated for each receiving address (with newer HD wallets, this is done deterministically). The public key and their associated private keys (or the seed needed to generate them) are stored in the wallet data file. This is the only file users should need to backup. A "send" transaction to a specific Bitcoin address



Activities Firefox Web Browser Ahd 29 Sep, 20:11 zh

Omni*Web: Def'n Ownership of Digital Assets w' Ownership JSON, bs'd on decentra...

https://8a75-2001-d08-da-5986-f093-ef4e-d34-4828.ngrok-free.app//oxw/oxw.php?nn=adam 150%

https://www.youtube.com/watch?v=FofGUYJS7js

Omni*Web Login Demo 2024-08

omn

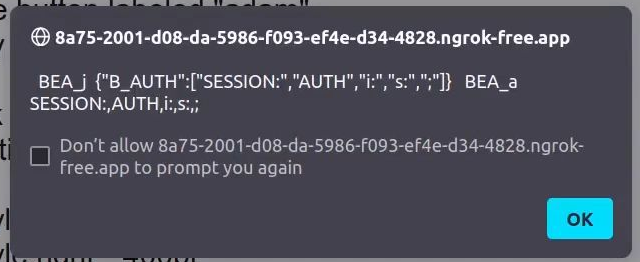
1. C
Ser

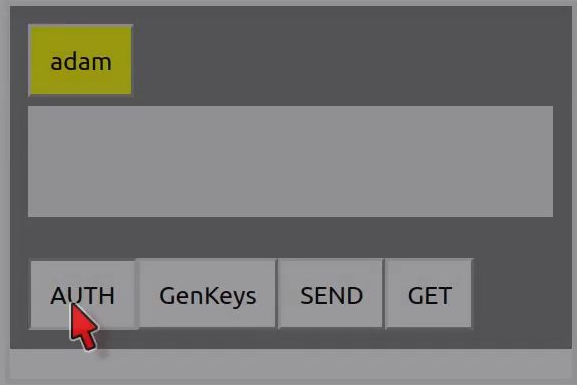
2. Click GenKeys to generate private-public keypair. Then click AUTH again to initiate Authentication.
Server will return Public Key PBK and browser will display it.

3. Enter "myhash e:" and click on the Nickname button labeled "adam"
The server will return the hash of your pulic key

4. Press F12 to open the Developer Tools, click
Enter the following command to adjust the positi

```
document.querySelectorAll('.chat-popup')[1].styl
document.querySelectorAll('.chat-popup')[1].style.right = "400pt"
document.querySelectorAll('.chat-popup')[1].style.display="none"
document.querySelectorAll('.chat-popup')[1].style.display="block"
```





fgl_s 333 < 2 > array (0 => array (0 => '{"B_AUTH":["SESSION:","AUTH","i:","s:",";"]}', 1 => 'jd:', 2 => 'cdw:', 3 => 'am:', 4 => 'lcdw:', 5 => 'B_AUTH',), 1 => NULL,)

4:01 / 9:36

omnixtar.github.io Home Page

1. Click AUTH button to initiate Authentication.
Server will return Public Key PBK = null if private-public keypair is not initialised.

2. Click GenKeys to generate private-public keypair. Then click AUTH again to initiate Authentication.
Server will return Public Key PBK and browser will display it.

3. Enter "myhash e:" and click on the Nickname button labeled "adam".
The server will return the hash of your pulic key (HPBK).

4. Press F12 to open the Developer Tools, click Console tab to open browser console.
Enter the following command to adjust the position and visibility of the dialog box:

```
document.querySelectorAll('.chat-popup')[1].style.bottom="200pt"
document.querySelectorAll('.chat-popup')[1].style.right="400pt"
document.querySelectorAll('.chat-popup')[1].style.display="none"
document.querySelectorAll('.chat-popup')[1].style.display="block"
```

fgl_s 333 < 2 > array (0 => array (0 => '{"B_AUTH":["SESSION:","AUTH","i:","s:",";"]}', 1 => 'jd:', 2 => 'cdw:', 3 => 'am:', 4 => 'lcdw:', 5 => 'B_AUTH',), 1 => NULL,)
fgl_s 333 < 2 > array (0 => array (0 => '{"B_AUTH":["SESSION:","AUTH","i:","s:",";"]}', 1 => 'jd:', 2 => 'cdw:', 3 => 'am:', 4 => 'lcdw:', 5 => 'B_AUTH',), 1 => array ('SP' => '6c1c1397', 'PASS' => '20240929_201140', 'PBK' => 'LS0tLS1CRUdJTiBQVUJMSUMgS0VZLS0tLS0KTUlJQ0lqQU5CZ2txaGtpRzl3MEJBUUVGQUFPQ0FnOEFNSUlDQ2dLQ0FnRUF3WkVQamFFNN' 'NN' => 'adam',),)
fgl_s 333 < 2 > array (0 => array (0 => '{"B_AUTH1":["SESSION:","AUTH","i:","PBK","i:","o_hash.json","l_cdwjs","h_b64","s:",";"]}', 1 => 'jd:', 2 => 'cdw:', 3 => 'am:', 4 => 'lcdw:', 5 => 'B_AUTH1',), 1 => 'Graph/dmeta/HnzRl9nOdQ==',)

Activities Firefox Web Browser Ahd 29 Sep, 20:12 zh

8a75-2001-d08-da-5986-f

https://8a75-2001-d08-da-5986-f093-ef4e-d34-4828.ngrok-free.app//oxw/oxw.php?nn=adam 150%

omnixtar.github.io Home Page

1. Click AUTH button to initiate Authentication.
Server will return Public Key PBK = null if private-public keypair is not initialised.

2. Click GenKeys to generate private-public keypair. Then click AUTH again to initiate Authentication.
Server will return Public Key PBK and browser will display it.

3. Enter "myhash e:" and click on the Nickname button labeled "adam".
The server will return the hash of your pulic key (HPBK).

4. Press F12 to open the Developer Tools, click Console tab to open browser console.
Enter the following command to adjust the position and visibility of the dialog box:

```
document.querySelectorAll('.chat-popup')[1].style.bottom="200pt"
document.querySelectorAll('.chat-popup')[1].style.right="400pt"
document.querySelectorAll('.chat-popup')[1].style.display="none"
document.querySelectorAll('.chat-popup')[1].style.display="block"
```

adam

myhash e:

AUTH GenKeys SEND GET

fgl_s 333 < 2 > array (0 => array (0 => '{"B_AUTH":["SESSION:","AUTH","i:","s:",";"]}', 1 => 'jd:', 2 => 'cdw:', 3 => 'am:', 4 => 'lcdw:', 5 => 'B_AUTH',), 1 => NULL,)
fgl_s 333 < 2 > array (0 => array (0 => '{"B_AUTH":["SESSION:","AUTH","i:","s:",";"]}', 1 => 'jd:', 2 => 'cdw:', 3 => 'am:', 4 => 'lcdw:', 5 => 'B_AUTH',), 1 => array ('SP' => '6c1c1397', 'PASS' => '20240929_201140', 'PBK' => 'LS0tLS1CRUdJTiBQVUJMSUMgS0VZLS0tLS0KTUlJQ0lqQU5CZ2txaGtpRzl3MEJBUUVGQUFPQ0FnOEFNSUlDQ2dLQ0FnRUF3WkVQamF
'NN' => 'adam',),)
fgl_s 333 < 2 > array (0 => array (0 => '{"B_AUTH1":["SESSION:","AUTH","i:","PBK","i:","o_hash.json","l_cdwjs","h_b64","s:",";"]}', 1 => 'jd:', 2 => 'cdw:', 3 => 'am:', 4 => 'lcdw:', 5 => 'B_AUTH1',), 1 => 'Graph/dmeta/HnzRl9nOdQ==',)


Omni*Web: Def'n Ownership of Digital Assets w Ownership JSON, bs'd on decentra...
// commands to create Ownership JSON
jb={owner: "HnzRl9nOdQ=="}
jb.nick='adam'
jb.jstr='{"phone":123,"email":"bac@abc","name":"beta"}'
jb.hjs=bnToB64(cyrb53(jb.jstr))
// Ownership JSON object in console
jb
Object {
owner: "HnzRl9nOdQ==",
nick: "adam",
jstr: '{"phone":123,"email":"bac@abc","name":"beta"}',
hjs: "GMpBTscsMw=="
}
// JSON string ownership JSON
JSON.stringify(jb)
'{"owner":"HnzRl9nOdQ==","jstr":"{\\"phone\\":123,\\"email\\":\\"bac@abc\\",\\"name\\":\\"beta\\"}
","hjs":"GMpBTscsMw=="}'
// // JSON string ownership JSON, prettified
'{
"owner":"HnzRl9nOdQ==",
"jstr":"{\\"phone\\":123,\\"email\\":\\"bac@abc\\",\\"name\\":\\"beta\\"}",
"hjs":"GMpBTscsMw=="
1:26 / 9:36
JavaScript
Tab Width: 2
Ln 16, Col 56
INS

```
jd.jstr="yo6sgmfq7pfvvp2e4kcuhjtfg7wfltt63igwcukhbmuqm6lu3a3a.b32.i2p"
"yo6sgmfq7pfvvp2e4kcuhjtfg7wfltt63igwcukhbmuqm6lu3a3a.b32.i2p"

jd
▶ Object { owner: "LlwHci_t", nick: "donald", jstr:
"yo6sgmfq7pfvvp2e4kcuhjtfg7wfltt63igwcukhbmuqm6lu3a3a.b32.i2p", date: "2024-10-18T15:18:49.260Z", hjs:
"Bw7BA+t48Q==" }

hjs=bnToB64(cyrb53(jd.jstr))
"A0tObR27Ow=="

jd
▼ Object { owner: "LlwHci_t", nick: "donald", jstr:
"yo6sgmfq7pfvvp2e4kcuhjtfg7wfltt63igwcukhbmuqm6lu3a3a.b32.i2p", date: "2024-10-18T15:18:49.260Z", hjs:
"Bw7BA+t48Q==" }
    date: "2024-10-18T15:18:49.260Z"
    hjs: "Bw7BA+t48Q=="
    jstr: "yo6sgmfq7pfvvp2e4kcuhjtfg7wfltt63igwcukhbmuqm6lu3a3a.b32.i2p"
    nick: "donald"
    owner: "LlwHci_t"
  ▶ <prototype>: Object { … }
```

Developer Tools — Omni*Contract: Decentralized Meta Contact — https://omnixtar.github.io/omnicontract/

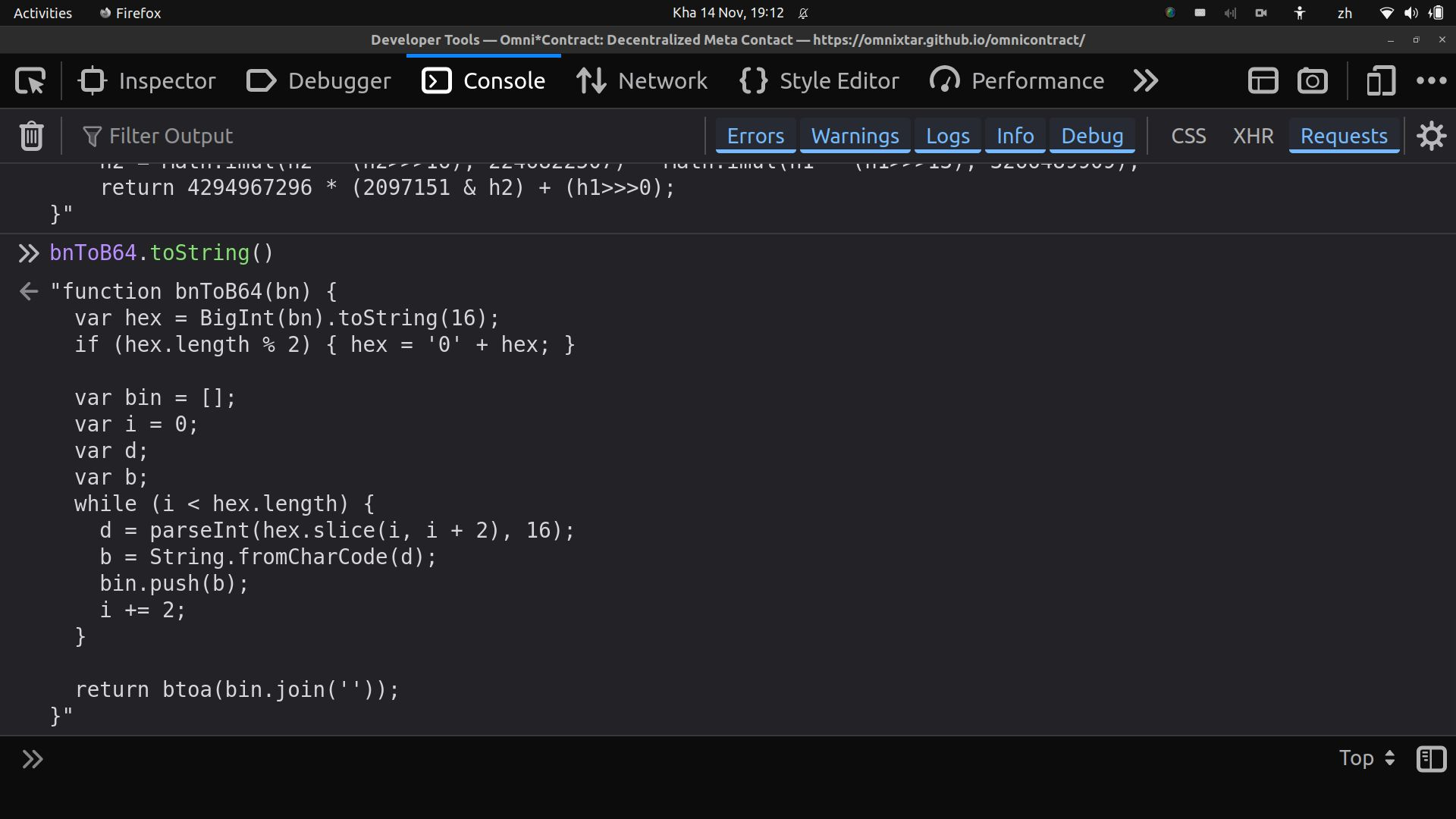
Inspector Debugger Console Network Style Editor Performance

Filter Output | Errors | Warnings | Logs | Info | Debug | CSS | XHR | Requests

```
>> cyrb53.toString()
<- "function(str, seed = 0) {
    let h1 = 0xdeadbeef ^ seed, h2 = 0x41c6ce57 ^ seed;
    for (let i = 0, ch; i < str.length; i++) {
        ch = str.charCodeAt(i);
        h1 = Math.imul(h1 ^ ch, 2654435761);
        h2 = Math.imul(h2 ^ ch, 1597334677);
    }
    h1 = Math.imul(h1 ^ (h1>>>16), 2246822507) ^ Math.imul(h2 ^ (h2>>>13), 3266489909);
    h2 = Math.imul(h2 ^ (h2>>>16), 2246822507) ^ Math.imul(h1 ^ (h1>>>13), 3266489909);
    return 4294967296 * (2097151 & h2) + (h1>>>0);
}"
>> bnToB64.toString()
<- "function bnToB64(bn) {
  var hex = BigInt(bn).toString(16);
  if (hex.length % 2) { hex = '0' + hex; }

  var bin = [];
  var i = 0;
```

Top


Activities
Firefox
Kha 14 Nov, 19:12
zh
Developer Tools — Omni*Contract: Decentralized Meta Contact — https://omnixtar.github.io/omnicontract/
Inspector
Debugger
Console
Network
Style Editor
Performance
Filter Output
Errors
Warnings
Logs
Info
Debug
CSS
XHR
Requests
return 4294967296 * (2097151 & h2) + (h1>>>0);
}"
bnToB64.toString()
"function bnToB64(bn) {
var hex = BigInt(bn).toString(16);
if (hex.length % 2) { hex = '0' + hex; }
var bin = [];
var i = 0;
var d;
var b;
while (i < hex.length) {
d = parseInt(hex.slice(i, i + 2), 16);
b = String.fromCharCode(d);
bin.push(b);
i += 2;
}
return btoa(bin.join(''));
}"
Top

Inspector Debugger Console Network Style Editor Performance 1

Filter Output | Errors | Warnings | Logs | Info | Debug | CSS | XHR | Requests

"Bw7BA+t48Q==" }

```
jd
```

Object { owner: "LlwHci_t", nick: "donald", jstr: "yo6sgmfq7pfvvp2e4kcuhjtfg7wfltt63igwcukhbmuqm6lu3a3a.b32.i2p", date: "2024-10-18T15:18:49.260Z", hjs: "Bw7BA+t48Q==" }

```
f_add.toString()
```

```
"function(){s.push(parseInt(s.pop())+parseInt(s.pop()))}"
```

```
f_h5364=function(){s.push(bnToB64(cyrb53(JSON.stringify(s.pop()))))}
```

```
function f_h5364()
```

```
s
```

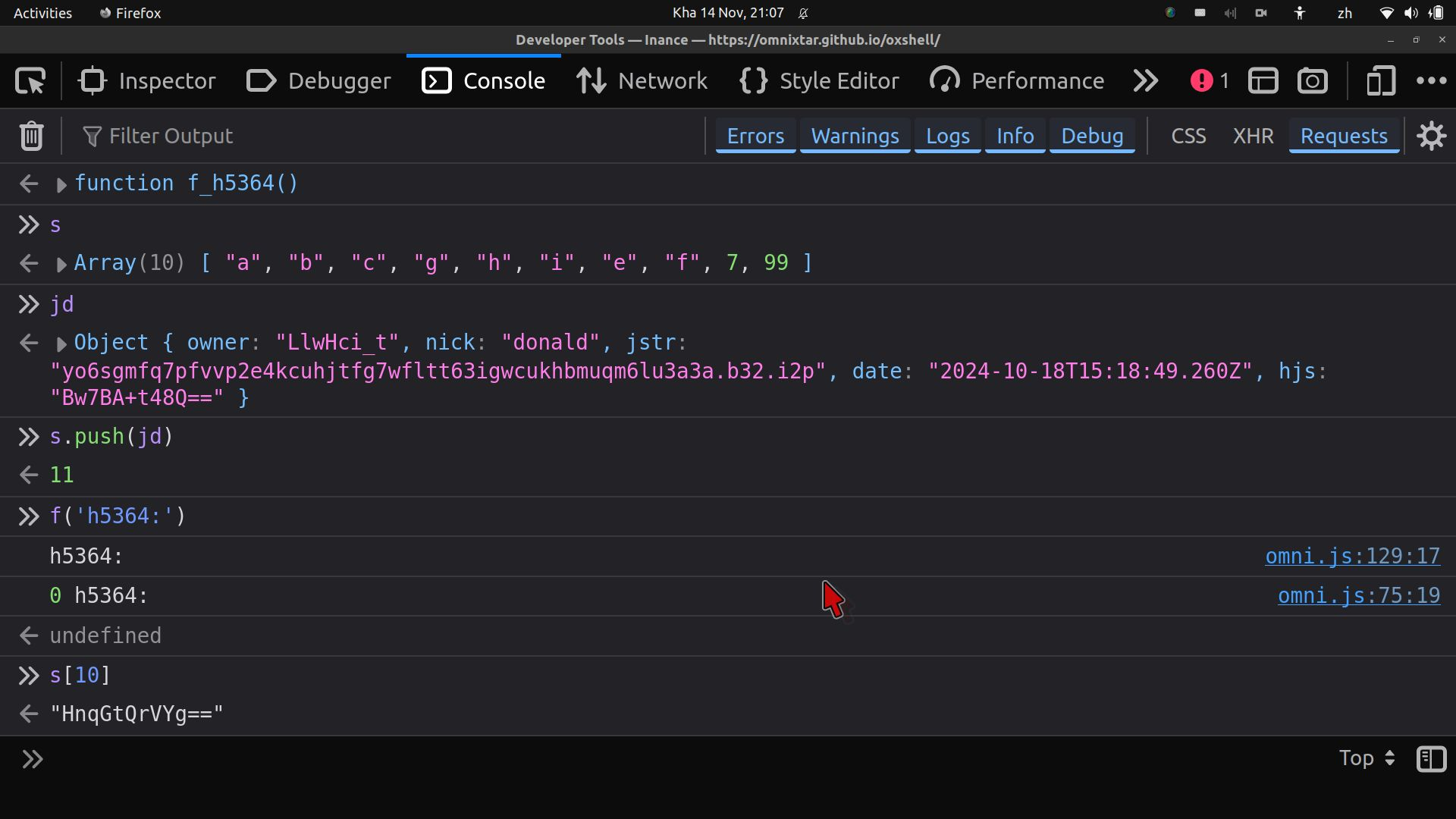
```
Array(10) [ "a", "b", "c", "g", "h", "i", "e", "f", 7, 99 ]
```

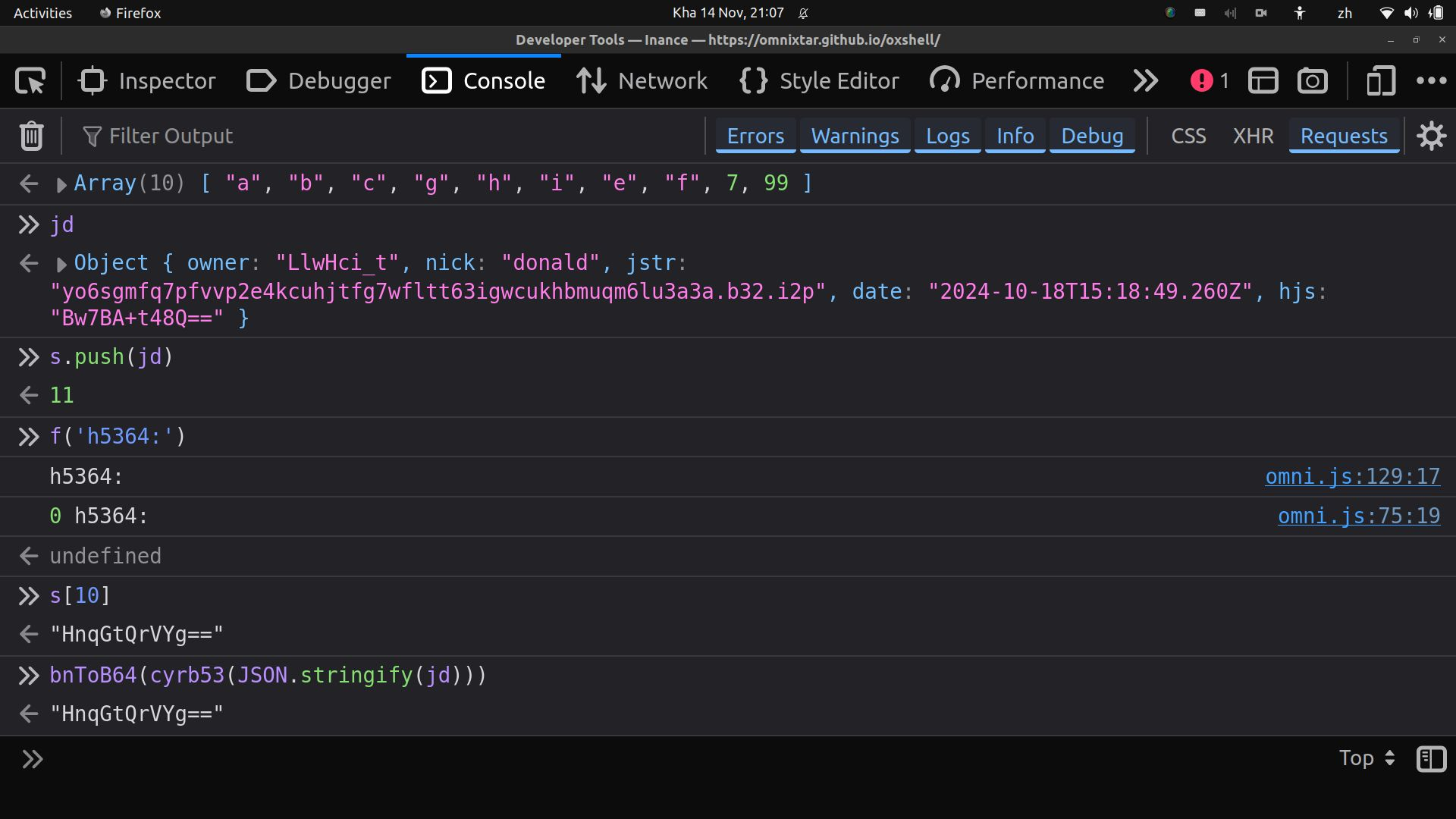
```
jd
```

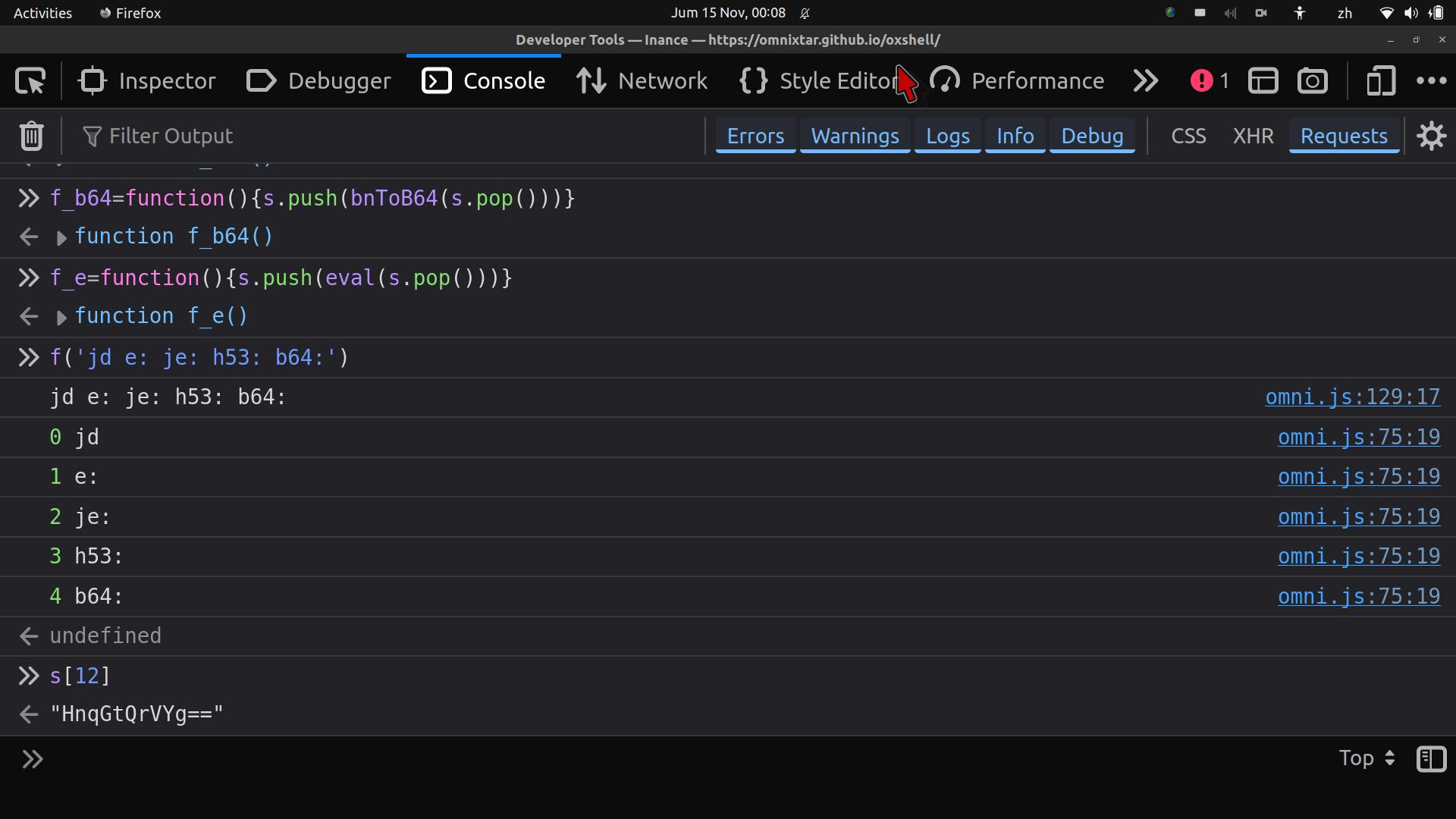
Object { owner: "LlwHci_t", nick: "donald", jstr: "yo6sgmfq7pfvvp2e4kcuhjtfg7wfltt63igwcukhbmuqm6lu3a3a.b32.i2p", date: "2024-10-18T15:18:49.260Z", hjs: "Bw7BA+t48Q==" }

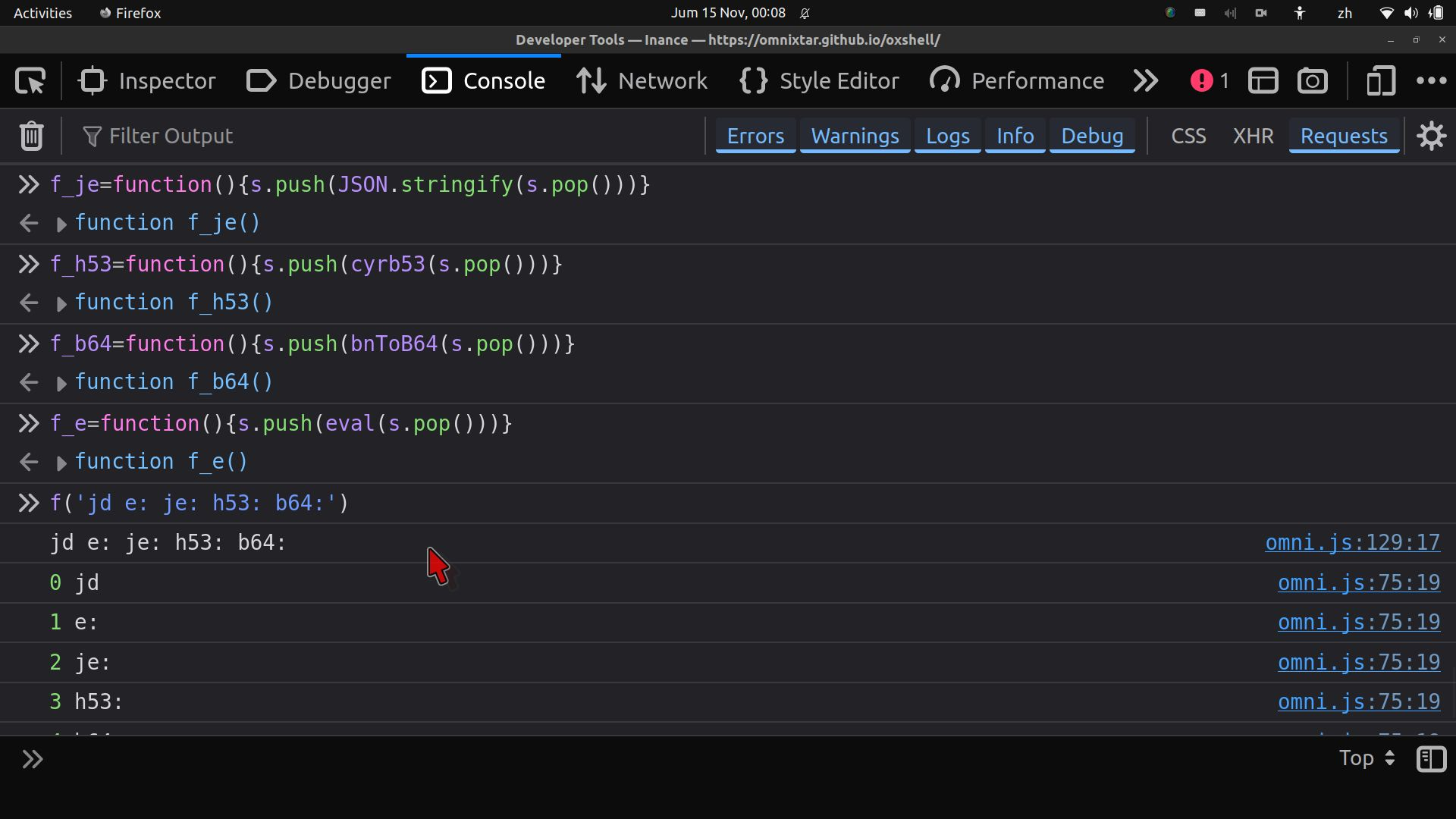
```
s.push(id)
```

Top


Activities
Firefox
Kha 14 Nov, 21:07
zh
Developer Tools — Inance — https://omnixtar.github.io/oxshell/
Inspector
Debugger
Console
Network
Style Editor
Performance
1
Filter Output
Errors
Warnings
Logs
Info
Debug
CSS
XHR
Requests
function f_h5364()
s
Array(10) ["a", "b", "c", "g", "h", "i", "e", "f", 7, 99]
jd
Object { owner: "LlwHci_t", nick: "donald", jstr: "yo6sgmfq7pfvvp2e4kcuhjtfg7wfltt63igwcukhbmuqm6lu3a3a.b32.i2p", date: "2024-10-18T15:18:49.260Z", hjs: "Bw7BA+t48Q==" }
s.push(jd)
11
f('h5364:')
h5364:
omni.js:129:17
0 h5364:
omni.js:75:19
undefined
s[10]
"HnqGtQrVYg=="
Top


Activities
Firefox
Kha 14 Nov, 21:07
zh
Developer Tools — Inance — https://omnixtar.github.io/oxshell/
Inspector
Debugger
Console
Network
Style Editor
Performance
1
Filter Output
Errors
Warnings
Logs
Info
Debug
CSS
XHR
Requests
Array(10) ["a", "b", "c", "g", "h", "i", "e", "f", 7, 99]
jd
Object { owner: "LlwHci_t", nick: "donald", jstr: "yo6sgmfq7pfvvp2e4kcuhjtfg7wfltt63igwcukhbmuqm6lu3a3a.b32.i2p", date: "2024-10-18T15:18:49.260Z", hjs: "Bw7BA+t48Q==" }
s.push(jd)
11
f('h5364:')
h5364:
omni.js:129:17
0 h5364:
omni.js:75:19
undefined
s[10]
"HnqGtQrVYg=="
bnToB64(cyrb53(JSON.stringify(jd)))
"HnqGtQrVYg=="
Top


Activities
Firefox
Jum 15 Nov, 00:08
zh
Developer Tools — Inance — https://omnixtar.github.io/oxshell/
Inspector
Debugger
Console
Network
Style Editor
Performance
1
Filter Output
Errors
Warnings
Logs
Info
Debug
CSS
XHR
Requests
f_b64=function(){s.push(bnToB64(s.pop()))}
function f_b64()
f_e=function(){s.push(eval(s.pop()))}
function f_e()
f('jd e: je: h53: b64:')
jd e: je: h53: b64:
omni.js:129:17
0 jd
omni.js:75:19
1 e:
omni.js:75:19
2 je:
omni.js:75:19
3 h53:
omni.js:75:19
4 b64:
omni.js:75:19
undefined
s[12]
"HnqGtQrVYg=="
Top


Activities
Firefox
Jum 15 Nov, 00:08
zh
Developer Tools — Inance — https://omnixtar.github.io/oxshell/
Inspector
Debugger
Console
Network
Style Editor
Performance
1
Filter Output
Errors
Warnings
Logs
Info
Debug
CSS
XHR
Requests
f_je=function(){s.push(JSON.stringify(s.pop()))}
function f_je()
f_h53=function(){s.push(cyrb53(s.pop()))}
function f_h53()
f_b64=function(){s.push(bnToB64(s.pop()))}
function f_b64()
f_e=function(){s.push(eval(s.pop()))}
function f_e()
f('jd e: je: h53: b64:')
jd e: je: h53: b64:
omni.js:129:17
0 jd
omni.js:75:19
1 e:
omni.js:75:19
2 je:
omni.js:75:19
3 h53:
omni.js:75:19
Top

```
58  phosinit=function(){
59  Phos=function(){
60      // var S=[] // this is local, not accessible outside
61      $ = this // macro
62      this.S=[]
63      var S=this.S // still need var S for local code access
64      S[0]={}
65      var S0=S[0]
66      var $CDW = {}
67      S[0].$CDW = $CDW;
68      S0.skip = 0;
69      S0.CDW = [];
70      S0.dlb = {};
71      var FGLA = function($WA) {
72      // arguments[0].split(' ').map(e=>{
73      var c_cdw=false; var i=0, ic, W=$WA;
```

```
58  phosinit=function(){
69      S0.CDW = [];
70      S0.dlb = {};
71      var FGLA = function($WA) {
72      // arguments[0].split(' ').map(e=>{
73      var c_cdw=false; var i=0, ic, W=$WA;
74      $WA.map(e=>{
75            console.log(i, e);
76            var $v=e, $vk=i; $l=$v.length;
77            if (!c_cdw && $v==':') {
78               c_cdw=true;
79               console.log('  CDW start ', W[i+1])
80               ic = i+2; // start index of CDW
81               CDN=W[i+1]
82               $CDW[CDN]=[]
83            }
```

```
58    phosinit=function(){
71        var FGLA = function($WA) {
110                   S0.CDW.pop();
111                   console.log(1183, 'after FGLA', JSON.stringify(S0.CDW));
112                   S0.cda = end(S0.CDW);
113               }
114               else if ($v[$l - 1] == ':') { // colon suffix word after symbol else : will fail
115                 var $fn = $v.substr(0, $l-1);
116                 if (typeof eval("f_"+$fn)!=="undefined")
117                   // console.log('is func', $v, typeof eval("f_"+$fn))
118                   eval("f_"+$fn+"()")
119               } else s.push(e)
120             }
121             i++
122           }
123           )
```

Developer Tools — Inance — https://omnixtar.github.io/oxshell/

Inspector Debugger Console Network Style Editor Performance 1

Filter Output | Errors | Warnings | Logs | Info | Debug | CSS | XHR | Requests

```
11 d                                                    omni.js:75:19
1176 before FGLA [["d",11,{}]]                          omni.js:107:25
0 e                                                     omni.js:75:19
1 f                                                     omni.js:75:19
1183 after FGLA []                                      omni.js:111:25
← undefined
» f('3 4 +')
3 4 +                                                   omni.js:129:17
0 3                                                     omni.js:75:19
1 4                                                     omni.js:75:19
2 +                                                     omni.js:75:19
← undefined
» s
← ▶ Array(9) [ "a", "b", "c", "g", "h", "i", "e", "f", 7 ]
» omnistart()
```

Top

Developer Tools — Inance — https://omnixtar.github.io/oxshell/

Inspector Debugger Console Network Style Editor Performance 1

Filter Output | Errors | Warnings | Logs | Info | Debug | CSS | XHR | Requests

```
2 +                                                    omni.js:75:19
← undefined
» s
← ▶ Array(9) [ "a", "b", "c", "g", "h", "i", "e", "f", 7 ]
» f_add.toString()
← "function(){s.push(parseInt(s.pop())+parseInt(s.pop()))}"
» f('44 55 add:')
  44 55 add:                                           omni.js:129:17
  0 44                                                 omni.js:75:19
  1 55                                                 omni.js:75:19
  2 add:                                               omni.js:75:19
← undefined
» s
← ▶ Array(10) [ "a", "b", "c", "g", "h", "i", "e", "f", 7, 99 ]
```

Top

```
gunzipinit=function(){gunzip=e=>{const r=Uint8Array.from(atob(e),(e=>e.charCodeAt(0))),n=new
DecompressionStream("gzip"),t=n.writable.getWriter();return t.write(r),t.close(),new
Response(n.readable).arrayBuffer().then((function(e){return(new
TextDecoder).decode(e)}))},getgzfn_js=async function(s){var s1=await
gunzip(s),j3=JSON.parse(s1),j4=JSON.parse(j3.s);return eval(j4.n+"="+j4.s),j3}}
omnigz="H4sIAAAAAAAAA8UZa1PbSPIz/hUdFxdrYqFY5o1PbBEI2aQg2Ttyy255vdRYGtkDsuSSxjzC8t+ve2YkyzbkNp/
OH0Dq6e7pd/eM4lkaKpmlINMwmUXCmeUJg8cGwC3PoYAAoiycTUSqvDAXXIn3iaA3p1mEuZyqJushbuEVQh0plcvhTAmnqR
6moulCU4l79faa3/LvIRd5iLi0L61W2w2z6MHj06lIo+OxTCKnwPWnRuPtW5jciFueXCVOM5ukcsyLcZMR/GTy8WP3g5/c3
QVBo1wKYquiwx4b4UM+3N6cgwqVu1AIEaGiHUZqAyRCwdgnwH0keDQUIoY/NZIL466Gb/nhTii2dy28p+niLAeHiCXhuBCO
e/j4T8A9vESkI0Xv7baxrvmFY0Sl9XDM8+MsEkfKkaxXrWsxzrkae3IySxx8/RNpXOjubG9tbW7v7vh15O4ictci+9v7u5u
bWzu7uxb5qfESb/x7eHjo7zDcoru1s9ftbnd2GS4sscW/hLeJeJvdnZ2tvf39zr7l/pwYFv+7fGv7P8s3F2qWp7DV3d/a39
nt7u/AG3C6nf1df9uH17gtg7Zl0NGRMky/Zu92tipngwU4w5SVAT4W9yjtOzn6iCGNcE9lFxiY6chBYWlfGSNPcW8dCP8A3
ObRkrU6LdwSn3vapMRvKFNc6A969l2HQvkSlQ9DerjDoBbgUITMNyiDg+JxyvNCkFy0XCQyFI50kWMbZUCv7li7DBHVyOzF
eTY5tpHkROW6TL3prBg7QwtADgF06ZmktmYdqow7hHqdydRptZg2YZlo84Q5/cweAVPtzVUCE8HTApIsHVHsT2Ao8J+AiUx
lLEVklb32g8f04PQzbXbte0VALB18qlubmUUVOKm4gxOsMw6j5Y8XX2oYml03eCwOPl18+ewVekXGD8iNueOD0r8myZ1VHM
aMzmGWFlkivCQbOUtEp5/bzaDZJkERfW6fGhztgmWI6s0SbdFhugw1P5zsHclf99NoFARNWKxYo1n6TU7RRMrUrJ9/u3t7c
x7df0PcxnyxXrUMNBDB4SNJriAP/iNTtXeU5/xBO93hKhs6grkOIol6NUGRmJsG2qoizCbTXBQFskWrCj5BcZBzk7kqSL27
XCo+RLOMhLrEZ5E7rGfVV3pVODmiemGSFegfl5j+WxRTlEk4qYcMI6JnHifB3s3imFh4aixSx6n0EezRMNWu/oodggSLRM6
8SD8gxhNjTy6KMfoWp1fXRcCLhzSEedVmj7o7+QG/41KBMRCC3evNQLtd545TYFhcb9Uh15vo11IpE4hbXmpcu+VpBk/LPQ
a7VajusSfNjMfeD8/3Pv32aaT+FWB3FMPZaCRyzQxI/IPNzQPfb8yp6q4ksYWb9m4zGUEnCII7mUbZnVe8fu2Uj0F/wHqEq
KrVXtUVeRS9v8WHM1kokYr8pxdXnCYGCzVgLbk7d0DGHnkicmzhv2ez1q0ATBFsgCqDDJstWCogMmi2M3S3RLa/tZu1t9+Z
K4Jqa2FmglOMxF+wfCinInJrBMqUIUHxOEXUKUqgpgK5Ci/DSSD/+ev5GXPrydkspp2mmyKFrY+HHTRU86AZBGm/hG34g5/
OvVMnZQfCk2lqGAVIlXkY7mSTExHzWaIcDCz3lc8O5kZTiodjbTccJtIXLPb3DYapZH2m9/VMpP3Kk5kIXvkU1o3GdJwVyy
ne+AWBCwCq1BhsFAcXGBH0rMayAEkVN+SJC2mmqIch5TNmHaAJCxkJTbiOPUFjUyDzMM80lCAeMdNvhrOBEV6hZJJASrOQX
```

Omni*Chat -- Liang Ng -- 2024-04-27

https://github.com/omnixtar/omnixchat/blob/master/core/src/com/biglybt/plugin/net/buddy/ 200%

Files | master | omnixchat / core / src / com / biglybt / plugin / net / bu / BuddyPluginBeta.java

Code | Blame | Raw

```
2292      logMessage(
2335          Stack<Object> stackAny = new Stack<>();
2336          Stack<HashMap<String,String>> stackHM = new Stack<>();
2337          // Stack<> stackAll = new Stack<>();
2338          String expr = msg;
2339
2340          System.out.println(expr);
2341          System.out.println("Input\tOperation\tStack after");
2342
2343          // Phoscript (Phos) Engine
2344          for (String token : expr.split("\\s+")) {
2345            System.out.print(token + "\t");
2346            switch (token) {
2347              case "+":
2348                System.out.print("Operate\t\t");
2349                stack.push(stack.pop() + stack.pop());
2350                break;
2351              case "-":
2352                System.out.print("Operate\t\t");
```

```
ECHO {"a":"b","k":"j","d":"c","n":"m","dbl":"221265.0"}
ECHO {"a":"b","k":"j","d":"c","n":"m","dbl":"3399.0"}
[Mon 6pm] EE6A59
ECHO {"a":"b","k":"j","d":"c","n":"m","dbl":"3326.0"}
ECHO {"a":"b","k":"j","d":"c","n":"m","dbl":"3329.0"}
ECHO {"a":"b","k":"j","d":"c","n":"m","dbl":"3330.0"}
[13:04] BDF12C
OMNI jo: gc: dup: hm: b a put: c d put: j k put: m n put: 1130 1300 +
[13:04] EE6A59
ECHO {"a":"b","k":"j","d":"c","n":"m","dbl":"2430.0"}
[13:52] EE6A59
magnet:?xt=urn:btih:00af0929765f52687c8834fb9415d042fcfa77
[23:23] EE6A59
ECHO {"a":"b","k":"j","d":"c","n":"m","dbl":"3326.0"}
ECHO {"a":"b","k":"j","d":"c","n":"m","dbl":"3329.0"}
ECHO {"a":"b","k":"j","d":"c","n":"m","dbl":"3330.0"}
ECHO {"a":"b","k":"j","d":"c","n":"m","dbl":"2430.0"}
[16:16] BDF12C
OMNI jo: gc: dup: hm: b a put: c d put: j k put: m n put: 1615 1300 + d2s: dbl put: je:
[16:16] EE6A59
ECHO {"a":"b","k":"j","d":"c","n":"m","dbl":"2915.0"}
```

beta chat

mous beta chat (I2P) Open

/join custom channel: key 224455 Anonymous (I2P) Open

The secret of Shell was given in the 6 Metaprogramming Magic words presented earlier "String Split Array, Loop Push Eval", and the code is within logMessage() function in BuddyPluginBeta.java, from lines 2321 to 2430. (URL given in YouTube descriptiops.)

OmniXApp-p1.mkv

5:00 / 24:32 • Coding >

00:07:32 00:05:30 -00:04:20 1/1

https://github.com/omnixtar/omnixchat/blob/master/core/src/com/biglybt/plugin/net/buddy/BuddyPluginBeta.java

```
2292            logMessage(
2333            Stack<Double> stack    = new Stack<>();
2334            Stack<String> stackStr = new Stack<>();
2335            Stack<Object> stackAny = new Stack<>();
2336            Stack<HashMap<String,String>> stackHM = new Stack<>();
2337            // Stack<> stackAll = new Stack<>();
2338            String expr = msg;
2339
2340            System.out.println(expr);
2341            System.out.println("Input\tOperation\tStack after");
2342
2343            // Phoscript (Phos) Engine
2344            for (String token : expr.split("\\s+")) {
2345              System.out.print(token + "\t");
2346              switch (token) {
2347                case "+":
2348                  System.out.print("Operate\t\t");
2349                  stack.push(stack.pop() + stack.pop());
2350                  break;
```

Super-program or Super-Shell?

Understanding of program – change after using “super shell” – metaprogramming in Phoscript.

Interaction between programs, via “super-shell” – become “super-program”.

UNIX pipe is limited to one computer.
Omnihash data + I2P = global data sharing.

Part D: “Trojan Horse Super-Shell?”

– Consolidating Users across Open Source Projects & Commercial Social Media by *adding Omni*Shell* to generate Omnihash User ID as *universal decentralised ID*.

From Technofeudalism to Technosocialism: FORTH unexpected roles in Decentralised Systems

Technofeudalism is a term coined by former Greek finance minister Yanis Varoufakis as the title of a book which compares top American technology companies to feudal lords, who literally enslaved individual users and free software programmers, exploiting their data and source code without fairly rewarding them.

Technosocialism has been used by many other authors but I use it to describe the very real but underreported progress made by free software programmers and free software, despite the lack of fair mechanisms for rewards so far. Free software programmers or technosocialists have made technofeudalism possible as a matter of fact, but they do not get to enjoy the rewards.

Bitcoin and related projects were supposed to address this problem, and are successful in their own way, so we are now half way towards technosocialism.

At section interval, recap past sections, preview this section

Trojan horse: small size of Omni*Shell, 3kb minified gzipped JavaScript.

Football Analogy: I2P (Invisible Internet Project) and Bitcoin have proven the *technical viability of fundamental Decentralised algorithms*, produced many millionaires and billionaires – carrying football past half field. (25 years, *beneficiaries are like Bankers*)

Omni*Web plan to *make more people rich – like McDonald's* vs. Hongkong & Shanghai Bank – second half of football field – next 25 years.

Existing web ecosystem – *dominated by a few huge social media applications* – difficult for small applications to grow – many reasons – *User Fragmentation.*

Solution: Omnihas User ID

Strategy: *"Trojan Horse Super-Shell"*, not the computer virus type – *security guaranteed by hash & public key cryptography*.

Omni*Shell in *JavaScript & Java* – cover most client applications in web & Android – *easily ported to other programming languages* – Apple – *server side – PHP, Python, C++* – (show screenshots)

Probability of success?

Cloning: GNU & Linux began by cloning UNIX.

Now most commercial software companies use unspecified amount of free software.

Modify free software licence – separation of disclosure & royalties.

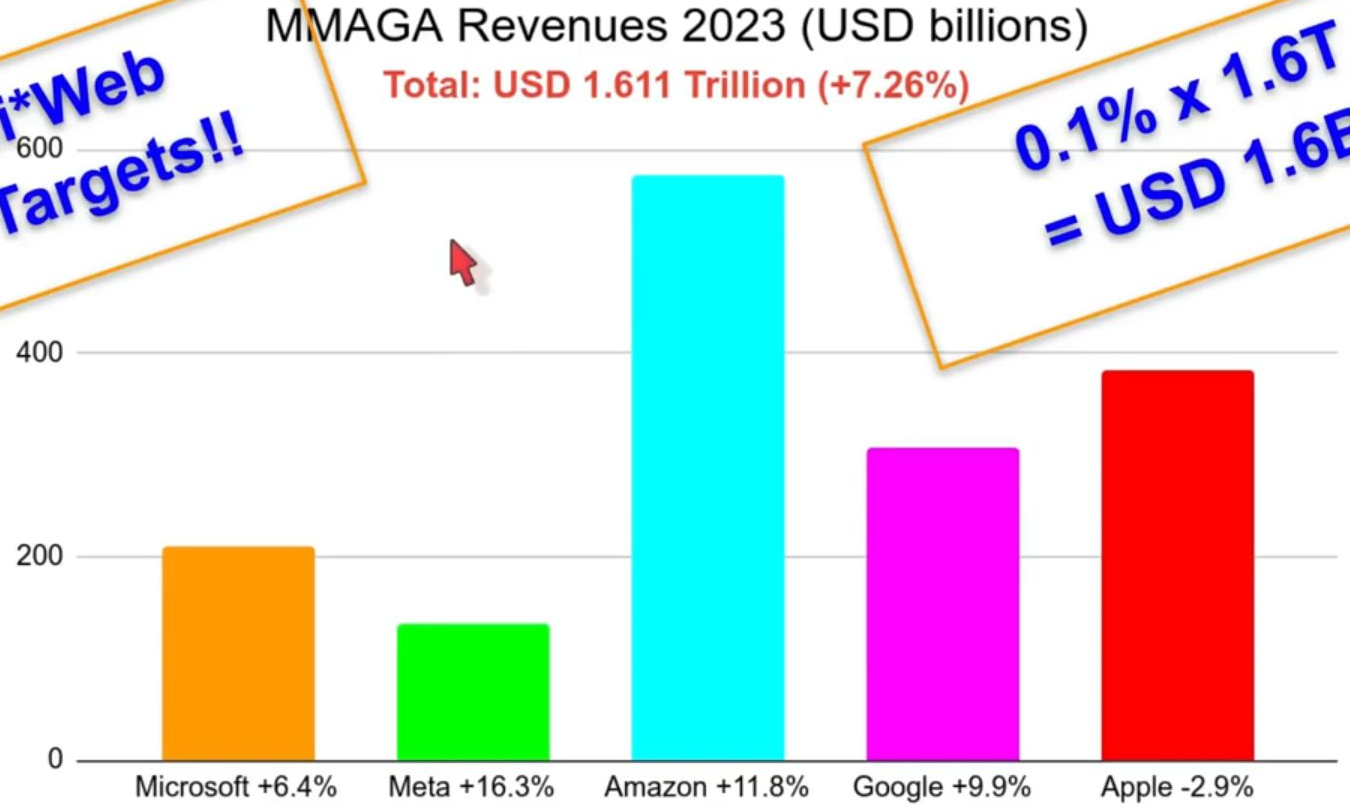
Track usage (execution) of free software using hashcodes – claim payments.

BitTorrent used to be carrying >30% of all Internet traffic.

Heavy censorship on popular platforms such as Reddit & TikTok – decentralised social media – impossible to censor – *FILTER model – users may choose various types of filters to filter unwanted text, videos, photos or user IDs* – data stored on devices owned by users – or access control belongs to users, not platform.

Omni*Web
2030 Targets!!

0.1% x 1.6T
= USD 1.6B



A literal trillion dollar question: Do you know that the combined revenues of the top 5 technology companies in 2023 already exceeded USD 1.6 trillion and are growing strongly above 5% annually?

Part E: FORTH roles in future

Bitcoin & cryptocurrencies actually made FORTH (variant) *the most valuable and popular* (by number of devices, nodes, wallets) in the world – (need decentralised systems to gather statistics!!)

Omni*Web “trojan horse” strategy will (hopefully) attract more programmers to use FORTH (Phoscript or dialects) as interface programming language, if Omni*Web plan of making *Omnihash User ID as the universal decentralised ID* works.

Metaprogramming & Mathematics are important – SymForth example.

Decentralised AI – ideas threatening commercial interests are filtered – *hash based indices allowed AI training data to be distributed in users computers & mobile devices* – much bigger (?) than commercially owned assets?

Same dilemma in history – *land grab from farmers* – now land owned by nation state or corporation – do we want this to happen again – *now we know we can prevent it?*

Although we have used ideological terms such as *techno feudalism* and *techno socialism* to express our views of reducing global rich poor gaps through Decentralised systems, the Chinese or Asian *principle of Taichi views conflicts* not as binary black versus white, but *as forces in opposition feeding on each other*, as quoted by Danish Physicist Niels Bohr in Latin as *Contraria sunt complementa* or in English *"opposites are complementary".*

This philosophy can be seen in the *Chinese Weiqi or Go chess*, where on a board with 19 by 19 lines namely 361 spots, the goal of the game is to simply *outdo the opponent by occupying at least one spot more than the opponent* with the chosen black or white pieces, *instead of killing the King* of the opponent in the European chess.

Technosocialism = *"better pay for some programmers"*, *not threatening Washington & Wall Street.*

Metanarchy = Metaverse + (Mon)-archy = self governance through metaverse