nldebug

David Draco

2008

Einsatzbereiche

- Threads
- Webapplikationen
- Komponenten / Kontext

Vererbungsbeispiel

```
 5 class A:
 6     a = 3
 7     ident = None
 8     def __init__(self, ident):
 9         self.ident = ident
10         udpdebug("A-%d: __init__"% self.ident)
11
12     def change(self, var):
13         udpdebug("A-%d: change()" % self.ident)
14         time.sleep(5)
15         self.a = self.a*2 + var
16         udpdebug("A-%d: change: set a to %s" % (self.ident,self.a))
17
18     def output(self):
19         udpdebug("A-%d: output()" % self.ident)
20         return self.a
21
22 class B(A):
23     def __init__(self, ident):
24         A.__init__(self, ident)
25         udpdebug("B-%d: __init__" % self.ident)
26         self.a = 15
27
28     def change(self, var=0):
29         udpdebug("B-%d: change()" % self.ident)
30         self.a = self.a*3 + var
31         udpdebug("B-%d: change: set a to %s" % (self.ident,self.a))
32
```

Wie kommen Daten rein?

- UDP-Pakete (bindings)
- ssh -v -v -v $LOGIN 2>&1 | ./sendlines.py (oder netcat)

Vorteile

- Daten nochmal anders betrachten
- detaillierter / übersichtlicher
- ohne Szenario nochmal durchspielen zu müssen
- auf anderem Rechner (headless)
- Live
- Timings messen
- Crashes betrachten

Vorteile

- Daten nochmal anders betrachten
- detaillierter / übersichtlicher
- ohne Szenario nochmal durchspielen zu müssen
- auf anderem Rechner (headless)
- Live
- Timings messen
- Crashes betrachten

Nicht-Nachteile

- UDP - kein Performancenachteil
- kann auch bestehende Apps analysieren
- Bindings für C, Python, PHP
- Daten einfach rein und raus
- bei Webanwendungen: unabhängig von HTML-Output

Neue Features

Stack traces! Kann betroffene Datei öffnen und zur Zeile springen