net : Make it Userfriendly

reinventing the internet

Bernd Paysan

#wefixthenet, YBTI session, 32c3, Hamburg

Outline

2.5 years after Snowden

What happend to change the world:

Politics	Manhatten project to find “the golden key”?
Users	don’t want their dick picks be watched and use DuckDuckGo and encrypted chat
Software	NSA backdoors have been refitted by attackers (Juniper)
Solutions	net2o starts to be usable (somewhat)

2.5 years after Snowden

What happend to change the world:

Politics	Manhatten project to find “the golden key”?
Users	don’t want their dick picks be watched and use DuckDuckGo and encrypted chat
Software	NSA backdoors have been refitted by attackers (Juniper)
Solutions	net2o starts to be usable (somewhat)

2.5 years after Snowden

What happend to change the world:

Politics	Manhatten project to find "the golden key"?
Users	don't want their dick picks be watched and use DuckDuckGo and encrypted chat
Software	NSA backdoors have been refitted by attackers (Juniper)
Solutions	net2o starts to be usable (somewhat)

2.5 years after Snowden

What happend to change the world:

Politics	Manhatten project to find “the golden key”?
Users	don’t want their dick picks be watched and use DuckDuckGo and encrypted chat
Software	NSA backdoors have been refitted by attackers (Juniper)
Solutions	net2o starts to be usable (somewhat)

2.5 years after Snowden

What happend to change the world:

Politics	Manhatten project to find “the golden key”?
Users	don’t want their dick picks be watched and use DuckDuckGo and encrypted chat
Software	NSA backdoors have been refitted by attackers (Juniper)
Solutions	net2o starts to be usable (somewhat)

net2o in a nutshell

net2o consists of the following 6 layers (implemented bottom up):

2. Path switched packets with 2^n size writing into shared memory buffers
3. Ephemeral key exchange and signatures with Ed25519, symmetric authenticated encryption+hash+prng with Keccak, symmetric block encryption with Threefish onion routing camouflage probably with AES
4. Timing driven delay minimizing flow control
5. Stack–oriented tokenized command language
6. Distributed data (files) and distributed metadata (prefix hash trie)
7. Apps in a sandboxed environment for displaying content

net2o in a nutshell

net2o consists of the following 6 layers (implemented bottom up):

2. Path switched packets with 2^n size writing into shared memory buffers
3. Ephemeral key exchange and signatures with Ed25519, symmetric authenticated encryption+hash+prng with Keccak, symmetric block encryption with Threefish onion routing camouflage probably with AES
4. Timing driven delay minimizing flow control
5. Stack–oriented tokenized command language
6. Distributed data (files) and distributed metadata (prefix hash trie)
7. Apps in a sandboxed environment for displaying content

net2o in a nutshell

net2o consists of the following 6 layers (implemented bottom up):

2. Path switched packets with 2^n size writing into shared memory buffers
3. Ephemeral key exchange and signatures with Ed25519, symmetric authenticated encryption+hash+prng with Keccak, symmetric block encryption with Threefish onion routing camouflage probably with AES
4. Timing driven delay minimizing flow control
5. Stack–oriented tokenized command language
6. Distributed data (files) and distributed metadata (prefix hash trie)
7. Apps in a sandboxed environment for displaying content

net2o in a nutshell

net2o consists of the following 6 layers (implemented bottom up):

2. Path switched packets with 2^n size writing into shared memory buffers
3. Ephemeral key exchange and signatures with Ed25519, symmetric authenticated encryption+hash+prng with Keccak, symmetric block encryption with Threefish onion routing camouflage probably with AES
4. Timing driven delay minimizing flow control
5. Stack–oriented tokenized command language
6. Distributed data (files) and distributed metadata (prefix hash trie)
7. Apps in a sandboxed environment for displaying content

net2o in a nutshell

net2o consists of the following 6 layers (implemented bottom up):

2. Path switched packets with 2^n size writing into shared memory buffers
3. Ephemeral key exchange and signatures with Ed25519, symmetric authenticated encryption+hash+prng with Keccak, symmetric block encryption with Threefish onion routing camouflage probably with AES
4. Timing driven delay minimizing flow control
5. Stack–oriented tokenized command language
6. Distributed data (files) and distributed metadata (prefix hash trie)
7. Apps in a sandboxed environment for displaying content

net2o in a nutshell

net2o consists of the following 6 layers (implemented bottom up):

2. Path switched packets with 2^n size writing into shared memory buffers
3. Ephemeral key exchange and signatures with Ed25519, symmetric authenticated encryption+hash+prng with Keccak, symmetric block encryption with Threefish onion routing camouflage probably with AES
4. Timing driven delay minimizing flow control
5. Stack–oriented tokenized command language
6. Distributed data (files) and distributed metadata (prefix hash trie)
7. Apps in a sandboxed environment for displaying content

net2o in a nutshell

net2o consists of the following 6 layers (implemented bottom up):

2. Path switched packets with 2^n size writing into shared memory buffers
3. Ephemeral key exchange and signatures with Ed25519, symmetric authenticated encryption+hash+prng with Keccak, symmetric block encryption with Threefish onion routing camouflage probably with AES
4. Timing driven delay minimizing flow control
5. Stack–oriented tokenized command language
6. Distributed data (files) and distributed metadata (prefix hash trie)
7. Apps in a sandboxed environment for displaying content

Objectives

net2o's design objectives are

- lightweight, fast, scalable
- easy to implement
- secure
- media capable
- works as overlay on current networks (UDP/IP), but can replace the entire stack

Objectives

net2o's design objectives are

- lightweight, fast, scalable
- easy to implement
- secure
- media capable
- works as overlay on current networks (UDP/IP), but can replace the entire stack

Objectives

net2o's design objectives are

- lightweight, fast, scalable
- easy to implement
- secure
- media capable
- works as overlay on current networks (UDP/IP), but can replace the entire stack

Objectives

net2o's design objectives are

- lightweight, fast, scalable
- easy to implement
- secure
- media capable
- works as overlay on current networks (UDP/IP), but can replace the entire stack

Objectives

net2o's design objectives are

- lightweight, fast, scalable
- easy to implement
- secure
- media capable
- works as overlay on current networks (UDP/IP), but can replace the entire stack

Objectives

net2o's design objectives are

- lightweight, fast, scalable
- easy to implement
- secure
- media capable
- works as overlay on current networks (UDP/IP), but can replace the entire stack

Basic Frameworks

PKI	Create, import, and exchange keys
Named file copy	For testing only
Vault	A container for encrypted data without metadata exposure
DHT	Query key/value pairs (keys are pubkeys or hash keys)
Chat	Instant messaging 1:1 or in chat groups
Version control system	For larger content (not yet implemented)
Sync	to synchronize your computers (RSN)
Audio/Video Chat	Real time data streaming (RSN)

Basic Frameworks

PKI	Create, import, and exchange keys
Named file copy	For testing only
Vault	A container for encrypted data without metadata exposure
DHT	Query key/value pairs (keys are pubkeys or hash keys)
Chat	Instant messaging 1:1 or in chat groups
Version control system	For larger content (not yet implemented)
Sync	to synchronize your computers (RSN)
Audio/Video Chat	Real time data streaming (RSN)

Basic Frameworks

PKI	Create, import, and exchange keys
Named file copy	For testing only
Vault	A container for encrypted data without metadata exposure
DHT	Query key/value pairs (keys are pubkeys or hash keys)
Chat	Instant messaging 1:1 or in chat groups
Version control system	For larger content (not yet implemented)
Sync	to synchronize your computers (RSN)
Audio/Video Chat	Real time data streaming (RSN)

Basic Frameworks

PKI	Create, import, and exchange keys
Named file copy	For testing only
Vault	A container for encrypted data without metadata exposure
DHT	Query key/value pairs (keys are pubkeys or hash keys)
Chat	Instant messaging 1:1 or in chat groups
Version control system	For larger content (not yet implemented)
Sync	to synchronize your computers (RSN)
Audio/Video Chat	Real time data streaming (RSN)

Basic Frameworks

PKI	Create, import, and exchange keys
Named file copy	For testing only
Vault	A container for encrypted data without metadata exposure
DHT	Query key/value pairs (keys are pubkeys or hash keys)
Chat	Instant messaging 1:1 or in chat groups
Version control system	For larger content (not yet implemented)
Sync	to synchronize your computers (RSN)
Audio/Video Chat	Real time data streaming (RSN)

Basic Frameworks

PKI	Create, import, and exchange keys
Named file copy	For testing only
Vault	A container for encrypted data without metadata exposure
DHT	Query key/value pairs (keys are pubkeys or hash keys)
Chat	Instant messaging 1:1 or in chat groups
Version control system	For larger content (not yet implemented)
Sync	to synchronize your computers (RSN)
Audio/Video Chat	Real time data streaming (RSN)

Basic Frameworks

PKI	Create, import, and exchange keys
Named file copy	For testing only
Vault	A container for encrypted data without metadata exposure
DHT	Query key/value pairs (keys are pubkeys or hash keys)
Chat	Instant messaging 1:1 or in chat groups
Version control system	For larger content (not yet implemented)
Sync	to synchronize your computers (RSN)
Audio/Video Chat	Real time data streaming (RSN)

Basic Frameworks

PKI	Create, import, and exchange keys
Named file copy	For testing only
Vault	A container for encrypted data without metadata exposure
DHT	Query key/value pairs (keys are pubkeys or hash keys)
Chat	Instant messaging 1:1 or in chat groups
Version control system	For larger content (not yet implemented)
Sync	to synchronize your computers (RSN)
Audio/Video Chat	Real time data streaming (RSN)

Try it

Linux you need:

```
git automake autoconf make gcc libtool
libltdl7 fossil
```

you run: `mkdir net2o; cd net2o`

```
wget
https://fossil.net2o.de/net2o/doc/trunk/do
chmod +x do; ./do
```

This will install some stuff and take some time (I will try to improve that).

Android Get Gforth from play store or `https://net2o.de/Gforth.apk`

Open/close (back button) Gforth if you like; then open net2o.

Try it

Linux you need:

```
git automake autoconf make gcc libtool
libltdl7 fossil
```

you run: `mkdir net2o; cd net2o`

```
wget
https://fossil.net2o.de/net2o/doc/trunk/do
chmod +x do; ./do
```

This will install some stuff and take some time (I will try to improve that).

Android Get Gforth from play store or `https://net2o.de/Gforth.apk`

Open/close (back button) Gforth if you like; then open net2o.

Try it — Generate a Key

Linux you run:
`./n2o cmd`
`keygen` *<nick>*
Enter your passphrase twice.

Android Tap on the little nettie to start the app, it will autodetect that you don't have a key generated. Enter nick and passphrase twice.

Try it — Generate a Key

Linux you run:

```
./n2o cmd
keygen <nick>
```

Enter your passphrase twice.

Android Tap on the little nettie to start the app, it will autodetect that you don't have a key generated. Enter nick and passphrase twice.

Try it — Get another Key

- To get my key, search for it (32 bit is sufficient)
  `keysearch kQusJ`
- Try to chat with me
  `chat 32c3@bernd`
- Aquire more keys by observing a group chat. List your keys with
  `n2o keylist`
  from within the chat.
- Change networks with your Android and watch that the chat still works.

Try it — Get another Key

- To get my key, search for it (32 bit is sufficient)
  `keysearch kQusJ`
- Try to chat with me
  `chat 32c3@bernd`
- Aquire more keys by observing a group chat. List your keys with
  `n2o keylist`
  from within the chat.
- Change networks with your Android and watch that the chat still works.

Try it — Get another Key

- To get my key, search for it (32 bit is sufficient)
  `keysearch kQusJ`
- Try to chat with me
  `chat 32c3@bernd`
- Aquire more keys by observing a group chat. List your keys with
  `n2o keylist`
  from within the chat.
- Change networks with your Android and watch that the chat still works.

Try it — Get another Key

- To get my key, search for it (32 bit is sufficient)
  `keysearch kQusJ`
- Try to chat with me
  `chat 32c3@bernd`
- Aquire more keys by observing a group chat. List your keys with
  `n2o keylist`
  from within the chat.
- Change networks with your Android and watch that the chat still works.

Content or Apps?

- The current web is defined by content — web apps (JavaScript) are an afterthough
- Therefore, the application logic is usually on the server side
- This doesn't work for a P2P network!
- Content is structured text, images, videos, music, etc.

Content or Apps?

- The current web is defined by content — web apps (JavaScript) are an afterthough
- Therefore, the application logic is usually on the server side
- This doesn't work for a P2P network!
- Content is structured text, images, videos, music, etc.

Content or Apps?

- The current web is defined by content — web apps (JavaScript) are an afterthough
- Therefore, the application logic is usually on the server side
- This doesn't work for a P2P network!
- Content is structured text, images, videos, music, etc.

Content or Apps?

- The current web is defined by content — web apps (JavaScript) are an afterthough
- Therefore, the application logic is usually on the server side
- This doesn't work for a P2P network!
- Content is structured text, images, videos, music, etc.

App–Centric World

- There's a phenomenon I call "Turing creep": Every sufficiently complex system contains a user–accessible Turing–complete language
- Corollary: Every efficient sufficiently complex system can execute native machine code
- The application logic is to present the data; data itself is as above: structured text, images, videos, music, etc.
- Executing (especially efficient) code from the net raises obvious questions about security

App–Centric World

- There's a phenomenon I call "Turing creep": Every sufficiently complex system contains a user–accessible Turing–complete language
- Corollary: Every efficient sufficiently complex system can execute native machine code
- The application logic is to present the data; data itself is as above: structured text, images, videos, music, etc.
- Executing (especially efficient) code from the net raises obvious questions about security

App–Centric World

- There's a phenomenon I call "Turing creep": Every sufficiently complex system contains a user–accessible Turing–complete language
- Corollary: Every efficient sufficiently complex system can execute native machine code
- The application logic is to present the data; data itself is as above: structured text, images, videos, music, etc.
- Executing (especially efficient) code from the net raises obvious questions about security

App–Centric World

- There's a phenomenon I call "Turing creep": Every sufficiently complex system contains a user–accessible Turing–complete language
- Corollary: Every efficient sufficiently complex system can execute native machine code
- The application logic is to present the data; data itself is as above: structured text, images, videos, music, etc.
- Executing (especially efficient) code from the net raises obvious questions about security

How to securely execute code?

There are several options tried; as usual, things are broken:

1. Execute code in a controlled secure VM, see for example Java. This is broken by design, as securing something from the inside doesn't work.
2. Execute code in a sandbox. This has shown as more robust, depending on how complex the outside of the sandbox is.
3. Public inspection of code. This is how the open source world works, but the underhanded C contest shows that inspection is tricky.
4. Scan for known evil code. This is the security industry's approach, and it is not working.
5. Code signing can work together with public inspection — but using it for accountability doesn't work

Therefore the choice is to sandbox public inspected code.

How to securely execute code?

There are several options tried; as usual, things are broken:

1. Execute code in a controlled secure VM, see for example Java. This is broken by design, as securing something from the inside doesn't work.
2. Execute code in a sandbox. This has shown as more robust, depending on how complex the outside of the sandbox is.
3. Public inspection of code. This is how the open source world works, but the underhanded C contest shows that inspection is tricky.
4. Scan for known evil code. This is the security industry's approach, and it is not working.
5. Code signing can work together with public inspection — but using it for accountability doesn't work

Therefore the choice is to sandbox public inspected code.

How to securely execute code?

There are several options tried; as usual, things are broken:

1. Execute code in a controlled secure VM, see for example Java. This is broken by design, as securing something from the inside doesn't work.
2. Execute code in a sandbox. This has shown as more robust, depending on how complex the outside of the sandbox is.
3. Public inspection of code. This is how the open source world works, but the underhanded C contest shows that inspection is tricky.
4. Scan for known evil code. This is the security industry's approach, and it is not working.
5. Code signing can work together with public inspection — but using it for accountability doesn't work

Therefore the choice is to sandbox public inspected code.

How to securely execute code?

There are several options tried; as usual, things are broken:

1. Execute code in a controlled secure VM, see for example Java. This is broken by design, as securing something from the inside doesn't work.
2. Execute code in a sandbox. This has shown as more robust, depending on how complex the outside of the sandbox is.
3. Public inspection of code. This is how the open source world works, but the underhanded C contest shows that inspection is tricky.
4. Scan for known evil code. This is the security industry's approach, and it is not working.
5. Code signing can work together with public inspection — but using it for accountability doesn't work

Therefore the choice is to sandbox public inspected code.

How to securely execute code?

There are several options tried; as usual, things are broken:

1. Execute code in a controlled secure VM, see for example Java. This is broken by design, as securing something from the inside doesn't work.
2. Execute code in a sandbox. This has shown as more robust, depending on how complex the outside of the sandbox is.
3. Public inspection of code. This is how the open source world works, but the underhanded C contest shows that inspection is tricky.
4. Scan for known evil code. This is the security industry's approach, and it is not working.
5. Code signing can work together with public inspection — but using it for accountability doesn't work

Therefore the choice is to sandbox public inspected code.

How to securely execute code?

There are several options tried; as usual, things are broken:

1. Execute code in a controlled secure VM, see for example Java. This is broken by design, as securing something from the inside doesn't work.
2. Execute code in a sandbox. This has shown as more robust, depending on how complex the outside of the sandbox is.
3. Public inspection of code. This is how the open source world works, but the underhanded C contest shows that inspection is tricky.
4. Scan for known evil code. This is the security industry's approach, and it is not working.
5. Code signing can work together with public inspection — but using it for accountability doesn't work

Therefore the choice is to sandbox public inspected code.

How to securely execute code?

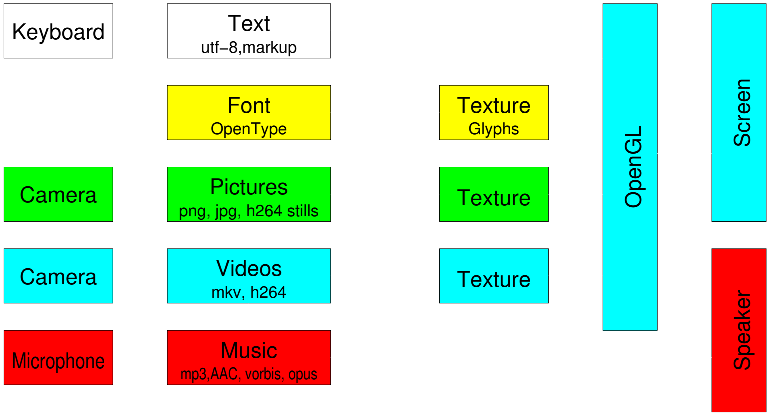
There are several options tried; as usual, things are broken:

1. Execute code in a controlled secure VM, see for example Java. This is broken by design, as securing something from the inside doesn't work.
2. Execute code in a sandbox. This has shown as more robust, depending on how complex the outside of the sandbox is.
3. Public inspection of code. This is how the open source world works, but the underhanded C contest shows that inspection is tricky.
4. Scan for known evil code. This is the security industry's approach, and it is not working.
5. Code signing can work together with public inspection — but using it for accountability doesn't work

Therefore the choice is to sandbox public inspected code.

Formats&Requirements

How to display things



Why OpenGL?

OpenGL can do everything

OpenGL renders:

1. Triangles, lines, points — simple components
2. Textures and gradients
3. and uses shader programs — the most powerful thing in OpenGL from 2.0.

Real requirement: visualization of *any* data. OpenGL can do that.

Why OpenGL?

OpenGL can do everything

OpenGL renders:

1. Triangles, lines, points — simple components
2. Textures and gradients
3. and uses shader programs — the most powerful thing in OpenGL from 2.0.

Real requirement: visualization of *any* data. OpenGL can do that.

Why OpenGL?

OpenGL can do everything

OpenGL renders:

1. Triangles, lines, points — simple components
2. Textures and gradients
3. and uses shader programs — the most powerful thing in OpenGL from 2.0.

Real requirement: visualization of *any* data. OpenGL can do that.

Why OpenGL?

OpenGL can do everything

OpenGL renders:

1. Triangles, lines, points — simple components
2. Textures and gradients
3. and uses shader programs — the most powerful thing in OpenGL from 2.0.

Real requirement: visualization of *any* data. OpenGL can do that.

Why OpenGL?

OpenGL can do everything

OpenGL renders:

1. Triangles, lines, points — simple components
2. Textures and gradients
3. and uses shader programs — the most powerful thing in OpenGL from 2.0.

Real requirement: visualization of *any* data. OpenGL can do that.

How to connect the media?

Lemma: every glue logic will become Turing complete

- currently used glue: HTML+CSS+JavaScript
- containers with Flash, Java, ActiveX, PDF, Google's NaCl...
- conclusion: use a powerful tool right from start!
- browser: run–time and development tool for applications

How to connect the media?

Lemma: every glue logic will become Turing complete

- currently used glue: HTML+CSS+JavaScript
- containers with Flash, Java, ActiveX, PDF, Google's NaCl...
- conclusion: use a powerful tool right from start!
- browser: run–time and development tool for applications

How to connect the media?

Lemma: every glue logic will become Turing complete

- currently used glue: HTML+CSS+JavaScript
- containers with Flash, Java, ActiveX, PDF, Google's NaCl...
- conclusion: use a powerful tool right from start!
- browser: run–time and development tool for applications

How to connect the media?

Lemma: every glue logic will become Turing complete

- currently used glue: HTML+CSS+JavaScript
- containers with Flash, Java, ActiveX, PDF, Google's NaCl...
- conclusion: use a powerful tool right from start!
- browser: run–time and development tool for applications

Frameworks

- libsoil for images (PNG+JPEG loading into a texture)
- freetype-gl for fonts (TrueType/OpenType into a texture)
- OpenMAX on Android, gstreamer on Linux: videos into a texture
- MINOΣ2: Lightweight OpenGL–based widget library in Forth (still a lot of work in progress)

Frameworks

- libsoil for images (PNG+JPEG loading into a texture)
- freetype-gl for fonts (TrueType/OpenType into a texture)
- OpenMAX on Android, gstreamer on Linux: videos into a texture
- MINOΣ2: Lightweight OpenGL–based widget library in Forth (still a lot of work in progress)

Frameworks

- libsoil for images (PNG+JPEG loading into a texture)
- freetype-gl for fonts (TrueType/OpenType into a texture)
- OpenMAX on Android, gstreamer on Linux: videos into a texture
- MINOΣ2: Lightweight OpenGL–based widget library in Forth (still a lot of work in progress)

Frameworks

- libsoil for images (PNG+JPEG loading into a texture)
- freetype-gl for fonts (TrueType/OpenType into a texture)
- OpenMAX on Android, gstreamer on Linux: videos into a texture
- MINOΣ2: Lightweight OpenGL–based widget library in Forth (still a lot of work in progress)

For Further Reading I

Bernd Paysan
net2o source repository and wiki
http://fossil.net2o.de/net2o